Dennis Kageni

CS146 LBA: The Cost of Basic Goods

Fall 2019

Data Pre-processing

Most of the data cleaning was conducted on Google Sheets. The imported dataset `data` contains data from the "LBA data gathering" spreadsheet where the columns 'Timestamp', 'Email address' and 'Your name' dropped and the product headers have been renamed. An additional column 'Location' has been added and contains information on each store's neighborhood from the "LBA data collection" spreadsheet. Lastly, I normalized the prices for each product and removed the data from Lotte Mart as it only represents one data point from South Korea and would, therefore, be an outlier.

```
#importing the required libraries
import pystan
import numpy as np
import pandas as pd
import scipy.stats as sts
from scipy import stats
import matplotlib.pyplot as plt
```

```
#loading and previewing grocery store data
data =
pd.read_csv("https://docs.google.com/spreadsheets/d/e/2PACX-1vTHbhLaXV_zjMeHN31IWY_99TZAERoAT0hlKm2dxbmS4zBCXgy_EAMSBOGe1IM5ye8B02G_N6T-zy7b/pub?gid=0&single=true&output=csv")
data.head()

## Since we'll be feeding this dataset into Stan, we have to convert all string variables into numbers ##

data_1 = data.copy() #copy data and save it into a new dataframe

#create two dictionaries, store_dict and location_dict where I'll assign
#numbers to the strings and map them into the `data_1` dataframe
store_dict = { "ALDI": 1, "EDEKA":2, "Lidl":3, "REWE":4, "Safeway": 5,
              "Sainsbury's": 6, "Tesco Express": 7, "Waitrose & Partners":8}

data_1['Store'] = data_1['Store'].map(store_dict)
```

```
location_dict = { "Alt-Treptow": 1, "Barbican": 2, "Clapton": 3, "Crouch End":
4, "Elephants and Castle": 5,
                 "Farringdon": 6, "Fitzrovia": 7, "Friedrichshain": 8,
"Hoxton": 9, "Islington": 10,
                 "Kreuzberg": 11, "Lichtenberg": 12, "Marylebone": 13,
"Mayfair": 14, "Mitte": 15, "Mountain View": 16,
                 "Neukölln": 17, "Palo Alto": 18, "Prenzlauer Berg": 19, "Saint
Pancras": 20, "San Francisco": 21,
                 "Schöneberg": 22, "Shoreditch": 23, "Tempelhof": 24}

data_1['Location'] = data_1['Location'].map(location_dict)
```

As we will be feeding this data into PyStan, it is important to ensure that:

1. There are no strings in the dataset
2. All nan values are removed

The next step is to reshape `data_1` such that the resulting data is tidy (that is, the dataset is arranged such that each variable is a column and each observation is a row). The underlying assumption that makes this possible is that although we were collecting different brands of the same product, we did not record the information about the brand name. Therefore, we assume that the three data points all relate to the same product which allows us to combine every observation of the variable into one column. The result of tidying the data is the data frame `lba_data` (Table 1).

```
matrix = data_1.iloc[:,2:]
columns = matrix.columns
columns = [x.split('_')[1] for x in columns]

columns_ = []
for i in range(0,30,3):
    columns_.append(columns[i])

num_samples = data_1.shape[0]
new_arr = np.empty((num_samples*3,len(columns_)))
for col in range(len(columns_)):
```

```
    item =  matrix.iloc[:,col*3:col*3+3].values
    for i in range(3):
        new_arr[i*num_samples:num_samples*(i+1),col] = item[:,i]

lba_data = pd.DataFrame()
stores =  data_1['Store'].values.tolist()
lba_data['Store'] = stores*3
locations = data_1['Location'].values.tolist()
lba_data['Location'] = locations*3

i = 0
for col in columns_:
    lba_data[col] = new_arr[:,i]
    i+=1

lba_data = lba_data.dropna() # delete NA values
lba_data.head()
```

	Store	Location	Apple	Banana	Tomato	Potato	Flour	Rice	Milk	Butter	Eggs	Chicken
0	1	12	2.50	1.69	3.52	0.56	0.39	1.99	0.99	5.56	0.12	6.65
1	1	17	2.99	1.15	3.58	0.56	0.39	3.98	1.05	9.56	0.16	6.65
2	1	17	1.25	1.15	1.99	0.46	0.39	1.99	0.63	5.56	0.16	6.65
3	1	11	2.99	1.15	6.30	0.60	0.39	1.99	1.05	9.56	0.16	6.65
4	1	11	3.25	1.69	1.99	0.58	0.39	0.98	0.71	6.36	0.16	8.54

Table 1. First 5 rows of `lba_data` dataframe after data cleaning and preprocessing

PyStan Model

The model contains the following parameters:

- a base price for each product (base_prices)
- a multiplier for each store brand (grocery_store_multiplier)
- a multiplier for the geographical name (location_multiplier) where I used Berlin's administrative divisions.

The base price and multipliers were drawn from a Cauchy distribution[1]. Since we're dealing with survey data that may be prone to data entry error (including but not limited to fabricated data, using the wrong units), the distribution allows us to accommodate possible outliers yet concentrate the mass around the median values. In the context of the base prices, this is ideal since the normalized average prices of the items in our baskets ranged from €0.36 to €9.72. Therefore, we center the distribution at 5 with a scale parameter of 2. The scale parameter gives us a broader prior and allows us to account for the variance in price differences between different products; say chicken and apples.

In the context of the grocery store and location multipliers, a Cauchy distribution is a safe choice since we have little information about the multipliers. In both cases, we use a Cauchy distribution centered at 1 (as mentioned in the assignment prompt) to achieve an average multiplier of 1 and a scale parameter of 0.5 to represent a reasonably broad prior. I used the normal distribution as the likelihood function such that the prices are normally distributed, with a mean (μ) of base price * grocery store multiplier * location multiplier and a standard deviation (σ). Here, I use a Cauchy prior centered at 0 and with a scale parameter of 1.

```
#stan code[2]
stan_code = """
// The data block contains all known quantities - typically the observed
// data and any constant hyperparameters.

data {
  int<lower=0> P;         // the number of different types of products
  int<lower=0> N;         // the total number of prices observations collected
for each product
  int<lower=0> L;         // total number of locations in the data
  int<lower=0> S;         // total number of stores in the data
```

[1] #distributions
[2] #modeling

```
  // data collected
  real<lower=0>  prices[P, N];
  int stores[N];          // list of grocery stores for each observation
  int locations[N];       //list of locations for each observation


}

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

// we set the lower bound of all the parameters to 0 since
// the cauchy distribution's support includes negative numbers

parameters {
  real<lower=0>  base_prices[P];
  real<lower=0>  grocery_store_multiplier[S];
  real<lower=0>  location_multiplier[L];
  real<lower=0>  sigma;                          //random noise when sampling
from the normal distribution

}

// The model block contains all probability distributions in the model.
// This of this as specifying the generative model for the scenario.
model {
  sigma ~ cauchy(0,1);

  for (i in 1:P){
    base_prices[i] ~ cauchy(5, 2);
  };

  for (i in 1:S){
    grocery_store_multiplier[i] ~ cauchy(1, .5);
  };

  for (i in 1:L){
    location_multiplier[i] ~ cauchy(1, .5);
  };

  for (i in 1:P){
    for (j in 1:N){
      prices[i, j] ~ normal(base_prices[i]*
```

```
grocery_store_multiplier[stores[j]]*location_multiplier[locations[j]], sigma);
    }
  }

}
"""


stan_model = pystan.StanModel(model_code=stan_code)

prices = [lba_data['Apple'], lba_data['Banana'], lba_data['Tomato'],
lba_data['Potato'],
          lba_data['Flour'], lba_data['Rice'], lba_data['Milk'],
lba_data['Butter'],
          lba_data['Eggs'], lba_data['Chicken']]

#passing the data to the stan model above
stan_data = {
    'P': 10,
    'N': len(lba_data),
    'L': len(lba_data["Location"].unique()),
    'S': len(lba_data["Store"].unique()),
    'prices': prices,
    'stores':  lba_data['Store'],
    'locations': lba_data['Location']
    }
stan_results = stan_model.sampling(data=stan_data)
print(stan_results)
```

Stan Results

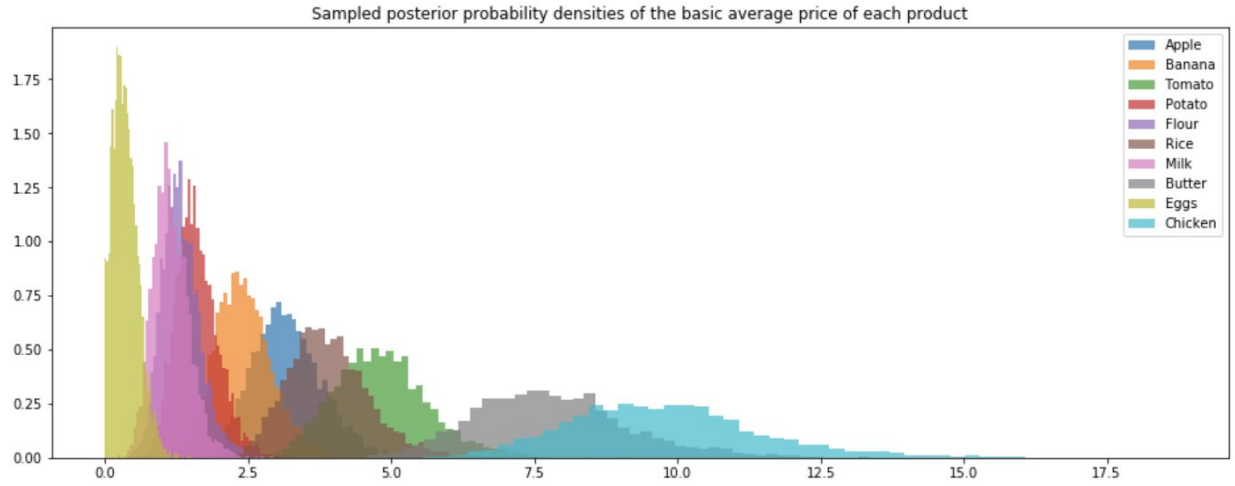
```
Inference for Stan model: anon_model_198d22b743323a232aa7f62915a1bd76.
4 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.
```

	mean	se_mean	sd	2.5%	25%	50%	75%	97.5%	n_eff	Rhat
base_prices[1]	3.19	0.02	0.62	2.17	2.73	3.13	3.56	4.59	895	1.0
base_prices[2]	2.41	0.02	0.5	1.58	2.06	2.35	2.71	3.57	1011	1.0
base_prices[3]	4.83	0.03	0.87	3.43	4.18	4.76	5.36	6.77	840	1.0
base_prices[4]	1.6	0.01	0.38	0.97	1.33	1.57	1.82	2.46	1221	1.0
base_prices[5]	1.33	9.3e-3	0.34	0.76	1.09	1.3	1.53	2.09	1353	1.0
base_prices[6]	3.89	0.02	0.72	2.72	3.37	3.82	4.32	5.47	868	1.0
base_prices[7]	1.15	8.3e-3	0.32	0.61	0.92	1.11	1.34	1.87	1471	1.0
base_prices[8]	7.77	0.05	1.4	5.55	6.75	7.62	8.6	11.04	835	1.0
base_prices[9]	0.37	4.3e-3	0.22	0.04	0.21	0.34	0.5	0.86	2564	1.0
base_prices[10]	9.74	0.06	1.75	6.94	8.48	9.53	10.78	13.74	821	1.0
grocery_store_multiplier[1]	0.75	4.6e-3	0.14	0.49	0.65	0.75	0.85	1.04	942	1.01
grocery_store_multiplier[2]	1.0	5.9e-3	0.18	0.65	0.88	1.0	1.13	1.37	931	1.0
grocery_store_multiplier[3]	0.67	4.1e-3	0.13	0.44	0.59	0.67	0.76	0.93	964	1.0
grocery_store_multiplier[4]	0.98	5.9e-3	0.18	0.64	0.85	0.97	1.1	1.33	921	1.0
grocery_store_multiplier[5]	1.08	7.5e-3	0.29	0.61	0.89	1.05	1.25	1.73	1449	1.0
grocery_store_multiplier[6]	1.21	7.0e-3	0.23	0.81	1.05	1.19	1.34	1.7	1079	1.0
grocery_store_multiplier[7]	0.7	4.3e-3	0.14	0.46	0.6	0.69	0.79	1.0	1070	1.0
grocery_store_multiplier[8]	0.97	5.5e-3	0.18	0.65	0.84	0.95	1.08	1.35	1092	1.0
location_multiplier[1]	0.95	3.9e-3	0.14	0.69	0.85	0.94	1.04	1.24	1325	1.0
location_multiplier[2]	1.07	3.7e-3	0.16	0.79	0.96	1.06	1.17	1.4	1782	1.0
location_multiplier[3]	0.95	4.4e-3	0.26	0.45	0.78	0.94	1.11	1.51	3488	1.0
location_multiplier[4]	1.13	4.1e-3	0.18	0.81	1.0	1.12	1.24	1.51	1903	1.0
location_multiplier[5]	0.64	3.4e-3	0.19	0.3	0.51	0.63	0.76	1.03	3105	1.0
location_multiplier[6]	1.2	4.1e-3	0.2	0.85	1.06	1.18	1.32	1.63	2297	1.0
location_multiplier[7]	0.99	3.6e-3	0.16	0.71	0.88	0.98	1.09	1.33	1934	1.0
location_multiplier[8]	0.99	4.0e-3	0.14	0.74	0.9	0.99	1.08	1.28	1191	1.0
location_multiplier[9]	1.07	4.6e-3	0.27	0.57	0.9	1.07	1.24	1.64	3360	1.0
location_multiplier[10]	1.21	4.2e-3	0.15	0.93	1.1	1.2	1.3	1.54	1357	1.0
location_multiplier[11]	1.08	4.3e-3	0.14	0.81	0.97	1.07	1.17	1.38	1135	1.0
location_multiplier[12]	1.01	4.0e-3	0.15	0.74	0.91	1.0	1.1	1.32	1398	1.0
location_multiplier[13]	1.21	4.3e-3	0.19	0.89	1.08	1.19	1.33	1.62	2012	1.0
location_multiplier[14]	1.22	4.6e-3	0.19	0.9	1.09	1.2	1.34	1.63	1647	1.0
location_multiplier[15]	1.11	4.6e-3	0.15	0.84	1.01	1.1	1.2	1.41	1027	1.0
location_multiplier[16]	1.02	5.9e-3	0.26	0.58	0.84	0.99	1.16	1.62	1898	1.0
location_multiplier[17]	1.17	4.7e-3	0.16	0.88	1.06	1.16	1.27	1.5	1121	1.0
location_multiplier[18]	1.41	8.4e-3	0.35	0.86	1.17	1.36	1.6	2.22	1768	1.0
location_multiplier[19]	1.09	4.3e-3	0.17	0.79	0.97	1.08	1.19	1.46	1557	1.0
location_multiplier[20]	1.01	3.7e-3	0.17	0.71	0.89	1.0	1.12	1.37	2079	1.0
location_multiplier[21]	1.02	5.9e-3	0.25	0.62	0.85	0.99	1.16	1.59	1770	1.0
location_multiplier[22]	0.9	3.6e-3	0.14	0.65	0.81	0.89	0.99	1.18	1431	1.0
location_multiplier[23]	1.2	4.6e-3	0.24	0.78	1.03	1.18	1.34	1.74	2727	1.0
location_multiplier[24]	0.99	4.8e-3	0.25	0.52	0.82	0.98	1.16	1.52	2702	1.0
sigma	2.94	8.1e-4	0.05	2.84	2.91	2.94	2.98	3.05	4489	1.0
lp__	-2458	0.13	4.86	-2469	-2461	-2458	-2455	-2450	1355	1.0

```
Samples were drawn using NUTS at Mon Nov 18 20:29:09 2019.
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).
```

Questions to Answer:

The basic average price for each product[3]



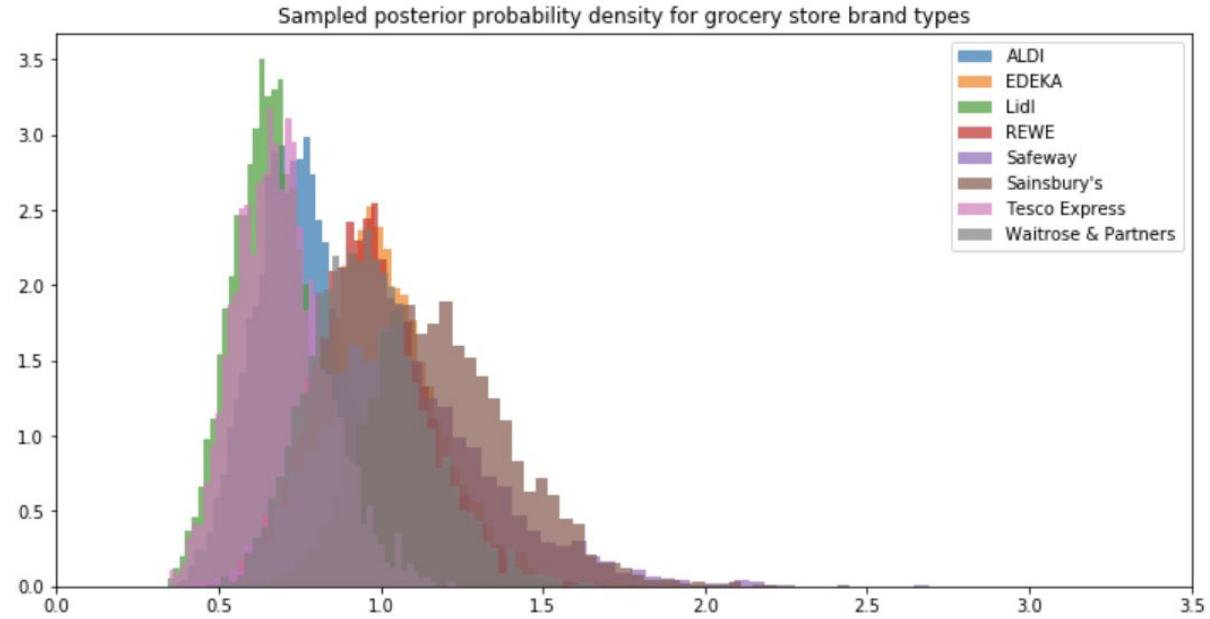
	Label	Mean	Posterior 95% confidence interval	Standard Error
0	Apple	3.20	[2.184677230599934, 4.475090552105302]	0.59
1	Banana	2.43	[1.5867651037486283, 3.512720541736519]	0.49
2	Tomato	4.85	[3.3863110138498795, 6.72759512119559]	0.83
3	Potato	1.61	[0.9869538830451338, 2.4262799342863275]	0.37
4	Flour	1.34	[0.7716325802453086, 2.120142188533375]	0.34
5	Rice	3.90	[2.7190334622054158, 5.474270200494524]	0.69
6	Milk	1.16	[0.6005664769534635, 1.8519317391258523]	0.32
7	Butter	7.81	[5.520328310408312, 10.824691292588394]	1.34
8	Eggs	0.36	[0.027748720351875335, 0.8676778261271055]	0.22
9	Chicken	9.77	[6.89949657260153, 13.58305143308861]	1.67

Figure 1. The basic average price for each of the 10 products.

Unsurprisingly, the most expensive good is chicken with an average base price of €9.77 per kilogram and is closely followed with butter. The cheapest goods are eggs, milk, flour, and potatoes with average base prices of €0.36, €1.16, €1.34 and €1.61 respectively.

[3] #dataviz

The effect of the grocery store brand on the basic price of the product



	Label	Mean	Posterior 95% confidence interval	Standard Error
0	ALDI	0.75	[0.49326196200268807, 1.0495734082721109]	0.14
1	EDEKA	1.00	[0.6690967190526561, 1.385731229478567]	0.18
2	Lidl	0.67	[0.44193203523537705, 0.937396654876085]	0.13
3	REWE	0.97	[0.6553600311355698, 1.3412762824630142]	0.17
4	Safeway	1.09	[0.6436477897825195, 1.7095291694933357]	0.27
5	Sainsbury's	1.20	[0.8170672907989022, 1.675971095990181]	0.22
6	Tesco Express	0.69	[0.45231377697452974, 0.9953867317819695]	0.14
7	Waitrose & Partners	0.96	[0.6466818263372608, 1.3498582227057316]	0.18

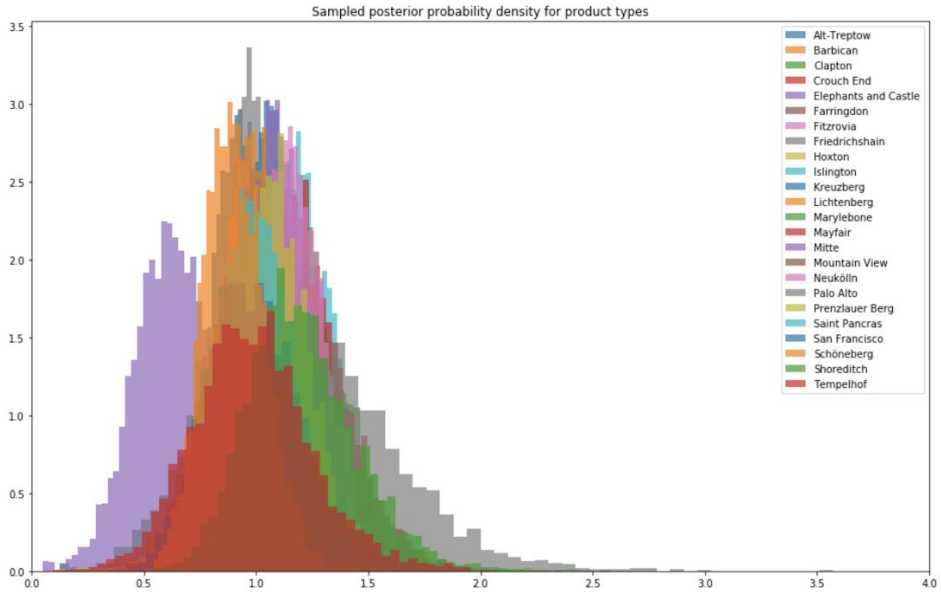
Figure 2. The effect of grocery store brand on the basic average price of goods

Safeway and Sainsbury's have the highest mean multipliers. Given their standard errors, we may attribute this to the fact that we had fewer data from London and California when compared to the Berlin data. We can also observe an overlap of the posterior distributions for ALDI, EDEKA, and REWE. To compare the average values in greater detail, I performed a t-test and got the following results[4]:

[4] #significance: calculated the t-test to check if there is a statistical difference in the values

- Goods in ALDI are significantly cheaper than those in EDEKA, REWE, Safeway, Sainsbury's and Waitrose & Partners; and are significantly pricier than those in Lidl and Tesco Express
- Goods in EDEKA are significantly pricier than those in ALDI, REWE, Lidl, Tesco Express and Waitrose & Partners; and are significantly cheaper than those in Safeway and Sainsbury's
- Goods in Lidl are significantly cheaper than those ALL other stores
- Goods in REWE are significantly pricier than those in Tesco Express and Waitrose & Partners; and are significantly cheaper in the other stores

The effect of the geographical location of the store on the basic price of the product



	Label	Mean	Posterior 95% confidence interval	Standard Error
0	Alt-Treptow	0.95	[0.6984567211774809, 1.2565586954621653]	0.14
1	Barbican	1.07	[0.7857391221574752, 1.4162278929233307]	0.16
2	Clapton	0.96	[0.46451674285478245, 1.505301158028967]	0.26
3	Crouch End	1.13	[0.8256796500539838, 1.5242182519621674]	0.18
4	Elephants and Castle	0.64	[0.29147809463731655, 1.0142564081459535]	0.19
5	Farringdon	1.20	[0.8521117933066215, 1.6570424172129776]	0.20
6	Fitzrovia	0.99	[0.7051924862153555, 1.3337931057933143]	0.16
7	Friedrichshain	1.00	[0.7542846228384851, 1.2753651656541103]	0.13
8	Hoxton	1.08	[0.5814611118711953, 1.6701112494115053]	0.27
9	Islington	1.21	[0.9402035606856426, 1.545063487270615]	0.15
10	Kreuzberg	1.08	[0.8210412341249445, 1.3760723463654385]	0.14
11	Lichtenberg	1.01	[0.7442361265591221, 1.319771113476317]	0.15
12	Marylebone	1.21	[0.8833328858689758, 1.6152007041748944]	0.19
13	Mayfair	1.23	[0.9019767838726523, 1.6487018824308866]	0.19
14	Mitte	1.11	[0.8510393841409198, 1.4051551249708303]	0.14
15	Mountain View	1.00	[0.598390482806867, 1.526657020092177]	0.24
16	Neukölln	1.17	[0.8943384918454722, 1.497957090440857]	0.15
17	Palo Alto	1.39	[0.8591219036865809, 2.122546925060892]	0.32
18	Prenzlauer Berg	1.09	[0.805524818040581, 1.439062042904024]	0.16
19	Saint Pancras	1.01	[0.7049663794771951, 1.3735874550114557]	0.17
20	San Francisco	1.01	[0.6162248137825199, 1.5186351885967557]	0.23
21	Schöneberg	0.90	[0.6504908714924503, 1.1964335148805443]	0.14
22	Shoreditch	1.21	[0.7874856776318828, 1.7411440674842562]	0.24
23	Tempelhof	0.99	[0.5000171583333516, 1.5532785357173136]	0.26

Figure 3. The effect of geographical location on the basic average price of goods

Cost of Basic Goods LBA

A t-test yields the following results:

- Goods in Elephants & Castle, London are significantly cheaper than those in ALL other geographical locations
- Goods in Alt-Treptow, Berlin are significantly cheaper than those in ALL other geographical locations
- Goods in Mayfair, London are significantly pricier than those in ALL other geographical locations
- Goods in Palo Alto, CA are significantly pricier than those in ALL other geographical locations

How strong is each of the brand and location effects? Which of these has the greatest influence on price variation between shops?

Grocery Store Brand Multiplier:

- The greatest grocery store brand multiplier effect: 1.2
- The least grocery store brand multiplier effect: 0.67
- The difference in sample means is 0.53.

Geographical Location Multiplier

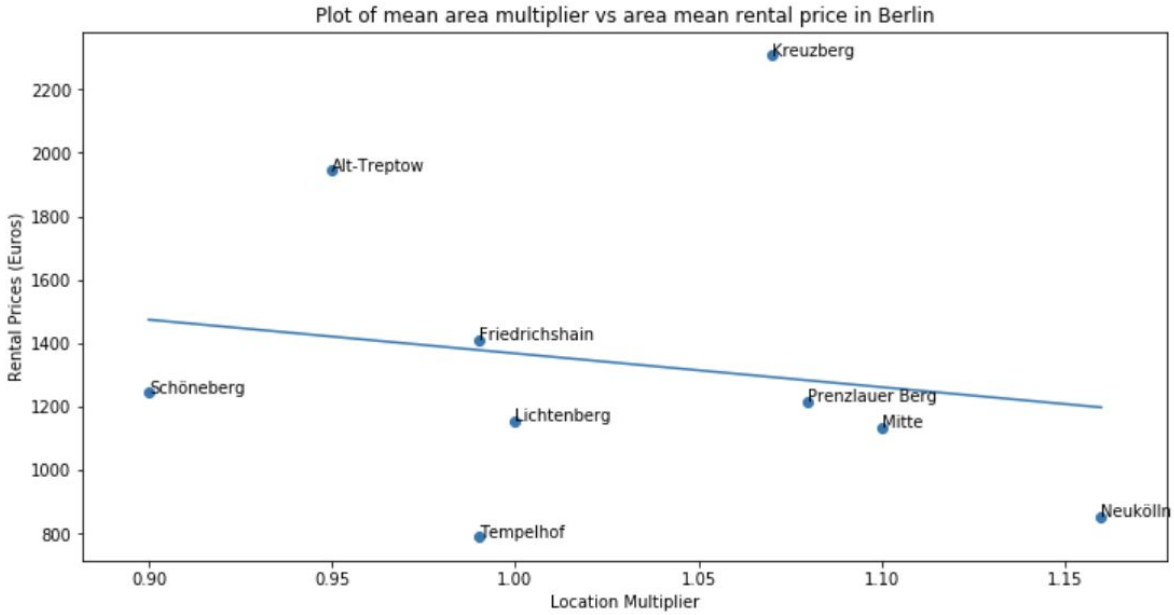
- The greatest geographical location multiplier effect: 1.39
- The least geographical location multiplier effect: 0.64
- The difference in sample means is 0.75.

Therefore, the Geographical Location multiplier has a greater influence than the grocery store multiplier

Does price variation by geographical location correlate with variation in rental prices in Berlin/London, or not?

```
berlin_neighborhoods = ["Alt-Treptow", "Friedrichshain", "Kreuzberg",
"Lichtenberg", "Mitte",
                        "Neukölln", "Prenzlauer Berg", "Schöneberg",
"Tempelhof"]

berlin_neighborhoods_price_averages = []

for i in berlin_neighborhoods:
  s = region[region['Label']== i]
  berlin_neighborhoods_price_averages.append(float(s["Mean"]))

# Got the berlin rental price data by searching listed rentals on
https://www.immobilienscout24.de/ by the neighborhoods where the shops were
located.
# Recorded the rental prices for the first 10 housing units listed in each
neighborhood

Alt_Treptow = [1470, 2741, 1600, 2099, 895, 1435, 2679, 3192, 2147, 1184]
Schöneberg = [1057, 3119, 434, 340, 2970, 2000, 585, 1100, 600, 1900]
Prenzlauer_Berg = [1749, 1459, 2849, 1197, 2490, 4680, 2498, 1998, 2890,
1258]
Neukölln = [668, 1710, 1440, 449, 1563, 1495, 832, 1215, 1162, 1000]
Mitte = [380, 1360, 1568, 1800, 1568, 1596, 1200, 680, 630, 550]
Lichtenberg = [1983, 779, 729, 1042, 632, 750, 579, 660, 585, 780]
Kreuzberg = [1249, 968, 1113, 1084, 1587, 1359, 1150, 507, 1575, 1555]
Friedrichshain = [712, 498, 1300, 1470, 1300, 1697, 1150, 1200, 1636, 1475]
Tempelhof = [950, 746, 760, 435, 746, 1097, 760, 655, 685, 1064]

x = berlin_neighborhoods_price_averages
y = []

berlin_property_prices = [Alt_Treptow, Schöneberg, Prenzlauer_Berg,
Neukölln, Mitte, Lichtenberg, Kreuzberg, Friedrichshain, Tempelhof]

for i in berlin_property_prices:
  y.append(np.mean(i))

plt.figure(figsize=(12, 6))
```

```
plt.scatter(x, y)
for el, text in enumerate(berlin_neighborhoods):
    plt.annotate(text, (x[el], y[el]))
plt.plot(np.unique(x), np.poly1d(np.polyfit(x, y, 1))(np.unique(x)))
plt.title('Plot of mean area multiplier vs area mean rental price in
Berlin')
plt.xlabel("Location Multiplier")
plt.ylabel("Rental Prices (Euros)")
plt.show()
print("Pearson's Correlation Coeffiecient", stats.pearsonr(x, y)[0])
```



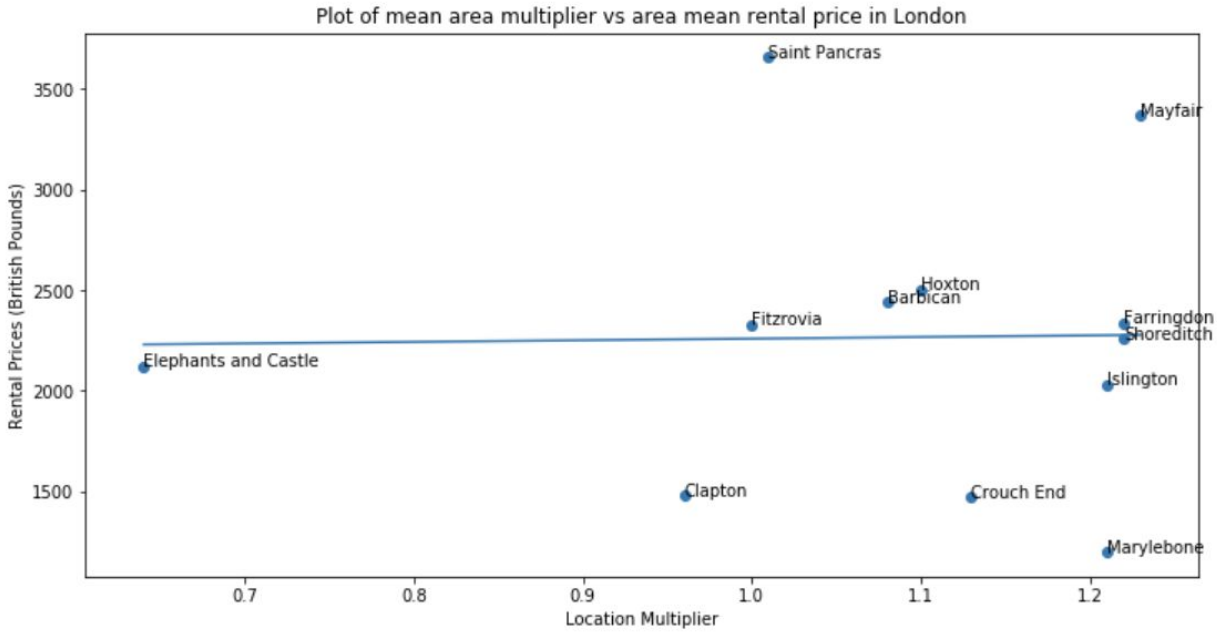

```
Pearson's Correlation Coeffiecient -0.17582487826895424
```

Figure 4(a). There is a weak negative correlation between the mean rental prices and mean geographical area multipliers in Berlin

```
london_neighborhoods = ["Barbican", "Clapton", "Crouch End", "Elephants and
Castle",
                        "Farringdon", "Fitzrovia","Hoxton", "Islington",
                        "Marylebone", "Mayfair", "Saint Pancras",
"Shoreditch"]

london_neighborhoods_price_averages = []
```

```
for i in london_neighborhoods:
  s = region[region['Label']== i]
  london_neighborhoods_price_averages.append(float(s["Mean"]))

# Got the London rental price data by searching listed rentals on
https://www.zoopla.co.uk/ by the
# neighborhoods where the shops were located. Recorded the rental prices
for the first 10
# housing units listed in each neighborhood


Barbican = [2427, 2600, 4312, 2250, 2600, 1820, 1000, 3012, 2350, 2058]
Clapton = [1101, 1400, 1300, 1600, 1750, 1300, 1400, 1452, 1700, 1800]
Crouch_End = [1647, 1950, 1400, 1600, 1250, 1625, 1560, 1625, 737, 1300]
Elephants_and_Castle = [1820, 1993, 3033, 800, 2730, 2730, 2167, 2383,
2080, 1473]
Farringdon = [2990, 1993, 800, 2383, 1800, 2275, 3250, 2275, 2600, 3012]
Fitzrovia = [3878, 1850, 3358, 2362, 2275, 1625, 1668, 2448, 2275, 1560]
Hoxton = [2648, 2253, 3100, 3012, 2492, 2275, 2492, 3100, 2600, 1049]
Islington = [2250, 2253, 2648, 2145, 2990, 1993, 600, 2012, 1300, 2100]
Marylebone = [1175, 1200, 1050, 1200, 1200, 1375, 1200, 1400, 1050, 1150]
Mayfair = [4160, 2167, 3250, 3207, 3250, 4225, 2947, 4117, 2492, 3878]
Saint_Pancras = [3250, 1543, 3592, 2999, 4400, 3680, 6338, 7220, 1170,
2383]
Shoreditch = [2253, 2648, 1993, 2102, 2100, 3359, 1185, 1200, 2920, 2817]

x_1 = london_neighborhoods_price_averages
y_1 = []

london_property_prices = [Barbican, Clapton, Crouch_End,
Elephants_and_Castle, Farringdon,
                          Fitzrovia, Hoxton, Islington, Marylebone,
Mayfair, Saint_Pancras, Shoreditch]

for i in london_property_prices:
  y_1.append(np.mean(i))

plt.figure(figsize=(12, 6))
plt.scatter(x_1, y_1)
for el, text in enumerate(london_neighborhoods):
    plt.annotate(text, (x_1[el], y_1[el]))
```

```
plt.plot(np.unique(x_1), np.poly1d(np.polyfit(x_1, y_1,
1))(np.unique(x_1)))
plt.title('Plot of mean area multiplier vs area mean rental price in
London')
plt.xlabel("Location Multiplier")
plt.ylabel("Rental Prices (British Pounds)")
plt.show()
print("Pearson's Correlation Coeffiecient", stats.pearsonr(x_1, y_1)[0])
```




```
Pearson's Correlation Coeffiecient 0.018923157328351318
```

Figure 4(b). There is a weak positive correlation between the mean rental prices and mean geographical area multipliers in London

In Berlin, there is a weak negative correlation between the mean rental prices and mean geographical area multipliers whereas, in London, we see a weak positive correlation between the mean rental prices and mean geographical area multipliers. These results are not intuitive as one would expect more expensive neighborhoods (as deduced from the rental prices) to be correlated with a higher cost of basic goods (where we use the geographical location as an indicator). This suggests the presence of confounding variables or more plausibly, reflects the fact that my method of collecting sample data for rental prices was not sophisticated enough as I simply conveniently sampled the first 10 rental listings from websites.

Metadata:

Store: Lidl - Prenzlauer Allee 44, 10405 Berlin, Germany
Visited: November 1, 12:10PM

Store: Lidl - Friedenstraße 94A, 10249 Berlin, Germany
Visited: November 1, 12:55PM

Appendix:

Raw Data:
https://docs.google.com/spreadsheets/d/e/2PACX-1vQoe0jJnNeKWc6koLjy32A32fN-7m6Ae9n0UJg7_QqcXuo7gRvEXb7QZiumtgkpPTE1czLnsVKRtiBf/pub?gid=0&single=true&output=csv

Code:
https://github.com/kagenidennis/CS146/blob/master/CS146_LBA_workbook_ipynb_Final.ipynb