CONTENTS

Empirical Demand Analysis of Airbnb Homestay Market Based on Random Coefficient Model

Fangshu Gao

ABSTRACT

This paper attempts to apply random coefficient demand model in analyzing Airbnb homestay market, to overcome consumer heterogeneity and product price endogeneity problem that traditional demand systems model faced. Based on Seattle Airbnb Open Data and web crawling data, we estimate the impact of different characteristics on homestay's market share and consumers' utility, in differentiated products market. The empirical results show that the random coefficient demand model have great internal validity in analyzing Airbnb homestay market: 1. homestay price has a significant negative impact on consumers' average marginal utility, the consumers having lower target prices are more sensitive to homestay price; 2. consumers have unobservable heterogeneous reactions to host's response rate, superhost certification, minimum nights of booking; 3. Consumers' preferences on reviews scores rating, accommodates per room, instant booking are more consistent and have no significant difference. The analysis framework and conclusions above on the one hand can help researchers to analyze the Airbnb market, on the other hand can guide Airbnb hosts aiming to introduce differentiated homestays to reduce competition to improve revenue.

Keywords: Airbnb, homestay market, random coefficient demand model

基于随机系数模型的Airbnb短租市场需求实证分析

高方澍

摘要

本文试图将随机系数需求模型应用于 Airbnb 短租市场，克服了传统需求系统模型所面临的消费者异质性和产品价格内生性问题。基于 Airbnb 在西雅图市的公开交易数据，本文估计了在产异化产品市场中，Airbnb 公司房源的各类特征对其市场份额和消费者效用的影响。实证结果发现，随机系数需求模型对 Airbnb 短租市场的分析有很强的内部有效性：1. 房源价格对消费者平均边际效用有显著的负面影响，且心理预期价格越低的消费者对房源价格越敏感；2. 消费者对房东回复率、房东认证、最低出租天数等特征有尚未观测到的差异化反应；3. 消费者对以往评价、单个房间人数、快速预订的偏好较为一致，不存在显著差异。以上分析框架及结论一方面能帮助研究者更好地分析 Airbnb 市场，另一方面能指导 Airbnb 房东针对性地推出差异化房源，以降低竞争提高收益。

关键词：Airbnb　短租市场　随机系数需求模型

I. Introduction

Airbnb is the largest hosting platform in homestay and short term rental market. And this kind of creative and successful homestay become a new form of visitor lodging throughout the world. Most studies about Airbnb are correlated to sharing economy (Ert et al., 2016, Zervas et al., 2016) and case studies (Edelman and Luca, 2014, Kaplan et al., 2015), but few of them concentrates on demand estimation.

Demand estimation is a critical topic in marketing analysis. However, in most cases, marketers and researchers only have aggregate dataset, which may cause troubles. Marketers need to get some insight into different product preference of different consumers, and then to relevantly produce differentiate products. Researchers need to solve endogenous problems, because there are some unobservable (unobservable for researchers but observed by marketers) characteristics that may influence product pricing.

Berry et al. (1995) apply random coefficient model (BLP model) to solve these problems in automobile market, but it lacks individual consumer data where Berry et al. have to draw samples of demographic data as substitute. In this paper, we use random coefficient model with micro data (both for products and consumers) to estimate Airbnb homestay demand market. Using micro data makes analysis of Airbnb demand market possible and improves the accuracy of estimation. Our empirical estimation mainly shows that: (1) price of homestays has a significant negative impact on marginal utility of consumers, and consumers with lower target price are more sensitive to the price; (2) host response rate, host certification like “superhost” and minimum number of nights when booking can significant influence consumers’ marginal utility and this impact have heterogeneity among consumers, which need to be discovered; (3) accommodates per room, reviews scores rating, instant bookable can influence all kinds of consumers’ utility without heterogeneity.

II. Literature Review

Demand systems are often used in empirical industrial organization studies. But the demand system based on “product” space, which defines utility on product itself, will encounter heterogeneous agents problem, simulation problem, new goods problem and too many parameters (Ackerberg and Benkard, 2007). Thus, models based on “characteristics” space, which can dates back to Lancaster (1966, 1971) and McFadden (1972, 1980), are formed to solve these problems.

However, the original demand system model based on characteristic space cannot capture unobservable characteristics of product, which causes heterogeneity and endogeneity problem. And Berry (1994) proposes a solution by adding observable differentiate characteristics and a variable for unobservable. Then, "BLP" model (Berry et al., 1995) appears, where random coefficient model is used in market analysis.

As for the method of estimation, Berry (1994) proofs that the predicted market share equals to observed market share at a unique value of unobservable variable, which is the fundament of estimation. Berry et al. (1995) propose the concentration mapping method in estimation, which can calculate market share by iterations, and they add supply equation to the demand model. Later, nonparametric model is built by Berry and Haile (2014), Berry and Haile (2009).

Moreover, source of data and topics where the model applied vary among different studies. Berry et al. (2004) use micro data to capture second choice of consumers, and improve the accuracy of estimation. Nevo (2000a) uses price of the same product in other market as a new kind of instruments. Random coefficient model is applied in many field such as school choice (Hastings et al. 2009, Neilson, 2013), news media (Gentzkow and Shapiro, 2010), health plan and insurance (Bundorf and Mahoney, 2008, Lustig, 2008), PC market (Eizenberg, 2014), etc. And we apply random coefficient demand model to homestay market for the first time.

III. The Model and Steps in Estimation

In model section, we mainly consider two issues. The first is establishing a proper model, and the second is estimating the parameters in the model.

1. Random-Coefficient Demand Model

We now show a basic model to illustrate the frame of random-coefficient demand model and then in chapter 3.1 we expand the model to the one used in real estimation.

Based on the BLP (Berry et al., 1995) model, for consumer i, his indirect utility can be written as a function:

$$u_{ij} = x_j\beta - \alpha p_j + \xi_j + \varepsilon_{ij} \tag{1}$$

where x_j is observable characteristics of product j, p_j is price of product j, ξ_j is unobservable characteristics of product j, and ε_{ij} is consumer characteristics which is independent identically distributed across both i and j. For products outside the market we observed, their utility is $u_{i0} = \varepsilon_{i0}$. Thus, the mean utility across consumers of product j is:

$$\delta_j = \beta x_j - \alpha p_j + \xi_j \tag{2}$$

The consumer will buy product j if u_{ij} is the highest utility among utilities of other products in product set J:

$$u_{ij} \geq u_{ir} \quad \forall r \in J \tag{3}$$

And it can be written as a set:

$$A_j = \left\{ (\varepsilon_{i1}, \varepsilon_{i2}, \ldots, \varepsilon_{iJ}) \mid u_{ij} \geq u_{ir}, \forall r \in J \right\} \tag{4}$$

Combining equation (1) and i.i.d. assumption of ε_{ij}, we have the market share that:

$$\sigma_j = \int_{A_j} dP(\varepsilon) \tag{5}$$

Noted by Mcfadden (1972), when ε obeys extreme distribution, σ_j can be expressed analytically:

$$\sigma_j = \frac{e^{\delta_j}}{1 + \sum_k e^{\delta_k}} \tag{6}$$

Now, the problem is how to estimate the coefficients in the model, which we will discuss in following contents.

2. Steps in Estimation

In above model, our goal is estimating the parameters like α, β in the model. Before introducing steps in estimation, we expand the basic model to a model considering consumer heterogeneity, by adding more interaction terms to equation (1):

$$u_{ij} = \delta_j + \sum_k \sum_r \theta^o_{kr} x_{kj} d_{rj} + \sum_k \theta^u_k x_{kj} v_i + \varepsilon_{ij} \tag{7}$$

where δ_j is the utility from product j:

$$\delta_j = \sum_k \beta_k x_{kj} - \alpha p_j + \xi_j \tag{8}$$

In equation (7) and equation (8), product j has k observable characteristics, price p_j and unobservable characteristics ξ_j. For interaction terms, $\sum_k \sum_r \theta^o_{kr} x_{kj} d_{rj}$ are interactions between observable consumer characteristics and product characteristics, which consider that different consumer characteristics may have an impact on the utility from product characteristics. $\sum_k \theta^u_k x_{kj} v_i$ are interactions between unobservable consumer characteristics and product characteristics, which consider that unobservable consumer characteristics may influence the utility from product characteristics. Therefore, we need to estimate a parameter set of $\{\theta^o, \theta^u, \alpha, \beta\}$, and there are three steps to finish estimation.

Step 1. For this model, market share of product j doesn't have closed-form function like equation (6), but we can compute it in a simulation way:

$$\sigma_j(\theta^o, \theta^u, \delta, D^I) = \frac{1}{I} \sum_{i=1}^{I} \frac{e^{\delta_j + \sum_k \sum_r \theta^o_{kr} x_{kj} d_{rj} + \sum_k \theta^u_k x_{kj} v_i}}{1 + \sum_{q \in J} e^{\delta_q + \sum_k \sum_r \theta^o_{kr} x_{kq} d_{rq} + \sum_k \theta^u_k x_{kq} v_i}} \tag{9}$$

where we take I random draws of v_i, and D^I is the distribution of v_i. The variance of simulation error can be reduced by increasing I (Reynaert and Verboven, 2014). Thus in order to reduce bias in estimation, we need to set I as large as possible.

Step 2. For each product j, we have its observed real market share s_j, then we combine our estimated market share σ_j and observed market share s_j by concentration mapping (Berry et al., 1995):

$$\delta^n_j = \delta^{n-1}_j + \ln(s_j) - \ln(\sigma_j(\theta^o, \theta^u, \delta^{n-1}, D^I)) \tag{10}$$

After finishing calculation of formula (9), plug σ_j into function (10) to compute δ^n_j. Noted that the initial value of δ^0_j can be an arbitrary number, because Berry et al. (1995) prove that function (10) has a unique fixed point δ_j that makes $s_j = \sigma_j(\theta^o, \theta^u, \delta^{n-1}, D^I)$. Thus, wherever the iteration start, δ^n_j will converge to a fixed value. But considering efficiency of computation, we need choose a proper initial δ^0_j to reduce iteration times.

Next, we can express the unobservable characteristics of product j, ξ_j, as:

$$\xi_j = \delta_j - (\sum_k \beta_k x_{kj} - \alpha p_j) \tag{11}$$

And ξ_j here can be regarded as residual term in regression equation (8). Thus, we can estimate $\{\alpha,\beta\}$,then calculate ξ_j by equation (11).

Step 3. For each ξ_j in step II, we apply GMM method to estimate parameter set $\{\theta^o,\theta^u\}$. We need an instruments vector Z which satisfies $E(\xi_j \mid Z) = 0$. Based on assumption of instruments, our goal is to find the parameters that can minimize the objective function value:

$$\{\hat{\theta}^o,\hat{\theta}^u\} = \arg\min_{\theta^o,\theta^u} \xi(\theta^o,\theta^u)^{\mathrm{T}} Z \Omega^{-1} Z^{\mathrm{T}} \xi(\theta^o,\theta^u) \tag{12}$$

where Ω is a weighting matrix, $\Omega = E(Z^{\mathrm{T}}\xi\xi^{\mathrm{T}}Z)$. And the initialization of Ω can refer to appendix of Nevo (2000b).

Please note that these three steps will repeat many times to estimate parameters. We firstly use $\sigma_j(\theta^o,\theta^u,\delta^0,D^l)$ to calculate δ_j^1, next use δ_j^1 to calculate a new σ_j, say $\sigma_j(\theta^o,\theta^u,\delta^1,D^l)$, and so on until reach the fixed point at δ_j^n. Then equation (11) implies the residual ξ_j and from equation (12) we get $\{\hat{\theta}^o,\hat{\theta}^u\}$ based on ξ_j. Furthermore, we turn back to step 1 to calculate new $\sigma_j(\hat{\theta}^o,\hat{\theta}^u,\delta^0,D^l)$ and start next loop. It means that we estimate in order of "step 1 – step 2 – step 3 – step 1 – step 2 – step 3 …".

As for the significance of estimation, we calculate the variance covariance after the convergence having been reached. Refer to Vincent (2015), the estimated asymptotic variance is:

$$\hat{V}(\hat{\theta}^o,\hat{\theta}^u,\hat{\xi}) = \{G(\hat{\theta}^o,\hat{\theta}^u)^{\mathrm{T}}\Omega(\hat{\xi})G(\hat{\theta}^o,\hat{\theta}^u)\}^{-1} \tag{13}$$

where

$$G(\hat{\theta}^o,\hat{\theta}^u) = Z^{\mathrm{T}}\{X_k, D_\theta(\hat{\theta}^o,\hat{\theta}^u)\} \tag{14}$$

The $G(\hat{\theta}^o,\hat{\theta}^u)$ is a $l \times (k + k \times 2)$ matrix, l is number of instruments. And:

$$D_\theta(\hat{\theta}^o,\hat{\theta}^u)=-\begin{pmatrix} \frac{\partial s_1}{\partial \delta_1} & \cdots & \frac{\partial s_1}{\partial \delta_J} \\ \vdots & \vdots & \vdots \\ \frac{\partial s_J}{\partial \delta_1} & \cdots & \frac{\partial s_J}{\partial \delta_J} \end{pmatrix}^{-1} \begin{pmatrix} \frac{\partial s_1}{\partial \hat{\theta}_1^u} & \cdots & \frac{\partial s_1}{\partial \hat{\theta}_K^u}, & \frac{\partial s_1}{\partial \hat{\theta}_1^o} & \cdots & \frac{\partial s_1}{\partial \hat{\theta}_K^o} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial s_J}{\partial \hat{\theta}_1^u} & \cdots & \frac{\partial s_J}{\partial \hat{\theta}_K^u}, & \frac{\partial s_J}{\partial \hat{\theta}_1^o} & \cdots & \frac{\partial sJ}{\partial \hat{\theta}_K^o} \end{pmatrix} \tag{15}$$

where

$$\begin{cases} \frac{\partial s_j}{\partial \delta_j} = \frac{1}{I}\sum_{i=1}^{I}\sigma_{ij}(1-\sigma_{ij}) \\ \frac{\partial s_j}{\partial \delta_m} = \frac{1}{I}\sum_{i=1}^{I}\sigma_{ij}\sigma_{im} \\ \frac{\partial s_j}{\partial \hat{\theta}_k^u} = \frac{1}{I}\sum_{i=1}^{I}\sigma_{ij}v_{ik}(x_{jk}-\sum_{m=1}^{J}x_{mk}\sigma_{im}) \\ \frac{\partial s_j}{\partial \hat{\theta}_k^o} = \frac{1}{I}\sum_{i=1}^{I}\sigma_{ij}d_i(x_{jk}-\sum_{m=1}^{J}x_{mk}\sigma_{im}) \end{cases} \tag{16}$$

Therefore, the variance of parameters we estimate is $diag(\hat{V}(\hat{\theta}^o,\hat{\theta}^u,\hat{\xi}))$.

IV. Data Processing

We use Seattle Airbnb Open Data on Kaggle Datasets platform[1] and crawl data of Airbnb consumers by Python web crawling program. We first introduce Seattle Airbnb Open Data and next illustrate details of crawling data.

Seattle Airbnb Open Data ("Airbnb data") is a dataset which including three parts: Calendar, Listings and Reviews. In Calendar, there are plans of available date decided by host, but we don't use this part of data in this paper. In Listings, there are many variables of homestays, such as price, number of accommodates, number of rooms, review scores rating, reviews per month, etc. In Reviews, there are all reviews of homestays (before 2016) listed in Listings and the reviewer id in Airbnb.

We take 1300 homestays in Seattle from Airbnb data, which can be regarded as 1300 differentiated products in Seattle Airbnb market. We choose price and observed characteristics of these 1300 kinds of products as p and x_k.

[1] See https://www.kaggle.com/airbnb/seattle, more Airbnb data of other cities can be found on http://insideairbnb.com

1. Market Share

As for observed market shall of each homestay, the ideal variable is "number of days a listing is booked out of the year/sum of days each listing is booked out of the year", because there is a certain number of days that Airbnb guests stay in Seattle and the homestays share these days as their market share. But we cannot get the data that which days the homestay is booked. Therefore, we need a substitute of that ideal market share variable. A policy analysis report by Budget and Legislative Analyst's Office (2015) of San Francisco introduce an utilization rate model:

$$D = \frac{(R_l/R_a)\times N_a}{A}\times 365 \tag{17}$$

where D is "number of days a listing is booked out of the year", R_l is "number of reviews for a listing", R_a is "review rate", N_a is "average number of nights for a listing", A is "number of days listing active". R_a is estimated as 72% or 30.5% by Budget and Legislative Analyst's Office (2015). However, we can assume R_a, N_a, A are constants, then D is only determined by R_l. Moreover, R_l is proportional to a variable, "number of reviews per month", in Airbnb Listings data. Therefore, we can indicate market share of product j by its number of reviews per month divided by all reviews in market.

2. Product Characteristics

Observable product characteristics are from Airbnb Listings data, we employ variables "number of available days (*Available*)", "host response rate (*Response*)", "host is superhost (*Superhost*)", "accommodates per rooms (*Accommodates*)", "minimum nights (*Nights*)", "reviews scores rating (*Rating*)", "instant bookable (*Instant*)", "cancellation policy (*Cancellation*)" (Table 1) as x_k.

TABLE 1. PRODUCT CHARACTERISTICS VARIABLES

Variable	Definition
Number of available days (*Available*)	Number of days that the homestay is available in a year, rescaled by dividing 365
Host response rate (*Response*)	Host response rate to guest when before the deal was completed, rescaled by dividing 100
Host is superhost (*Superhost*)	1 if the host is superhost (marked with a medal at listing page); 0 otherwise

Accommodates per rooms (*Accommodates*)	Number of guests that the homestay can accommodate divided by number of rooms in the homestay
Minimum nights (*Nights*)	Minimum number of nights that the guest can book, rescaled by dividing 365
Reviews scores rating (*Rating*)	Reviews scores from guests, rescaled by dividing 10
Instant bookable (*Instant*)	1 if the homestay can be booked without confirmation of host; 0 otherwise
Cancellation policy (*Cancellation*)	1 if the homestay has strict cancellation policy (50% refund up until 1 week prior to arrival); 2 for moderate policy (full refund 5 days prior to arrival); 3 for flexible policy (full refund 1 day prior to arrival)

3. Consumer Characteristics

Consumer characteristics include observable consumer characteristics and unobservable consumer characteristics. We use mean price per person of wish lists saved by guests (*Wish* or w) as an observable consumer characteristic d_j. Each homestay of 1300 ones we observed has many reviewers, who lived in the homestay for several days. The reviewers may save wish lists which indicate their homestay preference. Meanwhile, these reviewers' personal Airbnb pages can be reached from pages of homestay we observed. Thus, we can collect the reviewers' wish lists data by Python web crawling program (see Appendix).

Previous studies use draws from distribution of demographic data, i.e. Current Population Survey (Berry et al., 1995, Nevo, 2000b) as observable consumer characteristics. However, we can crawl and calculate the demographic data, mean price per person of wish lists saved by guests, corresponding to each homestay we observed. Our micro data of consumers solves the problem of demographic dataset and makes estimation more accurate.

As for unobservable characteristics, we use draws form type I extreme distribution.

4. Instruments

In Step 2 mentioned above, market share and price of homestay may be endogenous, thus we need a set of instruments $Z = \{z_1, z_2, \ldots, z_m\}$ to solve this problem. z_m is a $j \times 1$ vector and m is not less than number of endogenous variables. A common solution is using cost shifters or proxies for cost shifters as instruments of price (Berry et al., 1995, Berry et al., 1999, Nevo, 2000a). But in our model, it is impossible to find alternative homestay in the same market or the same products in other market.

To find an instrument that satisfies $E(\xi_j \mid Z) = 0$, we consider the situations from the angle of consumers: When a consumer chooses a homestay, his choice partly depends on the price of the homestay. Fundamentally, he compares the price of the homestay with his target price in mind, which can be reflected by the prices in his wish lists. Therefore, the average per-person price in consumers wish lists are correlated to price of the homestay, but the average per-person price is not correlated to other characteristics of the homestay, because prices in wish lists do not belong to this homestay.

Besides, we also set all x_k s as instruments to avoid that cost and markups may influence the endogenous variable only through price (Berry et al., 2016). Thus, our instrument contains observable product characteristics and observable consumer characteristics: $Z = \{x_k, w\}$, where Z is a $j \times (k+1)$ matrix.

The descriptive statistics, which contain x_k and w, are shown in Table 2 as follows:

TABLE 2. VARIABLES STATISTICS DESCRIPTION

Variable	Mean	Median	Standard Error	Max	Min
Price	0.133	0.109	0.085	0.910	0.025
Available	0.735	0.907	0.308	1.000	0.005
Response	0.954	1.000	0.112	1.000	0.250
Superhost	0.325	0.000	0.469	1.000	0.000
Accommodates	0.253	0.200	0.096	0.900	0.100
Nights	0.209	0.200	0.190	3.100	0.100
Rating	0.944	0.960	0.050	1.000	0.530
Instant	0.164	0.000	0.370	1.000	0.000
Cancellation	1.728	2.000	0.734	3.000	1.000
Wish	0.326	0.252	0.610	17.913	0.012

V. Empirical Results

To estimate parameters in the full model we mentioned above, we write an estimation program with R (see Appendix). In the R program, to keep a balance between computation efficiency and estimation accuracy, we set the sampling number $I=1000$. Referring to the advice by Dubé et al. (2012), we let the tolerance of iteration in Step 2 be 1×10^{-15} to avoid nonconvergence in Step 3, which caused by a too loose tolerance in Step2.

The results of estimations are shown in Table 3. The first column presents the estimation of parameters (β) before price (p) and observable characteristics of products (x_k), which indicates mean of consumers' marginal utility from price and observable product characteristics. The next two columns contain the estimation of heterogeneity around β. The column named "Standard Deviations" presents the estimation of parameters (σ) before interactions ($\sum_k \theta_k^u x_{kj} v_i$) between unobservable consumer characteristics and unobservable product characteristics, which captures the impact of unobservable consumer characteristics. And the last column presents the estimation of parameters before interactions ($\sum_k \sum_r \theta_{kr}^o x_{kj} d_{rj}$) between observable consumer characteristics and observable product characteristics, which indicates the demographic effect on mean of consumers' marginal utility.

TABLE 3. EMPIRICAL RESULTS

Variable	Means β	Standard Deviations σ	Interactions with Demographic Variables *Wish*
Price	-2.821*** (0.007)	0.175*** (0.028)	0.017*** (0.005)
Available	2.615*** (0.020)	0.839*** (0.153)	1.984*** (0.378)
Response	6.661*** (0.026)	0.992*** (0.310)	1.526 (2.262)
Superhost	0.482*** (0.010)	0.428*** (0.028)	0.239 (1.774)
Accommodates	-2.071*** (0.007)	0.615 (1.051)	0.384 (1.013)
Nights	-2.207*** (0.006)	2.419*** (0.750)	0.075 (0.875)
Rating	0.316*** (0.004)	0.208 (0.168)	0.831 (0.982)
Instant	0.956*** (0.046)	0.103 (0.369)	0.417 (7.759)
Cancellation	8.254* (4.867)	0.564*** (0.059)	2.112** (0.894)

Notes: 1. Based on 1300 observations, $R^2 = 0.637$
2. Robust standard error in brackets.
*** Significant at the 1% level.
** Significant at the 5% level.
* Significant at the 10% level.

The mean coefficient of homestay price (*price*) is significantly negative, it means that lower price of homestay can increase consumers' marginal utility, which is consistent with intuition. The interaction between homestay price (*price*) and mean price per person in wish lists (*wish*) is positive and significant at 1% level, which indicates that consumers with higher target price are less price-sensitive. The significance of standard deviation (σ) of mean price coefficient shows that other unobservable consumer characteristics also have an impact on consumers' price sensitivity.

The coefficient of availability of homestay (*Available*) shows that high availability of homestay improves the marginal utility of consumers. But we don't think that high availability has a direct positive impact on the marginal utility. Because homestay with high availability have more chance to be booked by consumer, rather than attract consumers directly. As an extreme case, a homestay will not be booked (marker share equal to 0) if it is not available in the calendar all year around, even it may be very "attractive". However, the estimates of interaction in last column is interesting. It is positive and significant at 1% level, indicating that consumers having higher target price are more sensitive to availability of homestay. Because consumers with higher target price may not care much about price of homestay, they are more possible to book the homestay they like no matter how expensive it is.

It should be noted that the coefficients of host response rate (*Response*), the host is superhost (*Superhost*), minimum night of booking (*Nights*), policy of cancellation (*Cancellation*) is significant and consistent with intuition, but their coefficients in last column (*Wish*) are not as significant as standard deviations (σ). This implies that the demographic variable (*Wish*) we use does not capture the effect from consumers' heterogeneity. There are some other demographic factors influence consumers' response to these four product characteristics (*Response, Superhost, Nights, Cancellation*), but we do not capture them in the model.

As for accommodates per rooms (*Accommodates*), reviews scores rating (*Rating*), instant bookable (*Instant*), the means coefficients are all significant but the coefficients of interactions are all not significant. It implies that accommodates per rooms, reviews scores rating, instant bookable influence consumers' marginal utility towards specific homestay, and the influence has little heterogeneity among consumers.

VI. Conclusions and Extensions

This paper has estimated the coefficients in random coefficient demand model and has analyzed the Airbnb homestay market in Seattle. All empirical results we discussed above show that the random coefficient demand model has strong explanatory power in Airbnb homestay market we observed. An instant-bookable homestay hosted by a superhost, with lower price, higher availability, higher response rate, less accommodates per room, less minimum nights of booking, higher reviews scores rating, will more likely to occupy more market share. And as a host, besides all factors noted above, he should concentrate on consumers' reaction towards response rate, superhost certification, minimum nights of booking, because different consumers have different sensitivities to these factors. Based on this, the host can make his homestay a differentiate product to reduce competition and increase margins.

However, there are more growing literatures about random coefficient demand model, which apply new instruments and new estimate method, even it can be combined with machine learning (Bajari et al., 2015). Hence, we can improve our estimation using newer method in the future and continue focusing on this area.

References

[1]Ackerberg, D. and L. Benkard. "Econometric Tools for Analyzing Market Outcomes." *Handbook of Econometrics*, 2007, *6a*(07), pp. 4171-276.

[2]Bajari, P., D. Nekipelov, P. Ryan and M. Yang. "Demand estimation with machine learning and model combination." *NBER Working Paper*, 2015, No. w20955.

[3]Berry, S. and P. Haile. "Identification in Differentiated Products Markets Using Market Level Data." *Econometrica*, 2014, *82*(5), pp. 1749-97.

[4]Berry, S. and P. Haile. "Identification in Differentiated Products Markets." *Annual Review of Economics*, 2016, 8, pp. 27-52.

[5]Berry, S. "Estimating Discrete-Choice Models of Product Differentiation." *RAND Journal of Economics*, 1994, *25*(2), pp. 242-62.

[6]Berry, S. and P. A. Haile. "Nonparametric Identification of Multinomial Choice Demand Models with Heterogeneous Consumers." *Ssrn Electronic Journal*, 2009, (1718).

[7]Berry, S., J. Levinsohn and A. Pakes. "Automobile Prices in Market Equilibrium." *Econometrica*, 1995, *63*(63), pp. 841-90.

[8]Berry, S., J. Levinsohn and A. Pakes. "Differentiated Products Demand Systems from a Combination of Micro and Macro Data: The New Car Market." *Journal of Political Economy*, 2004, *112*(1), pp. 68-105.

[9]Berry, S., J. Levinsohn and A. Pakes. "Voluntary Export Restraints on Automobiles: Evaluating a Trade Policy." *American Economic Review*, 1999, *89*(3), pp. 400-30.

[10]Budget and Legislative Analyst's Office. "Analysis of the impact of short-term rentals on housing". *http://sfbos.org/sites/default/files/FileCenter/Documents/52601-BLA.ShortTermRentals.051315.pdf*, 2015, pp. 1-54.

[11]Bundorf, M. K. and N. Mahoney. "Pricing and Welfare in Health Plan Choice." *American Economic Review*, 2008, *102*(7), pp. 3214-48.

[12]Dubé, X., Jean-Pierre, J. T. Fox and C. Su. "Improving the Numerical Performance of Static and Dynamic Aggregate Discrete Choice Random Coefficients Demand Estimation." *Econometrica*, 2012, *80*(5), pp. 2231-67.

[13]Edelman, B. G. and M. Luca. "Digital Discrimination: The Case of Airbnb.Com." *Ssrn Electronic Journal*, 2014.

[14]Eizenberg, A. "Upstream Innovation and Product Variety in the U.S. Home PC Market." *Review of Economic Studies*, 2014, *81*.

[15]Ert, E., A. Fleischer and N. Magen. "Trust and Reputation in the Sharing Economy: The Role of Personal Photos in Airbnb." *Tourism Management*, 2016, *55*, pp. 62-73.

[16]Gentzkow, M. and J. M. Shapiro. "What Drives Media Slant? Evidence from U.S. Daily Newspapers." *Econometrica*, 2010, *78*(1), pp. 35-71.

[17]Hastings, J., T. J. Kane, and D. Staiger. "Heterogeneous preferences and the efficacy of public school choice." *NBER Working Paper*, 2009, No.2145.

[18]Kaplan, RA. and ML. Nadler. "Airbnb: A case study in occupancy regulation and taxation." *U. Chi. L. Rev. Dialogue* 82 (2015): 103.

[19]Lancaster, K. "A New Approach to Consumer Theory." *Journal of Political Economy*, 1966, *74*(2), pp. 132-57.

[20]Lancaster, K. "Consumer demand: A new approach.", Columbia University Press, New York, 1971.

[21]Lustig, J. "The Welfare Effects of Adverse Selection in Privatized Medicare." *Department of Economics Working Paper*, 2008.

[22]Mcfadden, D. "Conditional Logit Analysis of Qualitative Choice Behavior." *Frontiers in Econometrics*, 1972, pp. 105-42.

[23]Mcfadden, D. "Econometric Models for Probabilistic Choice Among Products." *Journal of Business*, 1980, *53*(3), pp. 13-29.

[24]Neilson, C. "Targeted vouchers, competition among schools, and the academic achievement of poor students." Yale University, 2013.

[25]Nevo, A. "A Practitioner's Guide to Estimation of Random-Coefficients Logit Models of Demand." *Journal Of Economics & Management Strategy*, 2000b, *9*(4), pp. 513-48.

[26]Nevo, A. "Mergers with Differentiated Products: The Case of the Ready-To-Eat Cereal Industry." *The RAND Journal of Economics*, 2000a, pp. 395-421.

[27]Reynaert, M. and F. Verboven**.** "Improving the Performance of Random Coefficients Demand Models: The Role of Optimal Instruments." *Journal of Econometrics*, 2014, *179*(1), pp. 83-98.
[28]Vincent, D. W. "The Berry-Levinsohn-Pakes Estimator of the Random-Coefficients Logit Demand Model." *Stata Journal*, 2015, *15*(3), pp. 854-80.
[29]Zervas, G., D. Proserpio and J. W. Byers**.** "The Rise of the Sharing Economy: Estimating the Impact of Airbnb on the Hotel Industry." *Journal Of Marketing Research*, Received: December 29, 2016

Appendix: Python and R codes

In this paper, I apply Python to crawl Airbnb consumer data as demographic variables and instruments. And R is used to clean dataset and implement the estimation of random coefficient demand model. The codes are shown as following 3 parts:

1. Data Crawling Program in Python

This part of program contains 5 python file: *Run_with_pool.py*, *Control.py*, *meancut.py*, *Request_and_Soup.py*, *Calculate.py*.

Run_with_pool.py

```python
# -*- coding: utf-8 -*-

import re
from multiprocessing import Pool, Queue, Process, Manager
from Control import control_spider
from time import sleep
from meancut import meancut
import requests
import datetime


def main(nameList):
    countm = 0
    manager = Manager()
    q = manager.Queue()
    while 1:
        np = 30
        pool = Pool(processes=np)

        while 1:
            try:
                url0 = 'http://tpv.daxiangdaili.com/ip/'
                payload = {'tid': '557064921442104',
                           'num': '50000',
                           'protocol': 'https',
                           'longlife': '1',
                           'foreign': 'none'}
                r = requests.get(url0, params=payload)
                p = re.compile('[\r\n]')
                listip = p.split(r.text)
                listip = [x.encode('utf-8') for x in listip if x != '']

                ipList = meancut(listip, np)
                break
            except:
                sleep(10)
                continue

        for obs in range(np):
            # print nameList
            try:
                iplist = ipList[obs]
                # print nameList[countm * 2 + obs]
                pool.apply_async(control_spider, args=(q, nameList[countm * np +
obs], iplist,))
                sleep(0.1)
            except:
                pass
        countm += 1
        begin = datetime.datetime.now()
        begin.strftime('%Y-%m-%d %H:%M:%S')
        print str(begin) + " now: " + str(countm) + " loop at: " + str(countm * np +
obs) + " End point: " + str(
            len(nameList)) + " **********************************"
        if len(nameList) < (countm * np):
            break
        # print('Pool ' + str(countm) + ' Start')
        pool.close()
        pool.join()
        # print('Pool ' + str(countm) + ' End')
        write(q)


def write(q):
    fw = open('output.txt', 'a+')
    countw = 0
    while not q.empty():
        try:
            value = q.get()
            for element in value:
                fw.write(element)
            countw += 1
            print(str(countw) + " observations have been written\n")
        except:
            print(str(countw) + " observations written failed\n")
            break


if __name__ == '__main__':

    #
#####################################################################################
##########
    # read reviewer_id in .csv file, save dic{id: reviewer_id} and list[reviewer_id]
    p = re.compile('[^\t]+[\t|\n]')
    fr = open('input_gao.csv', 'r')
    count = 0
    name_list = []
    f_dic = {}
    while 1:
        count += 1
        line = fr.readline()
        if count == 1:  # skip the header
            continue
        if line:
            listA = p.findall(line)
            listA = [i[:-1] for i in listA]  # delete \t or \n at the end
            id_D_reviewer_id = listA[1] + "$" + listA[2]  # save as id$reviewer_id
            name_list.append(id_D_reviewer_id)
            f_dic[listA[1]] = listA[2]
        else:
            break
    fr.close()

    sum = 0
    fo = open('output.txt', 'r')
    while 1:
        line = fo.readline()
        if line:
            listB = p.findall(line)
            listB = [i[:-1] for i in listB]
            if f_dic.has_key(listB[0]):  # check whether "id" is in f_dic, if yes,
                name_list.remove(
                    (listB[0] + "$" + listB[1]))  # delete corresponding key in f_dic
and reviewer_id in name_list.
                del f_dic[listB[0]]
                sum += 1
        else:
            break
    print("Finished " + str(sum) + " \'reviewer_id\'s")
    fo.close()

    #
#####################################################################################
##########

    # print name_list
    main(name_list)
```

Control.py

```
# -*- coding: utf-8 -*-

import re
from time import sleep, time
from Calculate import calculate_price_per_guest
from Request_and_Soup import request_then_soup


def control_spider(q, id_d_reviewer_id, iplist):
    current_list = []

    p = re.compile(r'(\d*)\$(\d*)')
    id = p.search(id_d_reviewer_id).group(1)
    reviewer_id = p.search(id_d_reviewer_id).group(2)

    # write current id and reviewer_id in output file
    current_list.append(id + '\t')
    current_list.append(reviewer_id + '\t')

    ticksA = time()
    countip = 0
    while 1:
        ip = iplist[0]
        del iplist[0]
        countip += 1
        try:
            # print "*********listip\n"+ ip
            # print {'http': ip, 'https': ip}
            # print("Trying ip(" + str(countip) + "):" + ip)

            try:
                dic = {'http': ip, 'https': ip}
                url = 'https://zh.airbnb.com/users/show/' + reviewer_id
                soup = request_then_soup(url, dic)
                # get users sign-up time
                re_sign_time = re.compile(r'注册时间：(.*)')
                data_sign_time =
re_sign_time.search(str(soup)).group(1).decode('string_escape')
                print("Success at " + ip)
                # write users sign-up time
                # except:
                #     current_list.append('\t')
                #     print 'Failed @ write users sign-up time'
                #     pass
                break
            except:
                continue

        except Exception, e:
            print Exception, ":", e
            ticksB = time.time()
            if ticksB - ticksA > 90:
                break
            if len(iplist) == 0:
                print("All ip have been used")
                break

    current_list.append(str(data_sign_time) + '\t')


    # get users sign-up time
    try:
        re_sign_time = re.compile(r'注册时间：(.*)')
        data_sign_time =
re_sign_time.search(str(soup)).group(1).decode('string_escape')
        current_list.append(str(data_sign_time) + '\t')  # write users sign-up time
    # except:
    #     current_list.append('\t')
    #     print 'Failed @ write users sign-up time'
    #     pass
    except Exception, e:
        print Exception, ":", e

    # get number of wish lists and host reviews, maybe contain guest reviews
    try:
        mydivs = soup.findAll('div', {'class': 'social_connections_and_reviews'})
        re_wish_and_review = re.compile(r'\d*(?=\)<\/small>)')
        data_wish_and_review = re_wish_and_review.findall(str(mydivs))
        current_list.append(str(data_wish_and_review) + '\t')  # write number of wish
lists and host reviews
    except:  # TODO: improve output by re
        current_list.append('Error\t')
        # f.write('\t')
        print 'Failed @ write number of wish lists and host reviews'
        pass

    # get wish list url list TODO: only get 3 urls for now, others are hidden in html
    try:
        re_wish_url = re.compile('\/wishlists\/\d*')
        wish_url = re_wish_url.findall(str(soup.findAll(href=re_wish_url)))
        # Example: ['/wishlists/179659793', '/wishlists/182048270',
'/wishlists/182235459']
        current_list.append(str(wish_url) + '\t')  # write wish list url
    except:
        current_list.append('Error\t')
        print 'Failed @ write wish list url'
        pass

    # calculate mean price in wish lists, CNY/person, currency depends on your
country
    # try:
    if wish_url:

        price_per_guest = []
        for suburl_wish in wish_url:
            # sleep(0.5)
            full_wish_url = 'https://zh.airbnb.com' + suburl_wish
            price_per_guest.extend(calculate_price_per_guest(full_wish_url, dic))
        wishlist_price_per_guest = sum(price_per_guest) / len(price_per_guest)
        current_list.append(str(price_per_guest) + '\t')  # write mean price for each
wish lists
        current_list.append(str(wishlist_price_per_guest) + '\t')  # write mean price
in wish lists
        ipf = open('ipf.txt', 'a+')
        ipf.write(ip + '\n')
        ipf.close()
    else:
        current_list.append('\t')
        current_list.append('\t')
        pass
    # except:
    #     current_list.append('Error\t')
    #     current_list.append('Error\t')
    #     print 'Failed @ calculate mean price in wish lists'
    #     print Exception, ":", e
    #     pass
    # except Exception, e:
    #     print Exception, ":", e
    #     print 'Failed @ calculate mean price in wish lists'
    #     current_list.append('Error\t')
    #     current_list.append('Error\t')
    #     pass

    # url List of hosts who reviewed on this guest page TODO: only get 10 urls for
now, others are hidden in html
    try:
        re_hostreview_url = re.compile('\/users\/show\/\d*')
        hostreview_url = re_hostreview_url.findall(
            str(soup.findAll('div', {'class': 'avatar-wrapper'})))  # url List of
hosts who reviewed on this guest page
        # ['/users/show/60594796', '/users/show/102996772', '/users/show/822444',
'/users/show/87357']
        current_list.append(str(hostreview_url) + '\tFinished\n')  # write the url
List
    except:
        current_list.append('\tFinished\n')
        print 'Failed @ url List of hosts who reviewed on this guest page'
        pass

    q.put(current_list)
    print 'Put reviewer_id ' + str(id_d_reviewer_id) + ' to queue'
```

meancut.py

```
# -*- coding: utf-8 -*-

def meancut(a, N_part):
    L=len(a)
    L_part = L/N_part
    b = []
    for i in range(0,len(a),L_part):
        b.append(a[i:i+L_part])
    c = [b[x] for x in range(N_part)]
    return c
```

Request_and_Soup.py

```
# -*- coding: utf-8 -*-

import requests
```

```
from bs4 import BeautifulSoup


def request_then_soup(url, dic):
    hdr = {
        'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.11 (KHTML, 
like Gecko) Chrome/23.0.1271.64 Safari/537.11',
        'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
        'Accept-Charset': 'ISO-8859-1,utf-8;q=0.7,*;q=0.3',
        'Accept-Encoding': 'none',
        'Accept-Language': 'en-US,en;q=0.8',
        'Connection': 'keep-alive'}
    html = requests.get(url, headers=hdr, proxies=dic, timeout=2).content
    soup = BeautifulSoup(html, "html.parser")
    return soup
```

Calculate.py

```
# -*- coding: utf-8 -*-

import re
from operator import truediv
from Request_and_Soup import request_then_soup


def calculate_price_per_guest(full_wish_url, dic):

    soup_wish = request_then_soup(full_wish_url, dic)
    price_wish = str(soup_wish.findAll('span', {'data-pricerate':
'true'})).encode('utf-8')  # 心愿单中每个房源的价格，注意货币单位
    wish_data = (re.compile(r'\"\d*\"\>\d*(?=\<\/span\>)')).findall(price_wish)
    guest_wish = str(soup_wish.findAll('span', {'class':
'detailWithoutWrap_basc2l'})).decode('unicode_escape').encode(
        'utf-8')
    guest_data = (re.compile(r'\"\d*\"\>\d*(?=位房客)')).findall(guest_wish)

    data_guest = []
    data_wish = []
    for i in range(len(wish_data)):
        id_guest_i = int(re.search(r'\"(\d+)*\"', guest_data[i]).group(1))
        guest_i = int(re.search(r'\>(\d+)', guest_data[i]).group(1))
        data_guest.append([id_guest_i, guest_i])
        id_wish_i = int(re.search(r'\"(\d+)*\"', wish_data[i]).group(1))
        wish_i = int(re.search(r'\>(\d+)', wish_data[i]).group(1))
        data_wish.append([id_wish_i, wish_i])

    # print('data_guest: ' + str(data_guest))
    # print('data_wish: ' + str(data_wish))
    data_guest.sort()
    data_wish.sort()
    final_data_guest = []
    final_wish_guest = []
    for i in range(len(wish_data)): final_data_guest.append(data_guest[i][1])
    for i in range(len(wish_data)): final_wish_guest.append(data_wish[i][1])
    # print('final_data_guest: ' + str(final_data_guest))
    # print('final_wish_guest: ' + str(final_wish_guest))

    price_per_guest = map(truediv, final_wish_guest,
                          final_data_guest)  # TODO: Maybe can be faster in other
ways: http://stackoverflow.com/questions/1975250/when-should-i-use-a-map-instead-of-
a-for-loop
    # print('price_per_guest: ' + str(price_per_guest))
    # print 'price_wish: '
    # print price_wish  # Example: [<span data-pricerate="true" data-
reactid="67">262</span>, <span data-pricerate="true" data-reactid="156">637</span>,
<span data-pricerate="true" data-reactid="209">361</span>]
    # print 'guest_wish: '
    # print guest_wish
    # print guest_data

    return price_per_guest
```

2. Data Cleaning Program in R

dataclean.R

```
# This code forms the dataset used in my paper.
# Written by Gao Fangshu, Mar 2017.

location <- "C:\\Users\\Fangshu Gao\\Desktop\\output\\"

calendar <- read.csv(paste(location,"calendar.csv",sep = ""), stringsAsFactors =
FALSE)
listings <- read.csv(paste(location,"listings.csv",sep = ""), stringsAsFactors =
FALSE)
reviews <- read.csv(paste(location,"reviews.csv",sep = ""), stringsAsFactors = FALSE)

calendar1 <- calendar

calendar1[calendar[ , 3] == "t", 5] <- TRUE
calendar1[calendar[ , 3] == "f", 5] <- FALSE

# count available days of each room
available_day <- tapply(calendar1$V5, calendar1$listing_id, sum)

# calculate mean price of each room
calendar1[ , 6] <- as.numeric(substring(calendar1$price,2))
mean_price <- tapply(calendar1$V6, calendar1$listing_id, mean, na.rm = TRUE)

# merge available days and mean price
idMP <- data.frame(levels(as.factor(calendar1$listing_id)), mean_price)
idAd <- data.frame(levels(as.factor(calendar1$listing_id)), available_day)
idMPAd <- merge(idMP, idAd, by.x = colnames(idMP)[1], by.y = colnames(idAd)[1])
```

```
rooms <- calendar1[!duplicated(calendar1$listing_id), ]

rooms1 <- merge(rooms, idMPAd, by.x = colnames(rooms)[1],
                by.y = colnames(idMPAd)[1])

rooms2 <- rooms1[ , c(-6:-2)]

# choose variables I need
listings1 <- listings[ , c("id", "price", "host_since", "host_response_rate",
                          "host_acceptance_rate", "host_is_superhost",
                          "host_has_profile_pic", "accommodates","bathrooms",
                          "bedrooms","beds", "security_deposit","cleaning_fee",
                          "guests_included","extra_people", "minimum_nights",
                          "number_of_reviews", "review_scores_rating",
                          "review_scores_accuracy", "review_scores_cleanliness",
                          "review_scores_checkin", "review_scores_communication",
                          "review_scores_location", "review_scores_value",
                          "instant_bookable", "cancellation_policy",
                          "reviews_per_month")]

# calculate market share, by reviews_per_month
sum_reviews_count <- sum(listings1$reviews_per_month, na.rm = TRUE)
listings2 <- cbind(listings1, listings1$reviews_per_month/sum_reviews_count)
colnames(listings2)[length(colnames(listings2))] <- "market_share"

# form the DATA, merge listing2 to rooms2
DATA <- merge(rooms2, listings2, by.x = colnames(rooms2)[1],
              by.y = colnames(listings2)[1])

rm(list = setdiff(ls(), c("DATA", "location")))

# ------------------------------------------------------------------------------

output_w_20170304_0 <- read.csv(paste(location,"output_w_20170304_0.txt",sep = ""),
                                header = FALSE, sep = "\t",encoding = "UTF-8",
                                stringsAsFactors = FALSE)
output_w_20170304_1 <- read.csv(paste(location,"output_w_20170304_1.txt",sep = ""),
                                header = FALSE, sep = "\t",encoding = "UTF-8",
                                stringsAsFactors = FALSE)
output_w_20170305_0 <- read.csv(paste(location,"output_w_20170305_0.txt",sep = ""),
                                header = FALSE, sep = "\t",encoding = "UTF-8",
                                stringsAsFactors = FALSE)
output_w_20170305_1 <- read.csv(paste(location,"output_w_20170305_1.txt",sep = ""),
                                header = FALSE, sep = "\t",encoding = "UTF-8",
                                stringsAsFactors = FALSE)
output_g_20170309_0 <- read.csv(paste(location,"output_g_20170309_0.txt",sep = ""),
                                header = FALSE, sep = "\t",encoding = "UTF-8",
                                stringsAsFactors = FALSE)

data_1 <- output_w_20170304_0[ ,c(1:4, 6, 7)]  # has "Error" and colunm problem
data_2 <- output_w_20170304_1[ ,c(1:3, 5, 7, 8)]
data_3 <- output_w_20170305_0[ ,c(1:3, 5, 7, 8)]  # "has Error"
data_4 <- output_w_20170305_1[ ,c(1:3, 5, 7, 8)]
```

```
data_5 <- output_g_20170309_0[ ,c(1:3, 5, 7, 8)]

# clean data_1
# choose rows without "Error" or other problems
data_1 <- data_1[(data_1[,6] != "\t") & (data_1[,6] != "[]") & (data_1[,6] !=
"Error"), ]
data_1 <- data_1[(data_1[,1] != "Finished") & (substr(data_1[,4], 5, 5) != "年"), ]
double_date <- output_w_20170304_0[substr(output_w_20170304_0[,4], 5, 5) == "年", ]
rest <- double_date[double_date[,8] != double_date[1, 8], c(1:3, 5, 7, 8)]
data_1 <- data_1[data_1[,6] != data_1[1, 6], ]

# clean data_2
data_2 <- data_2[(data_2[,6] != "\t") & (data_2[,6] != "[]") & (data_2[,6] !=
"Error"), ]
data_2 <- data_2[data_2[,6] != data_2[1, 6], ]

# clean data_3
data_3 <- data_3[(data_3[,6] != "\t") & (data_3[,6] != "[]") & (data_3[,6] !=
"Error"), ]
data_3 <- data_3[data_3[,6] != data_3[1, 6], ]

# clean data_4
data_4 <- data_4[!is.na(data_4[,6]), ]

# clean data_5
data_5 <- data_5[!is.na(data_5[,6]), ]

newname <- function(data) {
  colnames(data) <- c("id", "reviewer_id", "signup_date", "count_list", "p_person",
"p_wlist")
  return(data)
}

data_1 <- newname(data_1)
rest <- newname(rest)
data_2 <- newname(data_2)
data_3 <- newname(data_3)
data_4 <- newname(data_4)
data_5 <- newname(data_5)

scrape_data <- rbind(data_1, rest, data_2, data_3, data_4, data_5)

rm(list = setdiff(ls(), c("DATA", "location", "scrape_data")))

# ------------------------------------------------------------------------------

idmatch <- read.csv(paste(location,"reviews.csv",sep = ""),
                    header = TRUE, stringsAsFactors = FALSE)[,c(1,2,4)]

scrape_iddata <- merge(scrape_data, idmatch[,1:2], by.x = colnames(scrape_data)[1],
by.y = colnames(idmatch)[2])

# transfer p_person to a p vector
```

```r
tranp <- function(p_person) {
  tranp <- strsplit(substr(p_person, 2, nchar(p_person) - 1), ", ")
  return(tranp)
}

# calculate sum of price and number of people
sump <- function(vecp) {
  sump <- c(sum(as.numeric(vecp)), length(vecp))
  return(sump)
}

pper_and_pnum <- lapply(tranp(scrape_iddata$p_person), sump)

vpper <- unlist(lapply(pper_and_pnum, function(x) x[1]))
vpnum <- unlist(lapply(pper_and_pnum, function(x) x[2]))

# calculate average price per person among listing_id
tpper <- tapply(vpper, scrape_iddata$listing_id, sum)
tpnum <- tapply(vpnum, scrape_iddata$listing_id, sum)
wish_price_per_person <- data.frame(names(tpper/tpnum), tpper/tpnum, stringsAsFactors
= FALSE)
names(wish_price_per_person) <- c("listing_id", "wish_price_per_person")

iscrape_iddata_pperp <- merge(DATA, wish_price_per_person, by.x = colnames(DATA)[1],
                              by.y = colnames(wish_price_per_person)[1])

rm(list = setdiff(ls(), c("iscrape_iddata_pperp")))

# -----------------------------------------------------------------------------
paper_data0 <- iscrape_iddata_pperp[!is.na(iscrape_iddata_pperp[,12]), ]
paper_data0[paper_data0[,12] == 0, 12] <- 1
accomm_per_bedrooms <- paper_data0[,10]/paper_data0[,12]/10
paper_data0 <- cbind(accomm_per_bedrooms,paper_data0)

paper_data <- paper_data0[, c("listing_id", "available_day", "host_response_rate",
                                       "host_is_superhost", "accomm_per_bedrooms",
                                       "minimum_nights", "review_scores_rating",
                                       "instant_bookable", "cancellation_policy",
                                       "market_share", "wish_price_per_person",
"mean_price")]

# data normalization
paper_data <- paper_data[paper_data[,3] != "N/A", ]
paper_data[,2] <- paper_data[,2]/365
paper_data[,3] <- as.numeric(substr(paper_data[,3], 1, nchar(paper_data[,3]) -
1))/100
paper_data[paper_data[,4] == "f", 4] <- 0
paper_data[paper_data[,4] == "t", 4] <- 1
paper_data[,4] <- as.numeric(paper_data[,4])
paper_data <- paper_data[paper_data[,6] != 1000, ]
paper_data[,6] <- paper_data[,6]/10
paper_data[,7] <- paper_data[,7]/100
paper_data[paper_data[,8] == "f", 8] <- 0
```

```
paper_data[paper_data[,8] == "t", 8] <- 1
paper_data[,8] <- as.numeric(paper_data[,8])
paper_data[paper_data[,9] == "strict", 9] <- 1
paper_data[paper_data[,9] == "moderate", 9] <- 2
paper_data[paper_data[,9] == "flexible", 9] <- 3
paper_data[,9] <- as.numeric(paper_data[,9])
paper_data[,11] <- paper_data[,11]/1000
paper_data <- paper_data[!is.na(paper_data[,12]), ]
paper_data[,12] <- paper_data[,12]/1000
paper_data <- paper_data[!is.na(paper_data[,7]), ]

MEAN <- apply(paper_data[,c(12,2:9,11)], 2, mean)
MEDIAN <- apply(paper_data[,c(12,2:9,11)], 2, median)
SD <- apply(paper_data[,c(12,2:9,11)], 2, sd)
MAX <- apply(paper_data[,c(12,2:9,11)], 2, max)
MIN <- apply(paper_data[,c(12,2:9,11)], 2, min)

t(rbind(MEAN, MEDIAN, SD, MAX, MIN))

rm(list = setdiff(ls(), c("paper_data", "iscrape_iddata_pperp",
                          "MEAN", "MEDIAN", "SD", "MAX", "MIN")))

# # > unlist(lapply(HHH[1:2], function(x) x[1]))
# # [1] 107 107
# # > unlist(HHH[1:2])
# # [1] 107   1 107   1
#
# tapply(c(1,2,3,4,5,6), c(3,2,1,3,2,1),mean)
```

3. Estimation Program in R

blp.R

```
# Written by Gao Fangshu, Mar 2017.

library(MASS)

k <<- 9  # number of x
j <<- 1300  # number of product
d <<- 1  # number of demographic variables
i <<- 1000  # number of sampling
l <<- 9  # number of instrument
tol_1 <<- 1e-15

# import data
mv <<- matrix(abs(rnorm(k*i)), ncol = i, nrow = k)
md <<- matrix(paper_data$wish_price_per_person, ncol = d, nrow = j)
mx <<- matrix(unlist(paper_data[,c(12,2:9)]), ncol = k, nrow = j)
sn <<- matrix(paper_data$market_share, ncol = 1, nrow = j)
z <<- matrix(unlist(paper_data[,c(11,2:9)]), ncol = l, nrow = j)
```

```
# parameter: pi, msigma
cpi <- abs(rnorm(k*d))/10

# data_transform <- function(mv, md, mx, sn, z,)

csigma <- abs(rnorm(k))/10

# calculate
calms <- function(mpi, msigma, delta) {
  # interact part between x and d, interd is a j*1 matrix
  interd <<- diag((mx %x% matrix(1, ncol = d, nrow = 1)) %*%  (mpi %*% (matrix(1,
ncol = 1, nrow = k) %x% t(md))))
  minterd <- matrix(rep(interd, i), ncol = i, nrow = j)

  # delta should be j*1, then copy the column for i times to form mdelta

  mdelta <- matrix(rep(delta, i), ncol = i, nrow = j)
  mdelta_0 <<- matrix(delta, ncol = 1, nrow = j)

  # j*i matrix of (delta_j + x*sigma*v + x*pi*d)
  ss <- mdelta + mx %*% (diag(as.vector(msigma)) %*% mv) + minterd

  # exp(each elements in ss)
  exps <- matrix(exp(as.vector(ss)), ncol = i, nrow = j)

  # sum exps at j, result is 1*i matrix
  sjexps <- matrix(1, ncol = j, nrow = 1) %*% exps

  # copy the rows of sjexps and form a matrix of denominators, result is j*i matrix
  de <- 1 + matrix(rep(as.vector(sjexps), j), ncol = i, nrow = j, byrow = TRUE)

  ed <<- exps/de

  # vector(j*1 matrix) of sj
  ms <<- (1/i) * (ed %*% matrix(1, ncol = 1, nrow = i))

}

cmap <- function(cpi, csigma) {

  mpi <- diag(cpi)
  msigma <- matrix(csigma, ncol = 1, nrow = k)

   if (di == 0) {
     delta <- abs(rnorm(j))/1000
     mdelta_0 <<- delta
     mdelta_1 <<- matrix(abs(rnorm(j))/1000, ncol = 1, nrow = j)
     print(!all(mdelta_1 - mdelta_0 < tol_1))
     print("di=0")
   }
  # delta <- rep(0,j)
  # mdelta_0 <<- delta
  # mdelta_1 <<- matrix(rep(1,j), ncol = 1, nrow = j)
  print("original value ready")

  di <<- di + 1

  i = 0

  while (!all(abs(mdelta_1 - mdelta_0) < tol_1)) {
    calms(mpi, msigma, mdelta_1)
    mdelta_1 <<- mdelta_0 + log(sn) - log(ms)
    i <- i + 1
    if (i > 10000) {
      print("delta cannot converge :(")
      break
    }
  }
  Dd <- mdelta_1 - mdelta_0
  ApD <<- cbind(ApD,Dd)
  print("converged in iteration")
  return(mdelta_1)
}

GMMs <- function(param) {  # param = [csigma,cpi]

  cpi <- param[1:k]
  csigma <- param[(k+1):(k+k*d)]

  mdelta_1 <<- cmap(cpi, csigma)

  beta <- ginv(mx) %*% mdelta_1  # GLOBAL problem
  Beta <<- beta
  omega <<- mdelta_1 - (mx %*% beta)

  # GMM objective function (1*j) %*% (j*l) %*% ((l*j) %*% (j*l)) %*% (l*j) %*% (j*1)
  Optm <- t(omega) %*% z %*% ginv(t(z) %*% z) %*% t(z) %*% omega
  return(Optm)

}

param <- c(csigma,cpi)
ApD <- c(1:1300)
di <- 0

result <- nlm(GMMs, param)
print(result)

# Calculate SE, referring to formula, see David W. Vincent(2015) Stata paper 861-862
DsjDdeltaj <- (1/i) * ((ed*(1-ed)) %*% matrix(1, ncol = 1, nrow = i))
DsjDdeltam <- matrix(0, ncol = j, nrow = j)
for (idraw in 1:i) {
  DsjDdeltam <- DsjDdeltam + ((ed[,idraw] %*% matrix(1, ncol = j, nrow = 1)) *
                                  (matrix(1, ncol = 1, nrow = j) %*% ed[,idraw]))
}
DsjDdeltam <- DsjDdeltam/i
```

```
Dtheta2delta1 <- DsjDdeltam - diag(diag(DsjDdeltam)) + diag(DsjDdeltaj)  # j*j matrix

DsjDsigmak <- matrix(0, ncol = k, nrow = j)
for (idraw in 1:i) {
  sumxde <- matrix(1, ncol = j, nrow = 1) %*% (mx *
matrix(rep(as.vector(ed[,idraw]),k), ncol = k, nrow = j))
  ixxp <- mx - matrix(rep(as.vector(sumxde), j), ncol = k, nrow = j, byrow = TRUE)
  vik <- matrix(rep(as.vector(mv[,idraw]), j), ncol = k, nrow = j, byrow = TRUE)
  ied <- matrix(rep(as.vector(ed[,idraw]), k), ncol = k, nrow = j)
  DsjDsigmak <- DsjDsigmak + (ied * vik * ixxp)
}
DsjDsigmak <- DsjDsigmak/i  # j*k matrix

DsjDpik <- matrix(0, ncol = k, nrow = j)
for (idraw in 1:i) {
  sumxde <- matrix(1, ncol = j, nrow = 1) %*% (mx *
matrix(rep(as.vector(ed[,idraw]),k), ncol = k, nrow = j))
  ixxp <- mx - matrix(rep(as.vector(sumxde), j), ncol = k, nrow = j, byrow = TRUE)
  dik <- matrix(rep(as.vector(md), k), ncol = k, nrow = j)
  ied <- matrix(rep(as.vector(ed[,idraw]), k), ncol = k, nrow = j)
  DsjDpik <- DsjDpik + (ied * dik * ixxp)
}
DsjDpik <- DsjDpik/i  # j*k matrix

Dtheta2delta2 <- cbind(DsjDsigmak, DsjDpik)  # j*(2k) matrix

Dtheta2delta <- -ginv(Dtheta2delta1) %*% Dtheta2delta2  # j*(2k) matrix

GtTheta <- ginv(z) %*% (cbind(mx, Dtheta2delta))  # l*(k+2k) matrix

# (k+2k)*l %*% (l*j %*% j*1 %*% 1*j %*% j*l) %*% l*(k+2k)
VarTheta <- ginv(t(GtTheta) %*% ginv(t(z) %*% omega %*% t(omega) %*% z) %*%
GtTheta)*i

SE <- sqrt(diag(VarTheta))
t <- c(as.vector(Beta[,1]), as.vector(result["estimate"]),recursive = TRUE)/SE
# 1.64 1.96 2.58
```

Acknowledgements

First, I would like to express my sincere gratitude to my advisor Professor Mengxia Zhang who spent much time and enthusiasm in offering me much valuable advice and guidance. And my special thanks go to Professor Mo Zhou, who leaded me into the field of random coefficient demand model.

This thesis would not have been finished without the contributions of several people. My friend Shenghao Chen helped me a lot on web crawling programming, which is indispensable to my thesis. And thanks to Yifan Tang, from whom I got much inspiration about this topic. Of course, I really appreciate Shuang Xu's company, who always encourages me when I meet difficulties.