CS146 LBA: The Cost of Basic Goods

Huey Ning Lok

Minerva Schools @ KGI

Introduction

In this assignment, we model the cost of groceries in different neighborhoods of Berlin. Do product and store brands affect product prices, or not? Are grocery prices and the distribution of different grocery stores correlated with other cost of living measures in a city — for example, rent and real estate prices?

Importing and Pre-processing the Data

```
#import libraries
import pystan
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as sts
sns.set()


#import data
data = pd.read_csv('shop.csv',encoding='latin-1')

#preprocess data for product brands
#get list of headers for product brand names
headers = [i for i in list(data.head(0))[1:] if i[-5:] == 'brand']

#search for all brand names containing the term 'bio'
bio_search = ['Bio', 'bio', 'BIO']

for i in headers:
    #replace all brand names containing the term 'no brand' with 1
    data.loc[data[i].str.contains('no brand',na=False), i] = 1
    #replace all brand names containing the term 'bio' with 2
    data.loc[data[i].str.contains('|'.join(bio_search),na=False), i] = 2
    #replace every other brand name with 3
    data.loc[data[i].str.contains('.*?',na=False), i] = 3

#take a peak at the data
data.head()
```

	Grocery store brand	Area	Apple 1 brand	Apple 1 price	Apple 2 brand	Apple 2 price	Apple 3 brand	Apple 3 price	Banana 1 brand	Banana 1 price	...	Eggs 2 brand	Eggs 2 price	Eggs 3 brand	Eggs 3 price	Chicken 1 brand	Chicken 1 price	Chicken 2 brand	Chicken 2 price	Chicken 3 brand
0	1	1	3	2.49	3	2.49	1	1.40	1	1.09	...	3	2.02	1	3.30	3	6.48	NaN	0.00	NaN
1	3	2	3	2.72	3	1.39	3	2.99	1	1.09	...	2	3.30	3	1.29	3	6.98	3	7.48	NaN
2	4	7	3	1.99	3	2.99	3	1.99	1	1.09	...	3	2.38	2	4.98	1	6.98	NaN	NaN	NaN
3	4	4	3	1.99	3	2.93	3	1.99	3	0.88	...	3	3.38	3	1.55	3	9.99	2	25.99	3
4	3	6	3	1.79	3	1.39	3	2.74	2	1.65	...	3	2.03	3	1.55	2	19.99	3	6.48	NaN

```
#define numerical code for base product types and multipliers
store_dict = { 1: 'Lidl', 2: 'Rewe', 3: 'Aldi', 4: 'Edeka' }

area_dict = {
1:"Mitte",2:"Schoneberg",3:"Neukooln",4:"Kreuzberg",5:"Friedrichshain",
             6:"Prenzlauer Berg",7:"Tiergarten",8:"Alt-Tempelhof",9:"Wedding",
             10:"Gesundbrunnen",11:"Moabit",12:"Rummelsburg",13:"Lichtenberg" }

type_dict = {1:"Apple",2:"Banana",3:"Tomatoes",4:"Potatos",5:"Flour",
             6:"Rice",7:"Milk",8:"Butter",9:"Eggs",10:"Chicken"}

brand_dict = {1:"No brand",2:"Bio brand",3:"Other"}


#preprocess data into lists of length: number of observed prices collected
prices = []
stores = []
types = []
areas = []
brands = []

for i in range(1,11):
    product = type_dict[i]
    price1 = data[[product+' 1 price',product+' 1 brand','Grocery store
brand','Area']].dropna()
    price2 = data[[product+' 2 price',product+' 2 brand','Grocery store
brand','Area']].dropna()
    price3 = data[[product+' 3 price',product+' 3 brand','Grocery store
brand','Area']].dropna()

    product_prices = price1[product+' 1 price'].values.tolist() + price2[product+'
2 price'].values.tolist() + price3[product+' 3 price'].values.tolist()
    product_stores = price1['Grocery store brand'].values.tolist() +
price2['Grocery store brand'].values.tolist() + price3['Grocery store
brand'].values.tolist()
    product_type = len(product_prices)*[i]
    product_area = price1['Area'].values.tolist() + price2['Area'].values.tolist()
+ price3['Area'].values.tolist()
    product_brand = price1[product+' 1 brand'].values.tolist() + price2[product+' 2
brand'].values.tolist() + price3[product+' 3 brand'].values.tolist()

    prices += product_prices
    stores += product_stores
    types += product_type
    areas += product_area
    brands += product_brand
```

```
#take a peak at the preprocessed data
print(prices[0:10])
print(stores[0:10])
print(types[0:10])
print(areas[0:10])
print(brands[0:10])
```

```
[2.49, 2.72, 1.99, 1.99, 1.79, 3.13, 1.99, 4.38, 1.52, 1.99]
[1, 3, 4, 4, 3, 1, 4, 3, 2, 4]
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
[1, 2, 7, 4, 6, 1, 1, 3, 4, 3]
[3, 3, 3, 3, 3, 2, 3, 2, 3, 3]
```

Building the Stan model[1]

```
#Build the stan model
stan_code = """

// The data block contains all known quantities - typically the observed
// data and any constant hyperparameters.
data {

    int<lower=1> N;           // number of observed prices collected
    int<lower=1> T;           // number of different types of products
    int<lower=1> S;           // number of different types of stores
    int<lower=1> A;           // number of different types of areas
    int<lower=1> B;           // number of different types of brands

    // data collected
    real<lower=0> prices[N];
    int types[N];
    int stores[N];
    int areas[N];
    int brands[N];

    // fixed prior hyperparameters
    real<lower=0> alpha;
    real<lower=0> beta;

}

// The parameters block contains all unknown quantities - typically the
// parameters of the model. Stan will generate samples from the posterior
```

[1] **#probability:** I used Pystan to calculate posterior probability distributions over the base prices of different goods and store, brand and location multipliers.

```
// distributions over all parameters.
parameters {

    real<lower=0> base_price[T];       //vector containing base prices (number of
base prices == number of product types)
    real<lower=0> multiplier_store[S]; //vector containing store multipliers
    real<lower=0> multiplier_area[A];  //vector containing area multipliers
    real<lower=0> multiplier_brand[B];  //vector containing brand multipliers
    real<lower=0> sigma2;              //random noise when sampling from the normal
distribution for observed prices

}


// The model block contains all probability distributions in the model.
// This of this as specifying the generative model for the scenario.
model {

    sigma2 ~ gamma(alpha, beta);      //generate random noise for normal
distribution

    for (i in 1:T) {
        base_price[types[i]] ~ cauchy(1.5, 2);         //generate base price
    };

    for (i in 1:S) {
        multiplier_store[stores[i]] ~ cauchy(1, 0.7);  //generate store multiplier
    };

    for (i in 1:A) {
        multiplier_area[areas[i]] ~ cauchy(1, 0.7);    //generate area multiplier
    };

    for (i in 1:B) {
        multiplier_brand[brands[i]] ~ cauchy(1, 0.7);    //generate brand
multiplier
    };

    for(i in 1:N) {
        prices[i] ~ normal(base_price[types[i]] * multiplier_store[stores[i]] *
multiplier_area[areas[i]] * multiplier_brand[brands[i]], sqrt(sigma2));
//likelihood function
  }

}

"""
```

```
#compile the model
stan_model = pystan.StanModel(model_code=stan_code)

#stan input data
stan_data = {
                'prices': prices,
                'types': types,
                'stores': stores,
                'areas': areas,
                'brands': brands,
                'N': len(prices),
                'T': len(set(types)),
                'S': len(set(stores)),
                'A': len(set(areas)),
                'B': len(set(brands)),
                'alpha': 1.5,
                'beta': 7
            }

# Fitting stan model to the data. This will generate samples from the posterior
over all parameters of the model.
stan_results = stan_model.sampling(data=stan_data)
print(stan_results)

# Extract the generated samples from the stan model
posterior_samples = stan_results.extract()
```

Stan Results

```
Inference for Stan model: anon_model_5bb11c484d7256e1f57d5440d8ac344e.
4 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.
```

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
base_price[1]	2.14	0.01	0.45	1.34	1.85	2.11	2.4	3.18	938	1.0
base_price[2]	1.26	9.6e-3	0.3	0.77	1.05	1.23	1.42	1.94	971	1.0
base_price[3]	3.23	0.02	0.72	1.99	2.76	3.16	3.61	4.91	910	1.0
base_price[4]	1.13	8.6e-3	0.27	0.68	0.94	1.1	1.27	1.79	1032	1.0
base_price[5]	1.06	8.2e-3	0.27	0.61	0.88	1.03	1.21	1.67	1071	1.0
base_price[6]	2.86	0.02	0.64	1.76	2.43	2.8	3.2	4.33	902	1.0
base_price[7]	0.95	7.1e-3	0.25	0.56	0.78	0.93	1.08	1.53	1202	1.0
base_price[8]	4.2	0.03	0.93	2.58	3.59	4.11	4.69	6.35	883	1.0
base_price[9]	2.56	0.02	0.57	1.57	2.19	2.51	2.87	3.88	907	1.0
base_price[10]	10.56	0.08	2.32	6.56	9.05	10.38	11.8	15.85	892	1.0
multiplier_store[1]	0.71	5.9e-3	0.19	0.38	0.58	0.69	0.83	1.14	1089	1.0
multiplier_store[2]	1.07	8.7e-3	0.29	0.58	0.87	1.04	1.24	1.71	1118	1.0
multiplier_store[3]	0.96	7.9e-3	0.26	0.52	0.78	0.93	1.11	1.54	1107	1.0
multiplier_store[4]	1.06	8.6e-3	0.29	0.57	0.86	1.03	1.23	1.7	1114	1.0
multiplier_area[1]	1.33	5.7e-3	0.18	1.02	1.21	1.32	1.44	1.73	976	1.0
multiplier_area[2]	0.91	4.2e-3	0.14	0.66	0.81	0.9	0.99	1.21	1137	1.0
multiplier_area[3]	0.92	4.0e-3	0.13	0.7	0.83	0.91	1.0	1.2	984	1.0
multiplier_area[4]	1.29	5.5e-3	0.17	0.99	1.17	1.28	1.4	1.67	977	1.0
multiplier_area[5]	1.31	5.6e-3	0.18	1.01	1.19	1.3	1.42	1.71	986	1.0
multiplier_area[6]	1.17	5.0e-3	0.16	0.88	1.05	1.15	1.26	1.52	1034	1.0
multiplier_area[7]	1.12	4.8e-3	0.16	0.85	1.01	1.11	1.22	1.47	1097	1.0
multiplier_area[8]	0.93	4.0e-3	0.14	0.68	0.83	0.92	1.02	1.23	1279	1.0
multiplier_area[9]	1.49	6.9e-3	0.27	1.02	1.31	1.47	1.66	2.06	1491	1.0
multiplier_area[10]	0.91	4.7e-3	0.17	0.62	0.79	0.9	1.02	1.27	1259	1.0
multiplier_area[11]	1.41	6.1e-3	0.2	1.06	1.27	1.4	1.54	1.85	1068	1.0
multiplier_area[12]	1.13	5.3e-3	0.18	0.83	1.01	1.12	1.24	1.52	1123	1.0
multiplier_area[13]	1.0	4.9e-3	0.19	0.67	0.87	0.99	1.11	1.42	1438	1.0
multiplier_brand[1]	0.89	7.4e-3	0.25	0.44	0.72	0.88	1.04	1.46	1196	1.0
multiplier_brand[2]	1.61	0.01	0.45	0.81	1.3	1.59	1.87	2.6	1187	1.0
multiplier_brand[3]	0.86	7.0e-3	0.24	0.43	0.7	0.85	1.01	1.41	1188	1.0
sigma2	2.52	1.8e-3	0.09	2.34	2.46	2.52	2.58	2.71	2661	1.0
lp__	-1371	0.12	4.14	-1381	-1374	-1371	-1368	-1364	1262	1.0

```
Samples were drawn using NUTS at Fri Nov  9 20:54:25 2018.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

Modeling Assumptions

The half-Cauchy distribution was chosen as the generative prior to the base prices, store multipliers, area multipliers, and brand multipliers. This choice was motivated by the Cauchy's fat tails, which represent a broad prior distribution where the probabilities assigned to values moving away from the mean approach 0 at a slower rate.

The store, area, and brand multipliers are centered around 1 to represent an average multiplier effect of 1. The scale parameter chosen was 0.7 to represent an averagely broad prior.

The base price prior half-Cauchy was centered around 1.5 since the data was cleaned to only contain stores from Berlin, and so a stable currency unit of Euro was safely assumed. Furthermore, the products consist of everyday goods that one would not expect to exceed 10 euros at the highest. The scale parameter chosen was 2 to represent a broader prior, since there can be a large difference in the price range between meat and vegetables.

The normal distribution was chosen as the likelihood function, with the equation: $f(x) = \text{base\_price} \times \text{multiplier\_store} \times \text{multiplier\_area} \times \text{multiplier\_brand}$ as the mean and σ as the standard deviation. We expect the observed prices to obey the Central Limit Theorem, which would make the normal distribution a good choice. The noise variable, σ, was generated from a gamma distribution with $\alpha=1.5$ and $\beta=7$. This distribution had a mean of 0.2143 and a standard deviation of 0.175, which I believed was a good representation of the uncertainty within the noise levels.[2]

Other Assumptions

Human error

Some variation in the data will be purely due to human error. The data was collected by different individuals who probably went about the process with different methodologies. Furthermore, while instructions were given to record prices according to set units (1kg, 1 dozen, etc.), there is a high chance of omitting this step. Some of the data could also have been falsely constructed since there is no guarantee that the individuals actually carried out the data collection process.

Product brand categorization

The product brands were separated into 3 distinct categories: No brand, Bio brand, and Other. It was assumed that all brands that contained the term "bio" would most likely share similar traits, and so they were placed within the same category. Due to the large variety of other brands, the

[2] **#distributions:** I identified appropriate distributions given the context of possible multiplier ranges and base prices of grocery goods, and justify my choices for each distribution chosen.

"non-bio" brands were categorized together as "Other". Products with no brand recorded were categorized as "No brand".

Analysis and Discussion[3]

The basic average price for each product:[4]

	Label	Mean	Posterior 95% confidence interval	Standard Error
0	Apple	2.14	[1.3438124035880277, 3.1752347199042834]	0.45
1	Banana	1.26	[0.7674075352667533, 1.936799162991015]	0.30
2	Tomatoes	3.23	[1.9935784046017044, 4.912630759996475]	0.72
3	Potatos	1.13	[0.6770802913648523, 1.7870121489438746]	0.27
4	Flour	1.06	[0.6086166827865962, 1.6691284192321152]	0.27
5	Rice	2.86	[1.7575618518950806, 4.327238050964559]	0.64
6	Milk	0.95	[0.5566670426258286, 1.5315904463745809]	0.25
7	Butter	4.20	[2.5863637962589325, 6.348694749188116]	0.93
8	Eggs	2.56	[1.571756039520534, 3.8675383287184304]	0.57
9	Chicken	10.56	[6.563774283743646, 15.849961589948958]	2.32

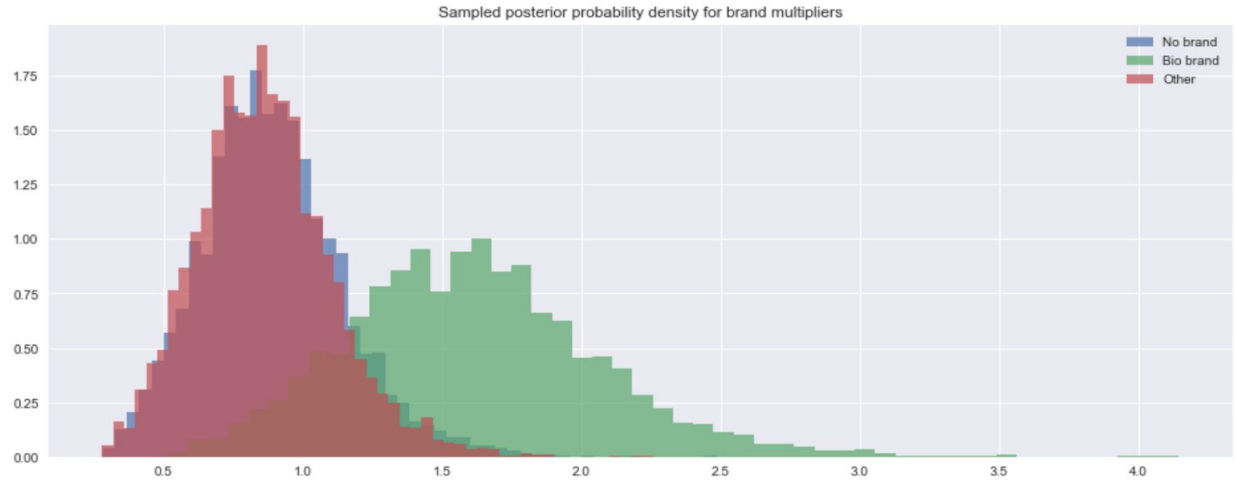
[3] **#confidenceintervals:** I incorporated 95% confidence intervals into my table results as I understood that while the mean of the distributions gives us a simple-to-reference metric of the distribution's average value, the confidence interval showed us how certain we are of our distribution by providing the range of values that we can be 95% confident that the population mean lies within. For example, we can be more certain of the values for milk and potatoes due to the small CI interval and less certain of the values for chicken due to the wide CI interval.

[4] All code for the graphs can be found in the attached Jupyter notebook or the GitHub link in the Appendix.

Effect of Various Factors on the Basic Price of the Product

Brand of the Product

	Label	Mean	Posterior 95% confidence interval	Standard Error
0	No brand	0.89	[0.44171754143918873, 1.455343933107465]	0.25
1	Bio brand	1.61	[0.811814044350757, 2.602807826972428]	0.45
2	Other	0.86	[0.43480697154335246, 1.4110550263669392]	0.24



From the graph and table above, we see that Bio brand has the highest mean multiplier effect of 1.59, whereas No brand and Other brands have a virtually similar mean multiplier effect of 0.88 and 0.85 respectively. This can also be visually inferred through the graph - the posterior distributions for No brand and Other have a high area of overlap.

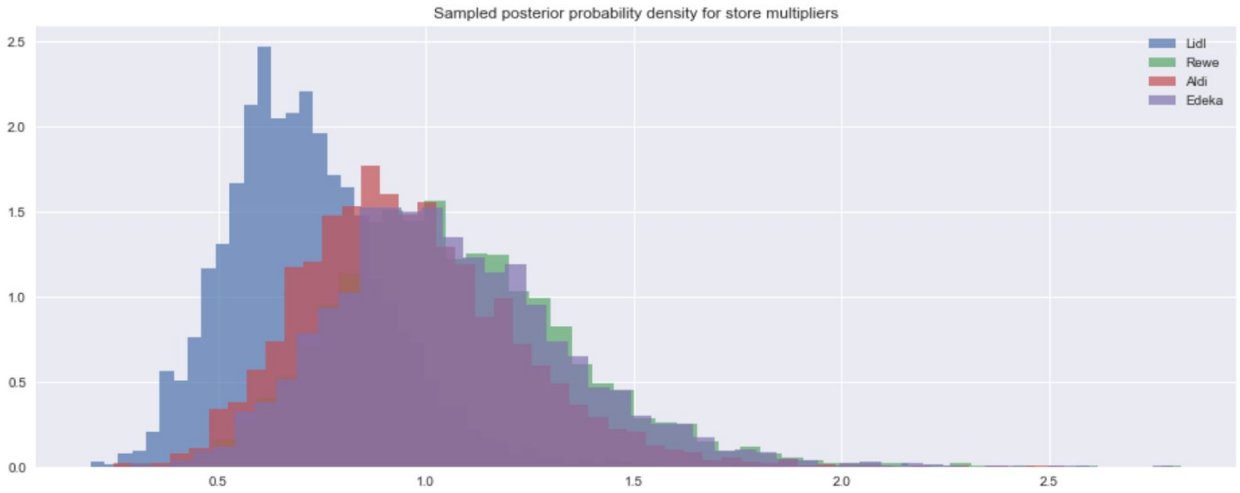
T-tests for the brand multipliers:[5]

- No brand is significantly cheaper than Bio brand
- No brand is significantly more expensive than Other
- Bio brand is significantly more expensive than No brand
- Bio brand is significantly more expensive than Other
- Other is significantly cheaper than No brand
- Other is significantly cheaper than Bio brand

[5] **#significance:** Although intuitive inferences can be made from observing the plot of distributions, I used t-tests to analytically calculate the statistical significance of the difference in multiplier effects, where the p-value is set at 0.05. The t-test is arguably a more accurate way of determining the difference in multiplier effects since it considers the sample size and variance of the datasets. The t-test is used here since there are a small number of comparisons to be made, and so the significance statements can be easily interpreted.

- **Brand of the Grocery Store**

	Label	Mean	Posterior 95% confidence interval	Standard Error
0	Lidl	0.71	[0.38259625432635913, 1.1408046592011876]	0.19
1	Rewe	1.07	[0.5775966099935697, 1.7094354067584412]	0.29
2	Aldi	0.96	[0.5181674656130385, 1.5403191775543301]	0.26
3	Edeka	1.06	[0.573710977932385, 1.695998657431865]	0.29



From the graph and table above, we see that Rewe and Edeka have the highest mean multipliers of 1.08 and 1.07 respectively, whereas Lidl has the lowest mean multiplier of 0.72.

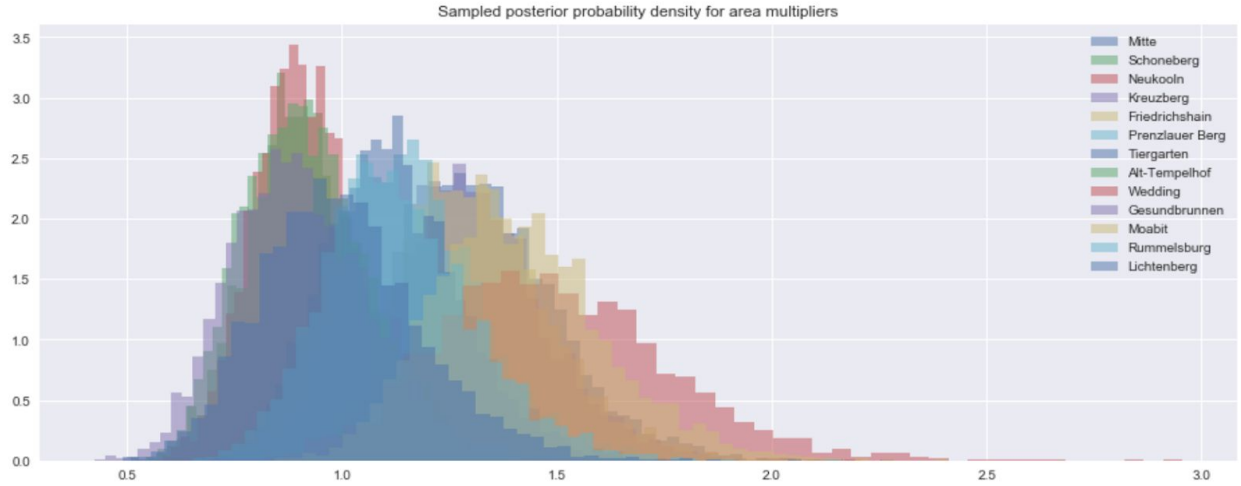
There is a high overlap area between the posterior distributions for Rewe, Edeka, and Aldi, whereas Lidl's distribution mean can be observed to peak to the left of the other 3 distributions.

T-tests for the store multipliers:

- Lidl is significantly cheaper than Rewe
- Lidl is significantly cheaper than Aldi
- Lidl is significantly cheaper than Edeka
- Rewe is significantly more expensive than Lidl
- Rewe is significantly more expensive than Aldi
- Rewe is not significantly more expensive than Edeka
- Aldi is significantly more expensive than Lidl
- Aldi is significantly cheaper than Rewe
- Aldi is significantly cheaper than Edeka
- Edeka is significantly more expensive than Lidl
- Edeka is not significantly more expensive than Rewe
- Edeka is significantly more expensive than Aldi

Geographical Location of the Grocery Store

	Label	Mean	Posterior 95% confidence interval	Standard Error
0	Mitte	1.33	[1.021426098355916, 1.7260595977315791]	0.18
1	Schoneberg	0.91	[0.661773206422215, 1.2140049972006672]	0.14
2	Neukooln	0.92	[0.7000744201598195, 1.1997355957149505]	0.13
3	Kreuzberg	1.29	[0.9938335295227927, 1.673201632422295]	0.17
4	Friedrichshain	1.31	[1.0057791058154884, 1.7081732372165321]	0.18
5	Prenzlauer Berg	1.17	[0.8852388347435959, 1.5210793214527463]	0.16
6	Tiergarten	1.12	[0.8459173293309369, 1.469633466503138]	0.16
7	Alt-Tempelhof	0.93	[0.6803380339020632, 1.2335126270519339]	0.14
8	Wedding	1.49	[1.0199318706012168, 2.055921314190519]	0.27
9	Gesundbrunnen	0.91	[0.6202692285588376, 1.2651782893379189]	0.17
10	Moabit	1.41	[1.0637527354903307, 1.8527699526070667]	0.20
11	Rummelsburg	1.13	[0.8270392061931661, 1.5240312814105796]	0.18
12	Lichtenberg	1.00	[0.6738777205488673, 1.4144551706208948]	0.19



From the graph and the table, we see that most of the neighborhood multiplier posterior distributions overlap, and the multiplier means lie within the range of 0.89 (Schoneberg & Gesundbrunnen) on the low end, and 1.47 (Wedding) on the high end.[6]

[6] **#dataviz:** I plotted the distributions and used their graphical positions, spread and overlap in order to gain intuitive insights and inferences into the significance of various multiplier effects. While the table provides relevant descriptive statistics of the distributions (mean, 95% CI and standard error), by plotting my results graphically, I am able to easily convey the message of my findings to an audience, and better understand it myself.

The Strength of Brand and Location Multiplier Effects and their Influence on Price Variation between Shops

The strength of the brand and location effects have been illustrated above, with a higher mean multiplier representing a stronger multiplier effect, and a lower mean multiplier representing a weaker multiplier effect. The mean is used to represent the multiplier effect with an understanding that the certainty of the mean value varies with the distribution variance, and so the multiplier effect differences have to be interpreted within the appropriate context of variance and 95% CIs as shown in the tables above. The summary is as follows:

Brand multiplier

- Highest multiplier mean for brand: Bio brands (1.59)
- Lowest multiplier mean for brand: Other (0.85)
- The range of the means is 1.59 - 0.85 = 0.74. This means there might be up to a 74% difference in mean price depending on whether you buy no brand, bio brand, or other.

Store multiplier

- Highest multiplier mean for store: Rewe (1.08)
- Lowest multiplier mean for store: Lidl (0.72)
- The range of the means is 1.08 - 0.72 = 0.36. This means there might be up to a 36% difference in mean price depending on whether you buy from Lidl, Rewe, Aldi or Edeka.

Area multiplier

- Highest multiplier mean for area: Wedding (1.47)
- Lowest multiplier mean for area: Schoneberg & Gesundbrunnen (0.89)
- The range of the means is 1.47 - 0.89 = 0.58. This means there might be up to a 58% difference in mean price depending on which neighborhood the product is located within.

Assuming that the means of the various distributions can be safely generalized to represent the multiplier effect (despite differing variances), the brand multiplier has the greatest influence on price variation between shops (range of means = 0.74), whereas the store multiplier has the lowest influence on price variation between shops (range of means = 0.36).[7]

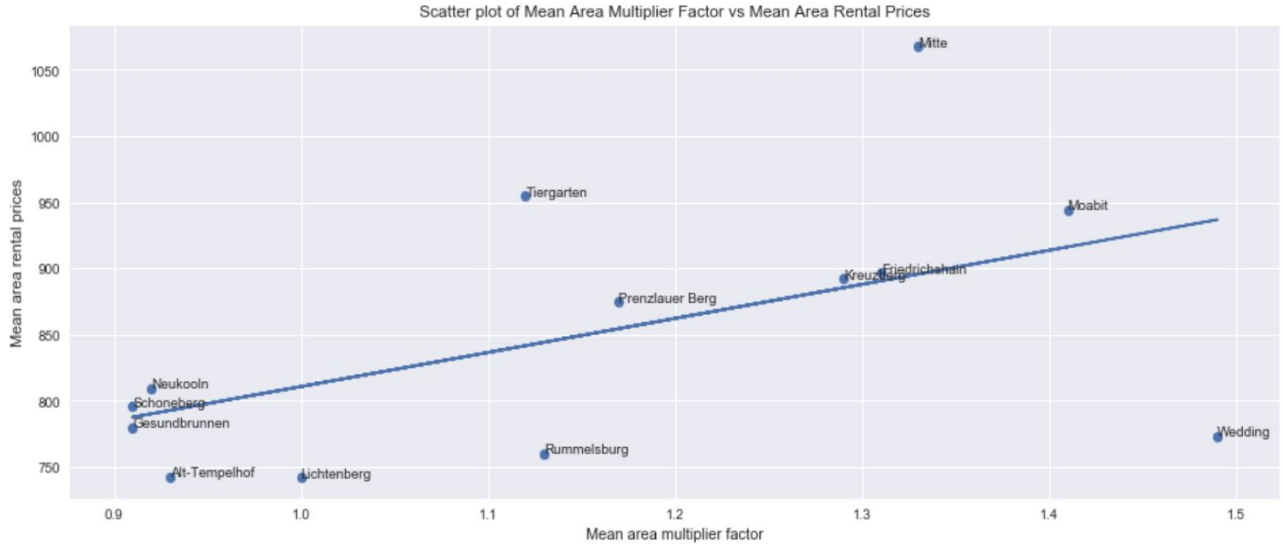
[7] **#descriptivestats:** I considered the relevant descriptive statistics calculated in the earlier section when justifying the method that I chose to measure the difference between multiplier effects. I acknowledged that using the distribution mean alone to represent the multiplier effect could only provide a rough estimation of the difference in multiplier effects, and I encourage the audience to refer to the variance and confidence intervals in the tables above in order to determine the certainty of the estimation. I intentionally avoided using a more analytical method of measuring difference significance, e.g. t-tests, due to the high number of comparisons that would need to be made, which could potentially decrease the interpretability of my results.

Correlation of Price Variation by Geographical Location vs Rental Price Variation in Berlin

Rental price data were obtained from the 2017 map of rental prices by UBahn and SBahn station: https://www.immobilienscout24.de/content/dam/is24/ibw/dokumente/mietmap-berlin-2017.jpg

	Label	Mean	Rental price means
0	Mitte	1.33	1067.60
1	Schoneberg	0.91	795.80
2	Neukooln	0.92	809.60
3	Kreuzberg	1.29	892.60
4	Friedrichshain	1.31	896.60
5	Prenzlauer Berg	1.17	874.50
6	Tiergarten	1.12	954.67
7	Alt-Tempelhof	0.93	742.67
8	Wedding	1.49	773.33
9	Gesundbrunnen	0.91	779.80
10	Moabit	1.41	944.50
11	Rummelsburg	1.13	760.00
12	Lichtenberg	1.00	742.33

```
Pearson Correlation Coefficient: 0.5299180711176752
```



The mean area rental prices and mean area multipliers show a positive correlation, with a Pearson Correlation Coefficient of 0.54.

Some of the more prominent outliers include Wedding and Mitte, whose x,y points are (1.47,773.33) and (1.32, 1067.60) respectively. These two neighborhoods lie on opposite ends of the spectrum - with Wedding having an unusually low mean rental price given its corresponding multiplier effect, and Mitte having an unusually high rental price given its corresponding multiplier effect.

While the positive correlation between area multiplier factor and area rental price seems intuitive, it has to be noted that the data collection method for the rental prices were rather crude. I manually picked prices from the map based on googling train stations that were closest to a given neighborhood, and selected the 5 closest neighboring stations to get the neighborhood's mean rental price. In cases where the train stations were rather isolated from other stations (low train station density), I would select fewer stations for the calculation of the average rental price.[8]

[8] **#correlation:** I identified the positive correlation between mean area multiplier factor and mean area rental prices by calculating the Pearson correlation coefficient. I noted that the positive correlation could be weaker than observed due to the shallow data collection method. The small number of data points means that the outliers might not be irregularities, but simply neighborhoods that are underrepresented in the data. As such, it might be appropriate to assign the outliers a higher weight, which would alter the correlation direction. Furthermore, the dataset only consists of a small number of neighborhoods. An extension of the linear regression line might reveal a different trend.

Appendix

Code:
https://github.com/hueyning/CS146-repo/blob/master/Lba/CS146%20Location%C2%AD-based%20Assignment.ipynb

Raw data:
https://github.com/hueyning/CS146-repo/blob/master/Lba/shop.csv

Store: ALDI, Ostseestraße 25
Visited: 10/27/2018 13:24:06

Store: EDEKA, Annenstraße
Visited: 10/29/2018 13:55:00