creative commons
COMMONS DEED

저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer

dCollection

A DISSERTATION FOR THE DEGREE OF
DOCTOR OF TOURISM SCIENCE

Platform Home Sharing on Jeju Island: An application of revealed preferences to the sharing economy

Jacob Charles Barr

Department of Tourism Management

GRADUATE SCHOOL

JEJU NATIONAL UNIVERSITY

2018. 02

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

Platform Home Sharing on Jeju Island: An application of revealed preferences to the sharing economy

Jacob Charles Barr

(Supervised by Professor Choi, Byoung-Kil)

A dissertation submitted in partial fulfillment of the requirement for the degree of Doctor of Tourism Science

FEBRUARY 2018

This thesis has been examined and approved.

2018. 02

Department of Tourism Management

GRADUATE SCHOOL

JEJU NATIONAL UNIVERSITY

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

Acknowledgements

To my advisor, Profesor Choi, Byoung Kil: thank you for choosing me as your student. Your trust and frank guidance throughout this process was a something on which I could always rely.

To Professor Park, Ounjoung: thank you for supporting me, giving me the opportunity to teach, and for countless office hours of advice.

To Jeong, Jong Hwan: thank you for editing my manuscripts again and again.

To Caleb Botton: thank you for your encouragement, your criticism and empathy. Most of all, thank you for the title of this dissertation.

To Candace: Меине бесте Фрей ндин, меине Гелиебте - данке, дасс Ду ес мит мир аусгехалтен хаст. Их лиебе Дир. Пай тинг!

Jacob Charles Barr

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

ABSTRACT

The sharing economy has been heralded as a social movement that is transforming consumer behavior norms. The purpose of this study was to empirically define the presence of platform home sharing on Jeju Island and to analyze this new market within the theory of revealed preferences to see if platform home sharing is driven by economic or social-experiential considerations. Are platform home sharing guests on Jeju Island exhibiting social or utilitarian consumer behavior through their purchase decisions? Web scraping and data mining techniques were employed to obtain a reliable data set, and multivariate regression analysis was employed to estimate the most important determinants in platform home sharing businesses in the local context. In order to further explore the relationships between property listing characteristics and the number of reviews as the dependent variable, classification and regression trees were employed to account for the nonlinearity of the data.

The results show that platform home sharing on Jeju Island is widespread across the entire island with heavier concentrations on the northeast side and a thinner distribution on the south west side. Consequently, latitudinal coordinates were more effective determinants of demand than longitudinal. Among the 31 listing property types. “houses” which are priced below 140,000 KRW per night are more likely to be rented than any other type. Furthermore, the host speaking additional languages to Korean is perceived as an asset in terms of the number of reviews. English and Russian are better determinants of demand than other foreign languages. The platform home sharing guests are most strongly motivated by low cost and entire properties, rather than shared spaces. Shared rooms were revealed as least preferred among the guests with less than 10% of the total listings. Entire homes, however, occupied more than 50% of the entire population in terms of listings

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

and 60% of the number of reviews. This shows that the new market of platform home sharing on Jeju Island is driven by traditional economic forces of maximizing satisfaction and minimizing cost, and not the alternative social and or experiential collaborative consumption motivations commonly associated with the Sharing Economy. Further inferential analyses on hosts who list multiple properties as well as sensitivity analyses on the efficacy of satisfaction as a determinant of demand were performed. The analysis yielded that hosts who list multiple properties did not directly receive a higher average number of reviews, however, they did receive a lower overall level of satisfaction. As the presence of platform home sharing on Jeju island continues to grow, policy makers and hoteliers would benefit from understanding a new potential source of competition in the market for tourism accommodation. Several implications are discussed in terms of the analyses as well as limitations and directions for future research on platform home sharing.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

CONTENTS

I. INTRODUCTION..1

1. Background..1

2. Research Purpose and Methods...6

3. Research Constructs..8

II. LITERATURE REVIEW..9

1. Platform Business Model...9

1) Four main categories of innovation...10

2) Disruptive Innovation's Essential Elements..13

3) Issues with Disruptive Innovation..14

4) Platforms...16

5) Traditional and Platform Accommodation..18

6) Three Pillars of Platforms..20

7) Platform Disruption...21

8) Moving Towards Open Innovation...25

2. Smart Tourism..27

1) Smart Tourism on Jeju..28

2) Phenomenon-based Research...30

3) User-Generated Content..33

4) Transitions of the Web..33

(1) Web 2.0...33

(2) Towards a Cooperative Web..34

(3) Web 3.0...36

3. The Sharing Economy...37

1) Motivational Factors in the Sharing Economy..37

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

2) Theories of Consumer Behavior and Revealed Preferences................44
3) Decision Trees used in Predictive Statistical Modeling.......................52
4) Airbnb on Jeju..53
5) Literature End Note..54

III. METHODOLOGY..55

1. Research Analysis..55
2. Hypothesis...56
1) Research Model..56
3. Data...57
1) Description of Data..58
4. Data Collection...64
1) Geographic Parameters...65
2) Scraping...66
(1) Configuring the Scraper...67
(2) Implementing the Scraper...69
(3) Adding a Survey and collecting information...70
(4) Exporting Room Information..73
5. Methods of Analysis..73

IV. ANALYSIS OF DATA..75

1. Descriptive Analysis..75
1) Hypothesis Test...83
2. Inferential Analysis...93
1) Stepwise Regression..93
2) CHIAD Regression Trees...102
3) Summary of Results..110

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

V. CONCLUSION..112

1. Conclusion..112

1) Findings..112

2) Discussion..113

(1) Satisfaction and Accommodation..114

(2) Multi-Listers..115

2. Implications..116

3. Limitations and Future Research..119

REFERENCES..122

APPENDIX A..134

APPENDIX B..143

APPENDIX C..146

APPENDIX D..149

APPENDIX E..164

APPENDIX F..184

APPENDIX G..191

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

<LIST OF TABLES>

<Table 2-1> Types of Innovation......10
<Table 2-2> Platform Home Sharing Motivations......38
<Table 3-1> Room Types......59
<Table 3-2> Bounding Box......66
<Table 3-3> Imported Modules to the Script......67
<Table 3-4> Scrapable Variables......72
<Table 4-1> All Listings......77
<Table 4-2> Sample of Listings with Review Threshold......79
<Table 4-3> Differences Between Population and Sample......81
<Table 4-4> Distribution of Shared Rooms......85
<Table 4-5> Distribution of Private Rooms......87
<Table 4-6> Distribution of Entire Homes......89
<Table 4-7> Geographic Areas of Concentrations......91
<Table 4-8> List of Property Types......94
<Table 4-9> Summary of Inferential Statistical Results......110

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

<LIST OF FIGURES>

<Figure 1-1> Research Constructs............8
<Figure 2-1> Theory of Disruptive Innovation............12
<Figure 2-2> Platform Model............24
<Figure 2-3> Open Innovation............26
<Figure 2-4> Model for Implementing Smart Tourism............30
<Figure 2-5> Smart Tourism............32
<Figure 2-6> Theory of Revealed Preferences............48
<Figure 2-7> Weak Axiom of Revealed Preferences............48
<Figure 3-1> Research Analysis............55
<Figure 4-1> All Reviews' Density Distribution............76
<Figure 4-2> Price Distribution of Entire Population............78
<Figure 4-3> Price Distribution of Three of More Reviews............80
<Figure 4-4> Distribution of Listings by Review Threshold............82
<Figure 4-5> Distribution of Shared Rooms............86
<Figure 4-6> Distribution of Private Rooms............88
<Figure 4-7> Distribution of Entire Homes............90
<Figure 4-8> Areas with High Concentrations of Listings............92
<Figure 4-9> Stepwise Regression for Reviews............95
<Figure 4-10> Stepwise Regression with Satisfaction>0............97
<Figure 4-11> Stepwise Regression for Reviews>3............99
<Figure 4-12> Satisfaction and Host Rooms............101
<Figure 4-13> Satisfaction as a Function of Multi-Listers............101
<Figure 4-14> Fewer Reviews Associated with Lower Satisfaction............102
<Figure 4-15> Property Type CHIAD Tree............105
<Figure 4-16> Additional Languages CHIAD Tree............107
<Figure 4-17> Geographic Position CHIAD Tree............109
<Figure 5-1> Log Price Review Distribution............118

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

I . INTRODUCTION

1. Background

"What a company owns matters less than what it can connect" – Alex Moazed

According to the United Nations World Trade Organization (UNWTO, 2015), tourism is "a social, cultural and economic phenomenon which entails the movement of people to counties or places outside of their usual environment for personal business or professional purposes". Tourism in the phenomenal context frames this research in the best way, as a situation that is observed to exist, yet one whose explanation is in question. The form of tourism that we experience today has changed dramatically from the traditional forms of brick and mortar travel agencies, national airlines and hotels of previous generations. The tourism of today may be characterized by an information-intensive industry, highly dependent on communication technologies and connectivity. As more and more people travel, the industry has broadened to include alternative tourism planning, alternative tourism experiences and most relevant to this research, alternative tourism lodging.

At certain moments in history, the confluences of technological and social advances have paved the way for new forms of innovation which transform traditional markets. The 'Sharing Economy' is one of these movements and has taken root in the tourism, travel and hospitality industry and seen phenomenal growth in the past five years. The term has been widely used in numerous forms to express various nuances in exchange (e.g. collaborative

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

economy, collaborative consumption, peer-to-peer marketplaces, peer economy, on-demand economy etc.), however, the underlying concept is a platform approach to connect buyers and sellers, producers and consumers and to create a community of purpose. Sociologically, theorists have postulated why more people are choosing to participate in the 'sharing economy', be it for the universal trust generated through interactions or social capital rendered by strengthening weaker community ties (Codagnone et al., 2016; Granovetter, 1973). The trust-building aspect through mutual accountability in the sharing economy has been one of the key drivers for its success, and makes the sharing of one's house to a stranger not such a foreign concept. This unprecedented level of cooperation among consumer peers in modern society has left regulatory bodies and competing traditional business owners in disarray in deciding how to restrict, monitor or curtail these 'sharing'practices. Moreover, the various stakeholders involved are at variance in regard to the positive or negative spillover effects for society as a whole that micro-entrepreneurs who rent their homes may have. The erosion of traditional labor contracts, tax base, security risks and unmitigated innovation are all items of tensions and trade-offs in the debate surrounding home sharing (Cohen et al., 2016; Sunil & Noah, 2015).

The terms Airbnb and Uber have largely become household names across the world and have fueled the discourse on modern micro-entrepreneurship, income diversification and the role of regulation (Codagnone, 2016; Foroohar, 2016). The ride sharing, pseudo-taxi company, Uber, despite regulatory efforts, has experienced phenomenal growth in the local transportation industry (Helft, 2017. Airbnb, famous for peer-to-peer home sharing and vacation rentals has lodged itself in the industry as a middle class mainstay with a market capitalization of more than $30 billion in 2017 (Tsang, 2017). In direct comparison to major hotel industries, this is $10 billion more than Hilton's market cap and approximately $5 billion less than the Marriott

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

group's market capitalization. What sets Airbnb apart, however, is how asset-light the company has become.

The platform paradox plays well to the advantage of the tech unicorns. Uber is the world's most highly valued private technology company, with Airbnb coming in second (Economist, 2017). The companies do not own hundreds of thousands of vehicles or properties but can help leverage individual owner's underused assets to profitable use. Both companies benefit from virtually zero marginal costs and have limitless scalability. The additional cost of adding one more unit of supply to the platform is virtually zero, and the platform can expand to the size of the market without incurring additional costs. This asset light - profit heavy dynamic sets platforms apart from traditional supply chain business models.

The platform is the predominant business model of the 21st century. It is a model wherein the company does not own its supply chain, but rather facilitates interactions between a network of third party producers to exchange value. In most cases, it does not create the value that is being produced. The transformative nature of the platform business model, in what has become known as the sharing economy, is a step from the digital to the physical world. The 'Internet of Things' refers not only to network interfaced home appliances but also other physical assets such as cars and accommodation. What Airbnb and Uber have done since 2009 is they have taken the digital, peer-to-peer networks of the previous decade (Napster, YouTube, Facebook etc.) and have applied them to the physical world. This business model is not well understood in its "disruptive" capacity to incumbent industries, especially in the tourism and hospitality industry. The rapidly growing presence of informal tourism accommodation, or home sharing, is an unprecedented phenomenon which lacks a set of managerial theories and data collection methodologies to effectively monitor its development and form cohesive policies around it, particularly from a local

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

level. This is due to the decentralized nature of supply that the platform attracts and the difficulty of obtaining reliable data to analyze the characteristics of this supply.

Recent research has called for academics to take a step back from preemptively jumping to conclusions based on theoretical implications, and rather to explore empirically driven results to see how the platform home sharing market is developing. Before a proper assessment of the impacts of the sharing economy is to take place, access to the platform and user data currently held by the platform is a necessary prerequisite (Frecken, 2017). This research contributes to the greater academic discourse by developing a repeatable methodology to access and analyze the pertinent platform data for policy makers to form data-driven decisions.

This research seeks to explain the new market of platform home sharing on Jeju Island in terms of the *homo economicus* approach to the theory of consumer behavior. This approach posits that consumers are primarily driven by the economic forces of cost savings and maximizing satisfaction. This has been the leading theoretical approach in classical economics since Adam Smith; however, due to the lack of empirical studies, it is not yet proven to be the predominant approach to consumer behavior in what has become known as the 'Sharing Economy' or 'collaborative consumption', which is allegedly driven by more experiential and social factors. In adherence with the economic theory of revealed preferences, which states that a consumer's preferences are revealed by their purchasing habits, this research explores the purchasing habits of more than 70,000 platform home sharing purchase events on Jeju Island to determine which type of property is most frequently rented.

Jeju is the largest and southernmost inhabited island isolated off the coast of the Korean mainland. It has an area of 1,845 square kilometers and a local population of 610,000 residents (Adedoyin, 2017). The population, however is Jeju growing quickly, with Jeju City's population alone expected to surpass

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

500,000 in 2018 (Elder, 2017). Much of this growth is attributed to young families moving from mainland Korea and a domestic rising birth rate on the island as well as IT companies relocating to the island.

Topographically, the island is covered in volcanic cones with the dormant Hallasan volcano occupying both the center of the island and the tallest mountain in South Korea. Its ecological significance has been recognized by several international agencies and granted various levels of protection and conservation statuses including the Biosphere Reserve in 2004, the World Natural Heritage in 2007, and the Global Geopark as well as the Rasmar Wetlands in 2011 and 2015 respectively (Kim, 2015; Barnes, 2013). The island has been experiencing a rapid growth of tourism since 2005, when the annual visitors numbered 5 million. Ten years later the number of tourists visiting the island was nearly 13 million (Adedoyin, 2017). The growth in the islands largest economic sector has led to an increase in local jobs and GDP, however, it has also spurred debate as to the ecological consequences and cost of living for local residents. Therefore, the focus in 2017 seems to be shifting from 'how much developed' to 'how well developed' and policy makers are searching for solutions mitigate the undesirable effects of coastline degradation and carrying capacity.

Jeju has a strong entrepreneurial sector wherein startups abound and small business owners compete in a creative environment over growing demand. In alignment with Jeju's ambitions to become an international tourism destination, an analysis of the most prolific example of a tourism and travel platform with the widest network of domestic and international users provides the most meaningful results. Tourism destination managers can stumble from short-sighted policy goals when they view the accommodation value pie as "fixed" and overly protect traditional providers, rather than seeing the untapped potential of creating new value through home sharing.

Jeju Island has also recently experienced a boom in the number of home

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

sharing listings, with 5,568 listings as of October, 2017, in the world's largest platform network provider at a time when South Korea is in the process of forming its regulatory framework regarding home sharing businesses. This research seeks to analyze the characteristics of the supply of this platform network on Jeju island, inform the theory of consumer behavior and provide policymakers with the first-ever quantitative results regarding supply in a local context.

2. Research Purpose and Methods:

Grounded in the theories of Consumer Behavior *homo economicus* and the theory of Revealed Preference, this dissertation seeks to prove one hypothesis.

In forming the research hypotheses for this dissertation several underlying objectives were addressed. First, from a Tourism Management perspective, it was necessary to determine if platform home sharing on Jeju Island should be regarded in terms of normative *homo economicus* consumer behavior, or as a social phenomenon that has negligible presence in the local economy. In order to ascertain this, the most popular type of platform home sharing listing in terms of number of reviews, be it entire homes, private rooms or shared rooms needed to be established. This is how consumer behavior preference would be established for the platform home sharing market on Jeju Island. Consumers who are more interested in fostering social relationships by staying with their hosts in shared spaces would make purchase decisions which reflect this inclination. Conversely, consumers who are pursuing lower costs and convenience above social interaction would reveal their preferences by purchasing entire homes. Accomplishing these research objectives and discussing their implications is the purpose of this dissertation.

The data collection and analytical methods employed by this research have

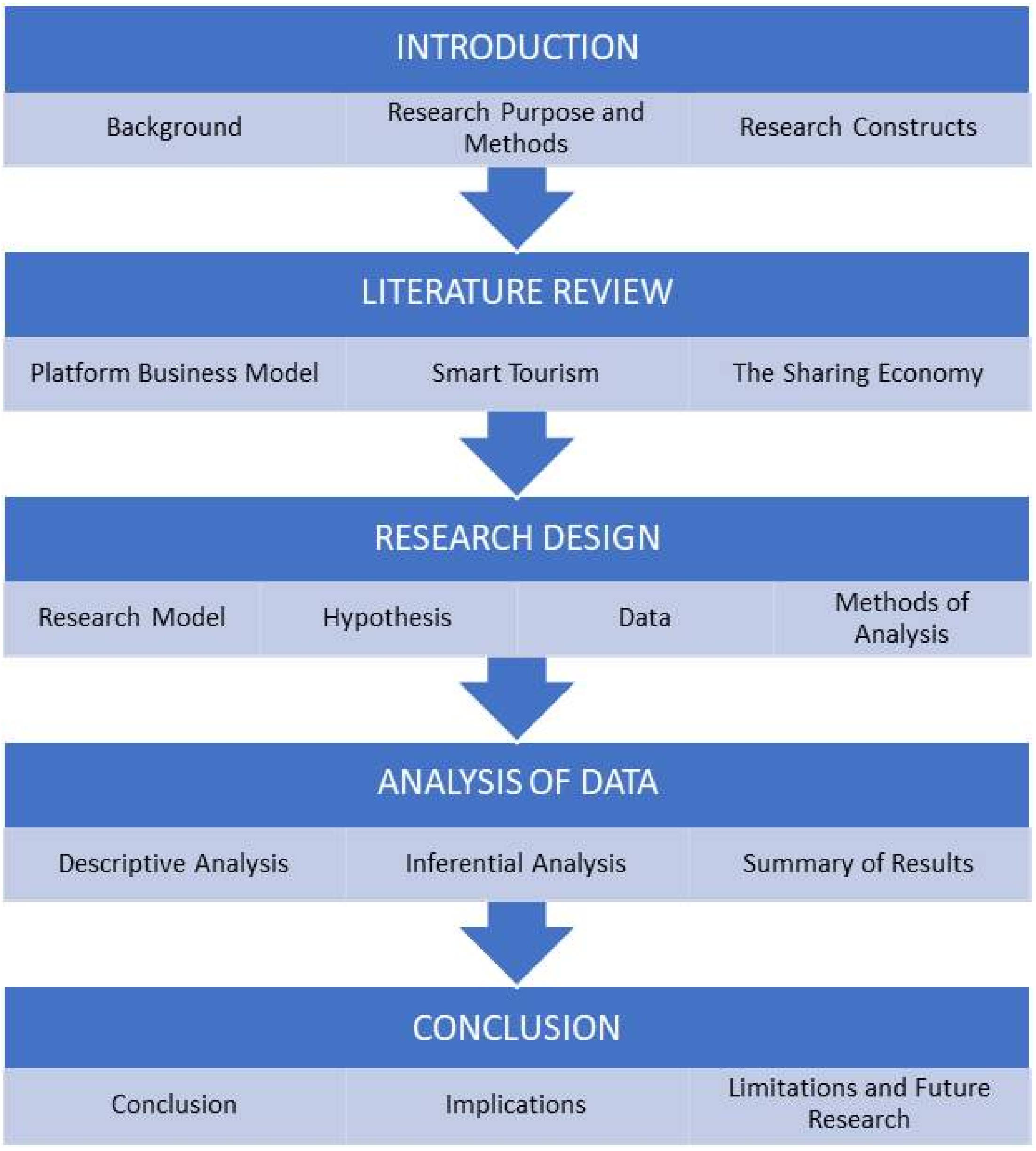
제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

seen little application in the field of Tourism Management and offer a unique contribution to the science in terms of big data analysis. A web scraping technique developed through the Python computer programming language was able to obtain the data. An SQL schema developed through the open source object-relational database management system PostgreSQL was able to categorize the data. The open source programming language for statistical computing, R, was able to parse the data. An open source QGIS system was able to geographically plot the data. Finally, stepwise categorical regression analyses through the statistical package software Stata were able to make productive inferences from the data. This research construct and progression of this dissertation is further illustrated in the section below.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

3. Research Constructs

This study consists of 5 chapters. The content of each chapter and flow of work are shown in <Figure 1-1>.



제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

Ⅱ. Literature Review

Given the unprecedented growth of platform-based home sharing and lack of reputable data collection methodologies, there have been several studies done recently seeking to explore the motivations for using Airbnb, participating in the Sharing Economy and its plausible impact on the traditional hospitality industry (Guttentag, 2015; Zervas, 2016; Stors, 2015). Each of these studies has contributed to a deeper understanding of the dynamics behind platform innovation and consumer behavior theory, but here has been little research in the way of direct empirical studies focusing on the supply side of home sharing, its characteristics, distribution and statistical trajectories. To more fully understand the rapid expansion an implications of platform home sharing in the research, it is necessary to combine academic insights with trends and developments identified through company reports and news media.

Consumer behavior motivational surveys for participating in platform home sharing have repeatedly shown that users are most strongly drawn to its practical rather than experiential aspects (Guttentag, 2016; Gravari-Barbas and Guinand, 2017). Users are attracted by the cost savings first, and social interaction with the hosts second. This is succinctly juxtaposed with Airbnb's marketing emphasis on the social and experiential benefits of using its platform.

1. Platform Business Model

Various studies have also approached platform home sharing from a consumer demand perspective, attempting to frame the phenomenon in terms

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

of conventional business-to-consumer patterns of exchange (Gerwe and Silva, 2016; Yu, 2017), but have met difficulties in reconciling the spectrum of innovation stemming from the platform with a single firm. Platform businesses, however, do not dictate or design the innovative offerings, but rather encourage a wide demographic of users to populate their inventories. For this reason, the theory of disruptive innovation, although initially thought to be readily applicable to platform home sharing, may not be best suited to describe this new mode of economy. It is applicable, therefore, to discuss the commonly accepted innovation typologies and select the most relevant one for the discourse. Due to its pervasive application to emerging technologies and consumer behavior patterns, disruptive innovation is dealt with more extensively in this study.

1) Four main categories of innovation

According to Harvard business scholar, Greg Satell, there are four basic types of innovation: basic research innovation, breakthrough innovation, sustaining innovation and disruptive innovation (Satell, 2017). Each of these types is characterized by set of strategies and applications to which researchers may apply them appropriately.

<Table 2-1> Innovation Types

Breakthrough Innovation Well-defined problem, poorly defined domain	Sustaining Innovation Well-defined problem, well-defined domain
Basic Research Poorly defined problem, poorly defined domain	Disruptive Innovation Poorly defined problem, well-defined domain

Table 2-1 Satell, 2017

Basic research innovation always begins with the observation of some phenomenon, and the path breaking innovations are never fully formed. It is

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

what most universities and doctoral students are engaged with – the pursuit of more fully understanding a unique occurrence. This research pursues basic research innovation of the phenomenon of platform home sharing on Jeju island, for example.

Breakthrough innovation is characterized by a well-defined problem or phenomenon that is difficult to solve or explain within a given paradigm. Social and STEM sciences are advanced through the creation of specific theoretical paradigms, however, the paradigms themselves can often hinder the creation of a solution or an explanation of an issue from within the field it arose (Kuhn, 1996). In these cases, open innovation is instrumental in addressing a phenomenon by exposing it to diverse domains and new ways of framing the issue. This research, for example, attempts to frame the phenomenon of home sharing through open platform innovation. Open innovation will be further discussed and applied to this research in subsequent sections.

Disruptive innovation describes the phenomenon where new market entrants are able to siphon off incumbents'customers by focusing on the bottom line, rather than sustaining innovations which may not appeal to their widest customer base. This research does not frame the phenomenon of house sharing as a form of tourism accommodation on Jeju with a disruptive innovation paradigm, but rather open innovation, given the ecosystem nature of platforms and not the supply chain value propositions of traditional businesses.

Finally, sustaining innovation is the most common type because the majority of businesses are striving to get better at what they are already doing (Satell, 2017). The goal is to improve existing products in existing markets, therefore the problems and techniques to solve them are more clearly defined.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

<Figure 2-1> Theory of Disruptive Innovation

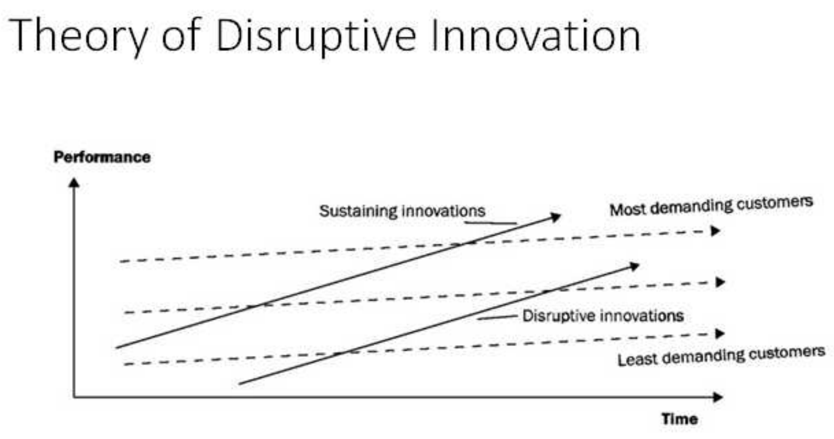


Figure 2-1 Christensen, 1995

Disruptive innovation is a business management theory published by Christensen in 1995 which describes modes of innovation-driven growth. The theory describes the phenomenon when a smaller company with fewer resources is able to successfully challenge an established incumbent (Christensen, 1995 and 2015). Specifically when incumbent firms overly focus on improving services for their most demanding customers (and often most profitable), they exceed the needs of some segments and ignore the needs of others. This provides market entrants with the opportunity to gain a market foothold by targeting those overlooked or ignored customer segments with a product that is 'good enough' - frequently at a lower price. Incumbent firms continue to pursue higher profitability and do not strategically respond to these entrants who appeal to the lower-market, whereas the new entrants move upmarket to provide the functionality and services that more demanding consumers requires while at the same time maintaining their original cost advantage. When the incumbent firm's mainstream customers start to adopt the entrant's offerings in volume, disruption has occurred.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

2) Disruptive Innovation's Essential Elements

Disruptive innovation has four key elements to consider. First, incumbents in a market are improving along a trajectory of sustaining innovation. This means that companies continually strive to provide new and improved products. These are the year-by-year improvements that all competitive companies produce such as increased horsepower or processing chip speeds. The trajectory is always upmarket for higher profits from more demanding and unsatisfied customers. Second, these sustaining innovations often overshoot customers' needs. A company whose products are positioned on mainstream consumers'needs will probably overshoot what most consumers can utilize in the near future. Third, these incumbent firms possess the capability to respond to disruptive threats. Larger incumbent firms have the resources and the ability to respond to potentially disruptive threats, but lack the incentive and initiative to do so. Potential disruptors avoid head-on competition with incumbent firms and thereby often are ignored by complacent larger companies. Fourth, and finally, incumbent firms ultimately fail as a result of the disruption. Performance oversupply by larger firms culminates in the possibility for simpler, more convenient and less expensive firms and products to enter the market and crush the incumbents.

Accordingly, there are three main types of disruption that occur. First, there is high-end disruption. This occurs when a new market entrant debuts a product that is superior to that of the incumbent. This is exceptionally rare as it requires vast amounts of capital to challenge incumbents head-to-head. The second type of disruption is low-end. This occurs when an entrant makes a product cheaper or more simple to use. This form of disruption is more common because they require less start up capital and incumbents often ignore them as they target their least-profitable customers. The third type of disruption is known as 'new market' disruption which emerges from

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

non-consumers and usually creates a category or even industry (Christensen, 1995; Sampere, 2016).

3) Issues with Disruptive Innovation

The theory of disruptive innovation (Christensen, 1995), although fitting for a conventional firm in the 1990's, has some issues in regards to the way modern platform businesses are run, and especially in regards to the operation and management of tourism related businesses. The Economist has called disruptive innovation "one of the most influential modern business ideas" (Economist, 2011).

Christensen's management theory has led academics and business leaders alike in how to implement innovative strategies, but recently questions have arisen regarding the effectiveness of disruptive innovation in describing and predicting how businesses truly operate and how this theory differs substantially from mere competition. Leading up to the most recent wave of criticism, commentators deemed the theory so widely accepted that its predictive power is seldom questioned (Lepore, 2014). The theory has been used in such a variety of contexts that its genuineness is questionable. Few studies have published confirmatory evidence for the theory in its original context, and the lack of evidence for numerical support for disruption, as Christensen claims, is due to the blunt measures in statistical analysis (Christensen, 2006).

Scholars at MIT recently reviewed Christensen's theory and found that many of Christensen's case study companies indeed did not exemplify the four key elements of disruption. While it is true that many business managers tend to ignore 'low-end' disruptors, not all of Christensen's cases have documented "sustaining innovation" practices. In 31% of the cases presented by Christensen after reexamination did not appear to have any

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

semblance of innovative trajectory before they were supplanted by a disruptive innovation (King, 2015). Furthermore. 78% of all the cases drawn upon by Christensen are critically reviewed as not overshooting their customers'needs. Even in the high tech and computer industries, overshooting is, in actuality, arare occurrence. In regards to the third key component of disruptive innovation, after examination it appears that 39% of the quoted companies were, in fact, incapable or disallowed from responding effectively to disruptive innovation. Considering the fourth and final component of disruptive innovation, many of the companies which Christensen examined were indeed displaced by new technologies, but not on the terms which the theory was grounded.

One of the most glaring faults in the theory is the assumption that incumbent firms continually overshoot their customers'needs. It is difficult to imagine a delivery system that is "too quick", a smartphone that has "too long" of a battery life, food that is "too healthy" or accommodation that is "too available" or "too cheap". Christensen posits that 'disruptive' innovation will always win out against 'sustaining'innovation, however, there are too many examples of conventional firms pursuing sustaining innovation in a responsible manner and defeating the potentially disruptive competitor's technological innovation (X-Ray v.s. The MRI, 2.5 disk v.s. The 1.8) (Bower, 1995). Furthermore, the ability to effectively respond to disruption is not always due to cognitive management failures, as posited by Christensen, but structural barriers. In the case with legacy airlines responding to the emergence of low cost carriers, the incumbents would have had to dramatically switch to a new fleet of aircraft, workers, gates and airports. In business management, "sometimes the best response is to adapt, but not dramatically change the business model" (Southwest Airlines, 2015). Inhibiting factors such as legacy costs, shifting demographics and changing economies of scale were the real disrupters which led to the demise of prolific

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

incumbents in Christensen's research.

In forming his theory, Christensen admits that his primary business of analysis, the disk drive, was highly unusual, "nowhere in the history of business has there been an industry like disk drives, where changes in technology, market structure, global scope, and vertical integration have been so pervasive, rapid, and unrelenting." (Christensen, 1997). His research was developed on the assumption that the model of disk drive manufacturers could be superimposed on a macro level across nearly all industries. Christensen's model works well enough when product or service development is clearly defined in a linear supply chain where value moves in one direction, from producer to consumer, and his four aforementioned conditions are met. However, when applied to platform businesses models such as Apple, Alibaba or Airnbnb, the theory is want to adapt. Platforms inherently behave and operate differently than traditional, linear businesses. As opposed to first building and then refining a supply chain, as traditional business models do, platforms grow networks. Platforms derive their supply from this network, and therefore do not control or own their supply in the same way that linear business models do. The conventional theory of disruptive innovation is a demand-side theory of customer dependence and competitive reaction in product markets, not a supply-side theory (Raynor, 2013). While his theory did manifest itself well in this particular high tech and specialized industry, less vertically integrated industries which focus heavily on the supply side as well as the demand are more difficult to categorically place within his linear supply chain model.

4) Platforms

In dealing with platform businesses, Christensen's theory cannot consistently accommodate them. Platform businesses have not one, but two distinct

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

customer groups, their supply network and their demand network. Platforms produce very little, yet they depend on producers as their ecosystem of supply which functionally become customers as well. As the most authoritative source on the subject of disruptive innovation, Christensen defines Apple as a case-in-point example of disruption due, in part, to its innovative app ecosystem, yet disallows Uber (a large taxi-ride sharing platform business) without considering Uber's drivers in the same sense as Apple's app developers (Christensen, 2015). This is inconsistent. Disruptive innovation must be understood in terms of platform, non-linear supply chain businesses, as these are quickly changing how business is done, and are arguably the most disruptive forces in global economics (Shaughnessy, 2016).

All this to say that the theory of disruptive innovation as published and defended by Christensen is not as robust as previously thought in describing change in modern platform businesses. Disruptive innovation as a business theory describing platform phenomenon should not be disregarded entirely, but rather understood and analyzed in its correct context.

"The problem with Borders [a retail bookstore chain in the United States], was that they needed all their customers, and so when Amazon started to siphon them off they simply couldn't cover fixed costs" (King, 2015). This idea of heavy fixed costs is crucial in understanding the advantage of platform-model businesses and their ability to effectively disrupt incumbent industries. The pattern of internet start-ups with low investments and few required physical assets enable most any entrepreneurially-minded individual to establish his or her own company. Additionally, platforms derive their innovativeness from external sources, their independent supply. This form of open innovation leads to radically different approaches to business and the formation of ideas. Open innovation is based on the assumption that an organization cannot just rely on its own resources, but has to engage with partners in order to innovate (Dahlander and Gann 2010; West and Gallagher

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

2006).

5) Traditional and Platform Accommodation

According to Christensen's theory, Apple ought to have been disrupted by more affordable alternatives, Airbnb ought to have substantially disrupted the hospitality industry and Uber does not qualify as an example of disruptive innovation (Christensen, 2015; Moazed, 2016). However, this does not seem to be the case, and the robustness of his theory in terms of platform dynamics is in question. Apple remains highly profitable, despite new market entrants and highly priced phones. Uber has experienced exponential growth globally, but has not substantially affected the taxi industry. Indeed, it appears that demand for self-employed drivers and professional taxi drives has risen across the board, dismissing the notion of a zero sum game (Berger et al., 2017).Finally, although Airbnb is the second most highly-valued private technology company in the United States it does not appear to be cutting into major hotel company profits (Yu, 2017). Theoretically, a simplified process that services to a wider and lower-income customer base should, in time, disrupt the market incumbent who continues to cater towards the highest paying customer.

Hilton and Marriott both have pivoted towards an asset-light approach to providing accommodation, with the majority of their revenue now coming from management and franchise fees. This idea of maximizing franchises in order to minimize liabilities in its portfolio has enabled the hotel conglomerates to remain flexible in the face of platform competition. The success of the brand, however, is tightly tied to the performance of its individual franchises upholding the meticulous attention to detail that has come to be expected from a Hilton or Marriott. This franchise business model works well in areas of the world that have similar standards of living and

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

cultural values, but is difficult to uphold elsewhere. Airbnb, however, thrives in its idiosyncrasies from place to place. Recent research suggests that rather than edging into Marriott and Hilton's margins, Airbnb is targeting a separate market, and both industries are experienced increased demand without destructive interference (Griswold, 2015).

As a platform home sharing business, Airbnb grew out of the collaborative consumption movement, which described the 'rapid explosion in traditional methods of sharing, bartering, lending, trading, renting and swapping' (Rick, 2013). The movement championed the internet's ability to post information about and rent out virtually anything that was not currently in use by the owner, accommodation included. From its inception as a way for two flat mates to afford rent in San Francisco by renting out their spare room in 2008, the company has grown into the single largest accommodation provider in the world. The business model allows for free membership and free access to listings of properties, which has led to rapid growth and market penetration. Airbnb collects revenue from each transaction on their platform, host and guests alike. The revenue model according to Airbnb is as follows:

"We make our money from our service fee. This service fee is what actually goes to the site's operation, enables the platform that we provide, and allows us to offer great customer support before, during and after travel. We charge travelers a 6-12% service fee, depending on the total of the reservation. The higher the total, the lower the percentage of the fee. The reason we scale the fee is so the traveler can save money when booking large reservations. Airbnb also charges the host a 3% fee for every booking that is completed. This fee covers the cost of processing the guest's payment." (Airbnb, 2017) Therefore, in order to maximize the number of transactions the user is only prompted to make a single payment as a service charge after the reservation has occurred. This has insured a smooth purchase process that circulates money apart from any government

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

institutions or banks.

Traditional, linear supply chain businesses often fail not only due to a disruptively innovative entrant, but because of internal legacy costs, external shifting demographics and changing economies of scale. Platform businesses typically lack these inhibiting factors and thereby have a distinct flexibly light advantage that the fixed cost laden incumbents carry. The list of successful platform disruptions to traditional businesses is long: Etsy, Facebook, Google, Instagram, WeChat, Youtube etc., yet it takes an updated model to understand the nuances at work. Understanding platforms as powerful disrupters in a broader sense has important implications.

Disruptive innovation, as a demand-side centric theory, will never fully comprehend the forces at work in platforms. Disruptive innovation indeed is a function of the customers served by the business; however, in a platform model, the supply-side are customers served by the central business. Successful platforms generally do not stop with one service or product, but leverage their networks in ways that traditional supply chains cannot and create markets, value and new customers around their original core transactions (Moazed, 2016). In this sense, platforms are not limited to a single industry, but rather through their extensive networks can suddenly pivot between industries and build new transactions off its existing network. This makes large platforms extremely powerful and unpredictable entities that must be understood, especially from the supply-side in the local context, to fully appreciate their impact and trajectory.

6) Three Pillars of Platforms

Christensen's concept of sustaining innovation is a point of contention when it comes to platform thinking. The proverbial "build a better mousetrap" cannot be so readily applied to platforms since the innovative content

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

generally comes from an uncontrollable supply; however, the platform itself can optimize its design to better facilitate ease of exchange. The concentration on seamless co-creation of value is a defining characteristic of platform models and has been driven by three transformative technologies: mobile, social and cloud.

Mobile is the set of technologies which allow platform users global access to a network of entrepreneurs, consumers as well as platform technicians, irrespective of location. This is much more pervasive than even the most widely distributed franchises. Social can be understood as the effectual working of this mobile technology to connect and people and industries across space. It also manifests itself in identity creation of users on the platform. Finally, cloud is server architecture that houses the entire process so that access of information is not contingent on access of a device. Platform users usually connect via mobile devices to participate in the social network which is hosted by cloud technologies.

Building off the three components of platform architecture, successful platforms tend to have high degrees of connection, gravity and flow (Boncheck, 2013). Connection being how easily others can join a platform to share and transact business. Gravity being how well a platform attracts participants, consumers and producers alike. Flow being how well the platform facilitates co-creation of value. When these three factors function fluidly, a platform generally performs well and a malfunctioning or suboptimal functioning platform can often be diagnosed with a faulty component.

7) Platform Disruption

The distinction at hand is between ‘product’ and ‘platform’. In short, a product is a platform that is used for one or very few products, and a platform is structure upon which variations for many products are built

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

(Sampere, 2016). As was the framework for product-disruption, so it is with platform-disruption; there are three main types: high-end, low-end and new-market. It is more than a semantic difference in discussing product versus platform types; it is a matter of function and purpose. Product type disruptions are challenging enough for incumbents and industries to react to, and platform type disruptions are even more so. The difficulty lies within the broad network nature of platforms. The massive number of people that a successful platform attracts makes regulating them difficult, and even if a platform may be stopped (such as Napster), once the concept is already abroad t is only a matter of time until another entrant adopts it and more legally commercializes it. Therefore, it is imperative that academics and practitioners alike more fully appreciate the shift from product to platform type disruptions.

One of the key tenets of platforms is their wide-reaching networks. A platform with a smaller network of users and producers cannot to a greater extent affect consumer behavior, however, platforms which have reached a critical mass of users can attract a multitude of third party companies from adjunct industries and generate multiple streams of revenue (Zhu, 2016). In aproduct business model, firms create value by developing differentiated products for specific customer needs, and they capture value by charging money for those items. In a platform business model, firms create value primarily by connecting users and third parties, and they capture value by charging fees for access to the platform.

Jeju Island is no exception to the pervasive reach of platform disruption, and seeds have already been planted in one of the island's economic mainstays - tourism accommodation. The case-in-point for this research will be the penetration and impact of Airbnb, a global home-sharing platform, on Jeju. The 2008 California startup is heralded as the most well-funded travel startup in the world, with a market capitalization of over $31 billion, more

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

than Hilton and Hyatt combined (Bensinger, 2017). Airbnb as a platform for home sharing and alternative tourism accommodation is an ideal example of platform disruption at work, and is ripe for analysis.

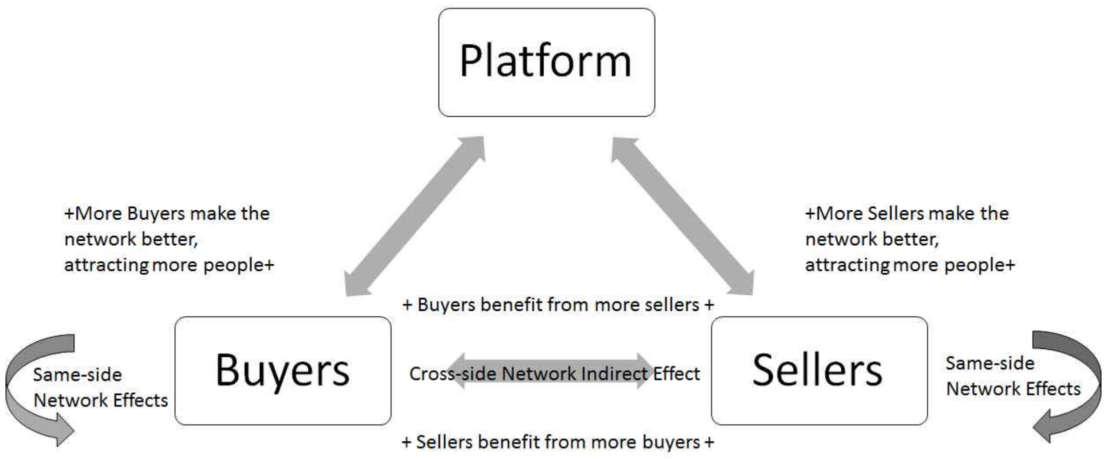
In summary of a recent report on the impact of Airbnb on the traditional hotel industry, global business data analytics company, STR, concluded that "Airbnb is here and it's here to stay. The hotel industry has to look at it as a new way for the traveling public to spend their lodging dollars. It's worth studying them very carefully in your specific submarkets and understanding what the competition in your submarket is like, and understanding that not every unit is necessarily competitive." (Freitag, 2017). An in-depth analysis of the characteristics and trends of Airbnb on Jeju is pertinent research.

In the case of Airbnb, the company can be most closely associated with the third type of platform disruption, new-market. This means that it has popularized and socially charged a new category of business, home sharing, built the network to leverage it and enabled a whole new population of people to derive supplementary income from it. The company brought a shift in emphasis in the tourism accommodation industry from meeting specific customer needs - as traditional accommodation providers do, to encourage mass-market adoption to maximize the number of interactions, thus strengthening their product differentiation and networks, from which they derive their actual value. This is what is known as the 'network effect'of connecting as many users as possible to third party providers (Sundararajan, 2003). The value model for Airbnb is simple. Homeowners derive value from a large enough population of demand customers on Airbnb (the renters). These travelers visiting the area have the potential to give homeowners a consistent flow of business. The renters derive value only when there is enough supply on the network over a wide array of cities, and in each city enough supply to provide several options. When the network obtains enough users (both renters and homeowners), value is created to match

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

accommodation availability with the demand customer's location, budget and time. As the network grows, so does its value and its attractiveness for more users to join. Renters see value in the abundance of homeowners and homeowners see value in the abundance of renters.

<Figure 2-2> Platform Model



Platform
+More Buyers make the network better, attracting more people+
+More Sellers make the network better, attracting more people+
+ Buyers benefit from more sellers +
Same-side Network Effects
Buyers
Cross-side Network Indirect Effect
Sellers
Same-side Network Effects
+ Sellers benefit from more buyers +

Figure 2-2 Author's Illustration

The network effect of Airbnb is part of its three-pronged competitive strategy, which is comparable to most all successful platform businesses: creation of new sources of supply, creation of new user behaviors on the demand side, and architecting a strong curation system (Choudary, 2016). This research primarily focuses on the first prong, the analysis of this new source of accommodation supply in a local context.

As Airbnb continues to grow, it seems to be changing and broadening its users'expectations from traditional, product-oriented service expectations of hotels, to customizable experience-oriented expectations of home sharing. This ability to gauge and bend its users'expectations is what will make the company resilient to disruption itself. Airbnb does not seem to directly compete with hotels, and it insists that it never has (Whiteman, 2014).

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

8) Moving Towards Open Innovation

In the recent past, industry data collection was a highly secretive affair without industrial espionage being as closely guarded against as political espionage (Satell, 2003). Today, however, the industry data culture is beginning to change and open innovation initiatives are being seen as a way to remain competitive in an increasingly diverse business ecosystem. Henry Chesbrough, a California Berkley professor of Business, developed several ways businesses stand to benefit from an open innovation approach (Chesbrough, 2003). Accordingly, open innovation is of particular relevance to this research in three ways. First, open innovation promotes full disclosure of business data and data-driven techniques to. Second, it prescribes an interdisciplinary approach to analyze a current phenomenon in the field. Thirdly, open innovation encourages the establishment of an open API platform, rather than a brand. In regards to the first way, this research strives to make home sharing data on Jeju island publicly available and the techniques on how other tourism science researchers may procure their own data. In regards to the second way, this research uses a wide variety of advanced GIS, Big Data, statistical and computer programing methodologies as well as more traditional Tourism Management theories to reach its conclusions. The interdisciplinary approach insures broader insights into this current business phenomenon. In regards to the third way, the co-creation of value by a wide variety of suppliers to an infinitely diverse consumer base through a platform marketplace is what drives home sharing, and is what deserves attentive research. The tenacity to succeed today seems to be not in closely guarding proprietary assets, but in leveraging them correctly. In the future, the accessibility of Big Data in a consumer-oriented format is what will drive businesses forward and inspire more innovative approaches to further the firm's purpose.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

<Figure 2-3> Open Innovation

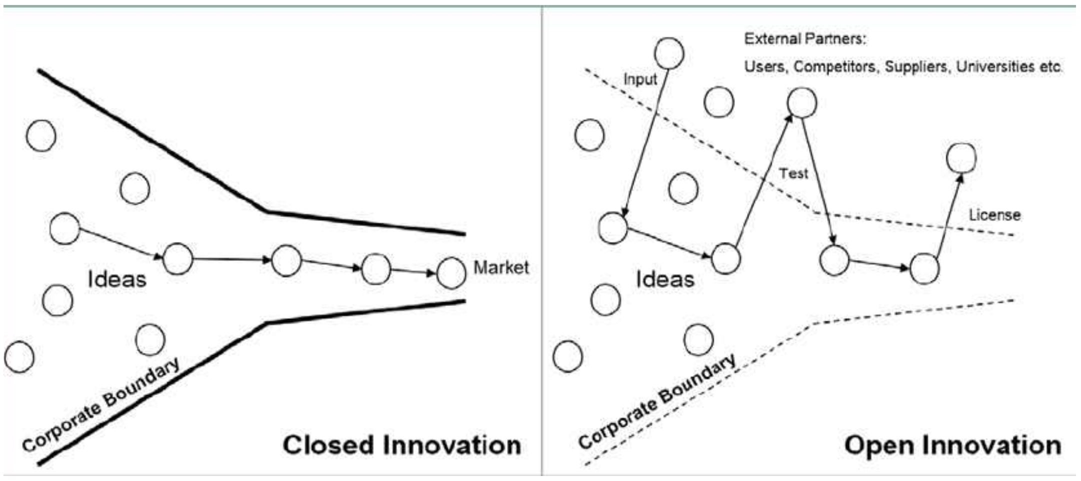


Figure 2-3 Chesbrough, 2003

Closed Innovation largely characterized the first era of industrial innovation, where corporate intellectual property boundaries defined the dispersion of thought and the creative commons. Ideas tended to flow unidirectionally from office inception to market deployment with little to no external proofing. In Open Innovation, ideas generated from within a firm are likely to have external influences, beta stages in test markets and university critiques before they are market deployed. This dispersion of innovation and crowdsourcing creativity enables unprecedented approaches, perspectives and solutions to contemporary issues. The enabling factor for this type of innovation, however, resides in the ability for extra-firm actors to collect and analyze internal firm data, something that can be more readily accessible across a platform business model.

In order to alleviate the increasing pressure of accommodation infrastructure, some smart cities are looking to open innovation to more efficiently manage services and improve the local quality of life for residents

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

(Peek, 2014). The local hackathons or app building competitions are encouraging civic entrepreneurs to develop services that improve the quality of life with the aim of creating sustainable business models.

2. Smart Tourism

Smart Tourism is an associated buzzword which, in recent years, has gained much attention in the tourism and leisure studies. The phrase in and of itself raises some controversial rhetoric, such as what the implied alternative of ‘smart’ tourism may be; however, the sentiment addresses a growing trend in tourism destinations. Public and private tourism destination marketing organizations have begun to rely more heavily on emerging forms of Internet Communication Technologies (ICT) which allow for massive amounts of data to be transformed into new value propositions (Gretzel et al., 2015). This research seeks to inform the growing body of literature in Smart Tourism, in how innovative approaches to Big Data can yield bright insights to an otherwise cloaked home sharing industry.

Aside from the implicit derogatory terminology, one of the conundrums with "Smart Tourism" is that it does not refer to any particular breakthrough technology, but rather an approach to interconnection and synchronization of communication technologies to yield new perspectives (Hoejer and Wangel, 2015). In other words, Smart Tourism is more software than hardware. The ‘smart’ aspect implies integrating and sharing data, using complex analyses, modeling and visualization techniques to make more informed operational and policy decisions. This research seeks to access otherwise obfuscated data and develop cross-disciplinary forms of analysis to open more educated policy discussions in the sphere of home sharing. Cities which have already begun these discussions based on innovative techniques have been labeled ‘smart cities’and have been able to achieve higher resource optimization, fairer

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

governance, sustainability and quality of life by further integrating digital and physical infrastructures (Gretzel etal., 2016). This term has even been applied to entire economies characterized by this phenomenal approach.

Smart tourism has manifested itself in various forms in different world regions. Across the European tourism landscape it has made its appearance in end-user applications, promoting the use of enriched tourism experiences combined with existing databases. In Australia the concentration has been on smart governance and open data. Whereas in Asia, smart tourism generally is most apparent in building technological infrastructure that supports this variety (Hwang, 2015). In this respect, the availability of open data has been a point of contention amongst researchers, local governments and platform businesses. The debate on whether or not a platform business engaged in tourism accommodation is obligated to release its user data is still being discussed (Airbnb, 2016c). Many inroads to smart tourism in the form of periodic data collection, management techniques and cross-over applications of geospatial technologies are just beginning to take root on the local level. At the time of writing, smart tourism is an emerging trend, a phenomenon. In the near future, however, due to the ubiquity of smartphones, mobile internet and the way modern society functions, the semantics of 'smart' will simply turn into 'tourism'.

1) Smart Tourism on Jeju

Jeju Island has become the forerunner for open innovation and smart tourism in South Korea. One of the regional development initiatives of the former South Korean president, Park Geun-hye, was to transform Jeju Island into a world standard for smart tourism and renewable energy by combining information and communication technologies with smart grid energy systems. (Yonghap, 2016). The initiative includes provisions for the "Jeju Creative

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

Economy Innovation Center" and is purposed for fostering information and communication startups as well as commercializing smart-grid energy. Smart tourism, however, has had less of an impact on the island than the Innovation Center or former President Park have envisioned. Moving away from printed brochures and information desks, the Creative Economy initiative is focused on digitalizing Jeju's tourism content and fostering a smart network of guesthouses, communal working spaces and cultural exchanges (Southcott, 2015). In comparison to world-class IT hubs such as Google, Facebook and Amazon, however, Jeju lacks a globalized workforce. Therefore one of the initial challenges remains to provide realtime digital content to international tourists and make information about the island's attractions more accessible to an international audience in a platform approach (Shin, 2017).

A conceptual platform model for how to leverage the open data and digital content of tourism activities on the island is illustrated below <Figure 2-4>. The model is aimed at managing information systems using geographic location-based parameters to guide tourists in making intelligent decisions and encouraging transactions with local residents (Adedoyin, 2017). This model, however, requires the underlying raw data to be available for app developers and local governments alike in order to pass the value to the consumer. The benefit therein is the potential to create a more informed consumer base and foster local entrepreneurship among small and medium-sized enterprises. Also, it can serve to help the local tourism sector to transition from domestic mass tourism to more foreign independent travelers and strengthen a new platform ecosystem in which IT startups can jointly produce value with local community residents through big data analysis.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

<Figure 2-4> Model for Implementing Smart Tourism

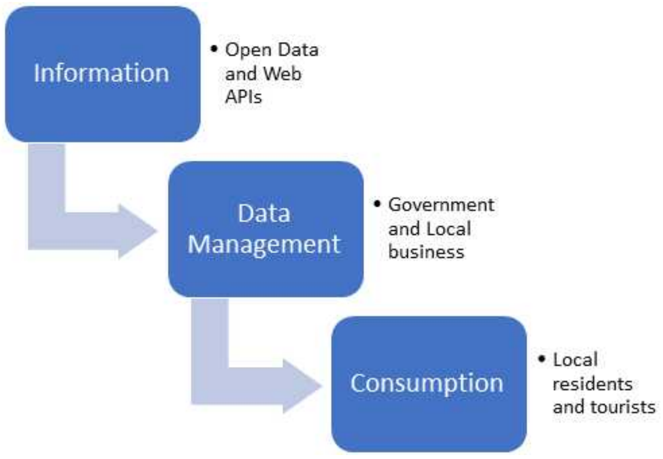

Information
• Open Data and Web APIs
Data Management
• Government and Local business
Consumption
• Local residents and tourists

Figure 2-4 Author's Illustration

2) Phenomenon-based Research

Phenomenon-based research has a mixed reception in academia. On the one hand, phenomenon-based research is ground-breaking in its scope and innovative in its techniques. It first takes current, real world issues into consideration and then applies a single or even set of theories to inform that reality. On the other hand, the absence of strong theoretical underpinnings from the outset open the issue for easy criticism from the wider academic community. As with any topical critique, however, the question of inevitability must be posed. That is, will a cursory dismissal of a topic make it a non-issue? Or will the practice of home sharing as a business continue to grow and affect local tourism-dependent economies irrespective of if academia investigates it? On grounds that a pervasive issue does not have a ready-made, stale theory to address it, make the issue all the more pressing to research.

In order to appropriately theorize a phenomenon, research that accurately captures, describes, documents and conceptualizes a current phenomenon must

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

first take place. Furthermore, research which focuses too strongly on theory may even "prevent the rich reporting of details about interesting phenomena for which no theory yet exists" (Von Krogh et al., 2012) It is this researcher's intent to develop a technique which can insightfully inform a real world phenomenon in the Tourism Management field, and not an incremental adaptation of a well-tried theory-and-methods study. Recently there has been a call for such phenomenon-based research in academic business and management journals. (Doh, 2015). There has been a perceptible dearth of research which focuses on specific events, trends, business evolutions and transformations. Included in this is the growing power of "Big Data" to track, monitor and influence a whole range of tourism related businesses. The sharing economy, as represented by Airbnb and Uber and others, is a prime example of a contemporary phenomenon that provokes research-worthy questions regarding contingency work forces, disruptive innovation and international business theories.

However, before describing the phenomenon of home sharing as alternative tourism accommodation and delving directly into an in-depth case study, it is necessary to first frame the discussion by way of delineation. What is 'Smart Tourism', how is it distinct from other forms of tourism and how does it apply to home sharing?

Smart Tourism may best be understood as a logical progression from e-tourism, which itself stemmed from traditional tourism. The groundwork was already laid with the technological innovations of the 1990's which positioned the tertiary industry into a heavily information and communication dependent service sector. Global distribution and reservation systems, online ticketing and travel review websites all formed what became known as e-Tourism (Werthner and Ricci, 2004).

A succinctly operational definition of Smart Tourism is a convergence of Smart Experiences, Smart Business Ecosystems and Smart Destinations all

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

fed by big data which has been collected, exchanged and processed (Gretzel at al., 2016). Smart Tourism is differentiated from its predecessor, e-Tourism, by relying more on big data than information, by fostering an ecosystem rather than a value chain and intermediaries, and finally by operating chiefly in the experience travel phase (Staab and Werner, 2002), rather than the pre- & post-travel phases.

<Figure 2-5> Smart Tourism

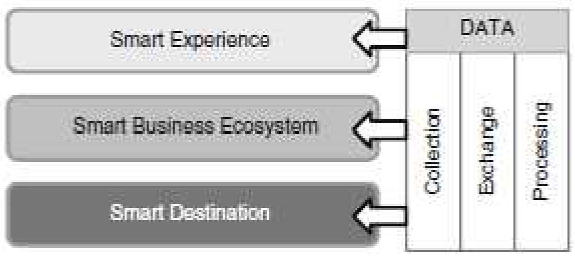


Figure 2-5 Gretzel et al., 2016

This research is primarily focused on the business ecosystem component of Smart Tourism and how platform businesses such as Airbnb in tourism home sharing facilitate a complex ecosystem of supply that manifests itself in a local economy.

As tourism destination cities continue to synchronize their infrastructure to provide more sustainable solutions to growing tourism demand, regionally collected data becomes all the more important for optimal policy-making decisions. Despite the globalized trends and characteristics of international tourism, the sharpest impacts are felt on the local level, where tourism takes place. That is why data collected at the local and regional level is most important for policy makers and business owners. Open Data-driven mobile applications are proving instrumental in providing major tourism destination residents as well as visitors with real-time information on accommodation vacancies, visitor attraction carrying capacities and even available parking

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

statistics for given areas (Deleneuville, 2016). This data is periodically collected, processed and trended in an automatically to provide useful information for consumers of tourist products, not just commercial analytics. The consumer, or end-user, orientation of big data applications are one of the hallmark evolutions of Smart Tourism. To this end, the content generated by users of these platforms, both in terms of reviews and purchasing behavior, is what yields insights into market trends and fosters a more informed consumer base.

3. User-Generated Content

One of the key aspects concerning platform thinking is that of ‘user-generated content’or UGC. Internet startup companies are often thought of as those who are in the business of creating technologies. More accurately, however, they ought be thought of as those who are in the business of leveraging technologies. Most successful internet startups are not the original innovators of the particular strain of technology from which they derive their value, but rather they have found a way to leverage that technology in a new, or more effective way. The underlying common characteristic in virtually every successful internet startup today is how it leverages its UGC.

4. Transitions of the Web

1) Web 2.0

The Web 2.0, a term popularized by Tim O’Reilly (O’Reilly, 2005), is a semantic shift in how internet users interact with web content. It represents a transition from static to participatory content, from a traditional supply chain to a platform supply network and businesses that understand this shift

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

are best positioned to succeed in modern commerce. Web 2.0 is the network as platform, spanning all connected devices – mobile or not; Web 2.0 applications are those that make the most of the intrinsic advantages of that platform and encourage an "architecture of participation," going beyond the page metaphor of Web 1.0 to deliver rich user experience. A whole terminology exists to describe the Web 2.0 dynamic including crowdsourcing, collaborative consumption, sharing economy etc.; however, it is most important to keep in mind the value created by user participation and how the company leverages this.

In this sense, in the era of Web 2.0 there has been a shift in how customers are to be considered. They are no longer merely sources of company revenue, but sources of labor; that is, the user is one who does work. Value creation is then outsourced to an external ecosystem where users participate as both consumers and producers and the company captures this interaction as new value. The particular platform is what hosts or enables this field of interaction and facilitates the exchange of business between consumers and producers. Wikipedia, for example, enables users to create and expand a base of knowledge that is vetted and consumed by other users. Airbnb facilitates commerce and communication between hosts and travelers. Flickr allows photographers to showcase their work and discuss photo production with others. These all point to the central platform business model of connecting consumer and producer roles, rather than producing value alone.

2) Towards a Cooperative Web

Due to the globalization of modern communication and e-commerce, web scraping, as a way to better understand market conditions, has become a ubiquitous practice that has been utilized by researchers across virtually all

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

fields, and is becoming more pervasive in Tourism and Hospitality Management as well. Online User Generated Content (UGC) has become the hallmark of the Web 2.0. Some researchers would further delineate the difference into a series of Web revolutions starting from the era of Web 1.0 (Fuchs et al., 2010). The first era of the Web was largely characterized by cognitive processes (including emotional ones) of humans obtaining information from a computer. The second era of the Web, Web 2.0, was largely characterized by communicative processes being inter-human interaction. These are and were often computer mediated processes. The current stage society is entering may be referred to as the cooperative Web, or the Web 3.0. The cooperative Web 3.0 era may be defined as one where humans and computers work in concert to make more informed decisions, and where human-to-human interactions culminate in more meaningful relationships.

The 'social'aspect of Web 2.0 (social software, social media, social networking etc.) is perhaps misleading and not indicative of the strong community ties one semantically might ascribe to social relationships. That people merely interact with one another, or that technology can facilitate a form of interaction, does not say anything to the quality of that interaction. What distinguishes Web 1.0 from Web 2.0 is the increased interaction between individuals, spurred by new technologies. This largely serves an individual to gain attention in a more social setting. Thus the communicative aspect of Web 2.0 serves greater cognition, rather than the cognitive aspect of Web 1.0 serving to deepen communication of the so-called Web 2.0. The community aspect of Web 2.0 is limited to weaker ties that need not thicken the more interaction occurs. The sense of community remains decidedly interactive and not integrative.

Steve Case, founder of AOL and entrepreneurial evangelist also wrote of this perceptible shift in the leveraging of network relationships in his book

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

entitled Third Wave (Case, 2016). His argument details how in the first wave, congruent with the process of the Web 1.0, established the internet communication technologies and solidified the infrastructure into our society. The second wave leveraged the Internet's networking effect through companies such as Google, Facebook and Amazon. Finally, the third wave society is currently entering is characterized by a rethinking of traditional relationships between companies, governments and citizens. One where ordinary people are empowered through the Internet and associated network technologies to leverage their own creative potential and substantiate more meaningful communities.

3) Web 3.0

Web 3.0 has begun to branch away from the more superficial "communities of interest" and "communities of practice" of the Web 2.0 where members individual actors gather to pursue a common interest and is creating "communities of action" (Fuchs et. al., 2010) where online communication often results offline action. Among numerous other examples, this is the case of Airbnb, where the communicative Web 2.0 aspect of guests engaging online with hosts and other users results in the cooperative Web 3.0 aspects of genuine and often in-person interactions and economic exchange between the interested parties.

Thus, in this 'third wave'of the Web local governments are beginning to view platform ventures not as threats to the local economy, but as tangible opportunities for lower cost solutions to social issues. Additionally, venture platforms are beginning to regard local governments not as barriers to scalability, but as collaborators in the process of cocreation across a wide variety of sectors previously thought to be under the domain of government bodies, such as creating more affordable tourism accommodation that

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

distributes tourism gains more equally through the local economy (Kokalitcheva, 2016; Airbnb, 2016). Such open innovation is only made possible, however, if the source data is made available and the platform-based transactions are understood in their proper context.

3. The Sharing Economy

The sharing economy can be understood in the Web 3.0 sense as a wider community of action. Though the term 'sharing' often engenders notions of gifts, favors and other non-monetary transactions, many peer-to-peer platforms in the sharing economy explicitly involve monetary negotiated exchanges. This, however, does not discount the sharing services based on monetary exchange since the core notion is a matchmaking service and not charity. Some researchers have even concluded that the monetary aspect in the peer-to-peer exchanges of the sharing economy not only do not weaken the intrinsic motivations for participation, but can actually strengthen them and act as a gateway for further interaction (Lampinen, 2016).

Participants in the sharing economy have decided to collaborate in sharing or renting goods and services without a formal legally binding framework, and largely without government intervention or authorization. These collaborative consumers arrange short term exchanges among peers to match physical haves with needs through online platforms, often in a local context. This has sometimes resulted in confrontations with local and national governments in regards to the legality of such exchanges (Ting, 2016).

1) Motivational Factors in the Sharing Economy

The question as to why people choose to participate in the sharing

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

economy has been debated since its inception; however, apart from motivational surveys, there has been little empirical evidence to suggest possible answers. Studies on Airbnb have led to two possible explanations: idealistic motives including the peer-to-peer authentic contact in an accommodation experience, and the other include economic benefits enjoyed by the hosts and the guests. Some have ascribed the consumer's desire for social interaction as the main impetus for growth in platform home sharing (Gansky, 2010; Ikkala and Lampinen, 2015). While others have described it as a way to creatively and progressively connect people (Rothkopf, 2014) and yielding a "taste of the authentic neighborhood life" (Tuttle, 2015).

Guttentag (2016) in his seminal work, explored the issue of motivation for tourists to use the platform home sharing service Airbnb. Through a survey wherein 844 respondents answered questions on motivation, he found that users primarily use the platform because it serves as a low-cost substitute to traditional hotels. Users are motivated by the platform's practical benefits of convenient location, low-costs and household amenities, rather than the platform's socially engaging, environmentally conscientious or community strengthening factors. His research did not dispute the veracity of users' ulterior motives for using the platform's services, but therein identified five types of platform home sharing guests.

<Table 2-2> Platform Home Sharing Motivation

Motivational Type	Characteristics
Money Savers	Choose Airbnb because it is affordable
Home Seekers	Interested in household amenities and space
Collaborative Consumers	Motivated by philosophy and social interaction
Pragmatic Novelty Seekers	Interested in novel accommodation types
Interactive Novelty Seekers	Interested in interaction with hosts and locals

Table 3-2 Guttentag 2016

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

Furthermore, Guttentag's research detailed that 61% of the respondents considered Airbnb to be a suitable substitute for budget to midrange hotels, 70% stayed in "entire homes"rather than shared spaces, and 26% said that staying with the platform led them to increase the length of their trip. Satisfaction levels of "satisfied" or "very satisfied" accounted for 89% of all respondents, confirming a positive skew in Airbnb's trip satisfaction rating system (Zervas, 2016).

The results from the motivation survey reveal a contrast to Airbnb's marketing efforts, which have focused on the platform's social and experiential aspects (Shead, 2016). Guests revealed a preference in the survey to save money first, and to rent entire homes rather than sharing the home with the host.

Brian Chesky, CEO and founder of Airbnb, in an interview inquiring how the company is planning to expand said, "We look to our community and try to figure out what's already popular there for cues as to where to go next." (Tsotsis, 2011). Accordingly, the social sharing aspect of the platform ought to be deemphasized, and the relative cost-savings for value should be at the fore. Shared spaces being generally more economic notwithstanding, survey participants valued their own private space more highly than shared spaces. The question remaining is, apart from motivation surveys, how to measure the consumer behavior of platform home sharing guests. Ultimately, consumers vote with their feet.

Sharing is not a new concept. Giving someone a ride (Uber), having a guest in your spare room (Airbnb), running errands for someone (Task Rabbit), participating in a dinner club (Grub Club) – all of these are not revolutionary concepts. What is new, in the "sharing economy," is that you are not helping a friend for free; you are providing these services to a stranger for money (Sundararajan, 2016). Sharing is commonly conceptualized as a sort of social exchange where no profit occurs, and the actors generally

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

engage in the exchanges based on reciprocity. Sharing is common among family members, tribes, close communities and people of similar religious organizations who share a common identity (Eckhardt and Bardhi, 2015). Also, it generally does not require mediation between parties. The concept essentially results in a reduction of property rights, where both sides agree to occupy or utilize the same object. If the two principle parties have no common identity, require an intermediate actor and exchange money for services or products, the common conception of sharing is obfuscated. In this scenario, the exchange appears to be utilitarian, rather than social in nature. In the case of platform home sharing, the proof in this would be to discover the most popular type of rented property to determine whether social or utilitarian motives dominated the consumer's behavior.

This question has implications not only for academic theorists, but also the sharing economy companies themselves. Airbnb, the most widely used platform home sharing service, has recently rebranded itself to highlight 'people, places, love and community' (McRae, 2016). Other socioeconomic theorists have also begun to laud the sharing economy for strengthening social ties, increasing trust and sentiments of reciprocity (Botsman, 2017), but they run the risk of overestimating the scope and purpose of this access economy by not testing their theories on empirical results beyond motivation survey techniques. If sharing economy consumers exhibited behavior that showed they were seeking interaction with people, community building and love, then the rebranding, marketing campaigns and social theorizing would merit deep discussion. The platforms themselves should not misinterpret their consumers' actions. Platform companies seeking sustainable business practices in an ever more competitive environment ought to recognize whether their customers are most interested in social interaction and community, or if they are simply seeking to make savvy purchase decisions in order to maximize convenience and minimize cost.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

As the industry standard in platform home sharing, Airbnb has led the way how academic theorists have begun to conceptualize the practice of renting one's own house or property to strangers on a short-term basis. Augmenting the basic function of acting as an intermediary between buyers and sellers, Airbnb has transformed its image into a global hospitality brand (Carr, 2014). In pursuit if this, the company has become focused on emitting a sense of "belonging" in its self-branding, even beyond its core function of renting homes. This has evolved into a whole range of experiential activity functions that hosts may post onto the platform and the emblem which the company hopes to become the "universal symbol of sharing" (Kessler, 2014).

This may, however, mislead those who in researching the new "sharing economy" seek to understand its innovativeness, which may lie not in its propensity for its user base to altruistically open their doors to strangers, but in its leveraging of utilize Web 3.0 technologies to dramatically reduce transaction costs (Furchtgott-Roth, 2016). In actuality, true sharing seldom happens among strangers, and policies to enforce sharing seldom result in positive economic conditions. If society functioned in a manner that personal assets were to be shared with strangers, assets would quickly lose their intrinsic exclusive value and incentives to accrue more would diminish. It is hypothesized that, apart from the rhetoric, a platform home sharing host does not wish to "share" his or her home except to the extent that the guest pays for it. The difference then may be the reduction of transaction costs made possible by Internet technologies. Twenty years prior, if a home owner was keen on renting out a part of his home on a short-term basis, the time and energy expended finding a customer may have made the value proposition untenable. Due to the recent innovations in platforms and match making services, the transaction costs of finding customers and hosts has virtually reduced to zero marginal cost. Oliver Williamson and Ronald Coase, Noel Prize laureate economists, both theorized that one of the greatest inhibitors to

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

efficient market outcomes were high transaction costs (Williamson, 1979, Coase, 1937). In this sense, the platform reduces the transaction costs of sharing (renting) one's home. This appears to be more of a classic example of more efficient rent seeking behavior than a manifestation of the "universal symbol of sharing".

Furthermore, apart from the sharing economy rhetoric and academic discourse potentially being misleading, the implications for labor, tourism and policy are wide. The areas which are effectively targeted by the sharing economy are, as some would argue (Das, 2017), inherently inefficient. Over time, the hotel and taxi industry have evolved to server narrower interests of higher paying customers, which leaves room for new low-market entrants to succeed in providing the most basic services for the majority of the market at a lower cost. This is textbook disruption theory. The workers (i.e. hosts and drivers) in the sharing economy are purported to primarily use the platforms to supplement their income (Bellin, 2017), but as contractual workers they are not medically or sustainably employed. A robust sharing economy is then potentially exploitive in that the pricing structure often requires a depressed economy, where abundant laborers are available (Reich, 2016). Other might argue that this form of free contractual economics has a net positive effect on the economy, freeing up employment from corporate intermediaries, and creating more social capital as well. Optimistic advocates of the sharing economy purport a more social utopian dynamic of the sharing economy.

"The sharing economy is not business but a social movement, transforming relationships between people in a new form of internet intimacy and humanitarianism. It builds trust and creates inherently more democratic communities. Customers are not getting cheap services, but being helped by new, interesting friends. Providers are engaged in rich and diverse work, gaining valuable independence and flexibility taking advantage of a reduction

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

in entry barriers to sources of work" (Das, 2017).

The initial web-driven initiatives of social travel such as Couch Surfing were centered on altruistic and adventurous motivations of offering people a chance to stay in your house, and sharing travel experiences (Oksam, 2015). This Goldilocksian concept of strangers sleeping in your bed has been taken to its logical end wherein modern networked hospitality businesses have transformed platform home sharing into a highly lucrative, for-profit enterprise. Classifying Airbnb as social sharing platform may lead to erroneous conclusions and diffuse one's ability to understand the economic effect of the company. When discussing the sharing economy and the companies operating under this umbrella term, it seems to be popular to focus on the 'sharing' so much that the 'economy' is all but forgotten.

The cooperative and social aspects of Airbnb resulting in real world exchanges do merit in-depth studies in the field of Tourism and Hospitality Management. Moreover, however, empirically data-driven studies, such as this research, are purposed to inform the theories and assumptions which academics are seeking to apply to the movement. Moving on from the simpler semantics of user-generated accommodation reviews, content analyses and frequency word clouds, this research seeks to apply big data tools and adaptive web scraping techniques to obtain actual information based on purchasing decisions and business environments. Not based on statistical inferences derived from a limited number of opinionated surveys, this research provides a first application of original empirical data covering an entire population. Furthermore, the techniques developed in this research are seldom utilized in the academia of Tourism and Hospitality Management; however, the field is ripe for their application.

The Theory of Planned Behavior clearly states that the best indicator of a person's actions is his or her intention (Ajzen, 2005). Many contemporary tourism studies have gone on to measure and redefine the theoretical

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

attributes comprising consumer behavioral, normative and control beliefs through numerous Likert scale surveys in order to inform an industry as to plausible consumer behavior. Notwithstanding the veracity and dubious real-world application of such techniques (Amitage, 2010), this research has sought to inform a more direct hypothesis grounded in the theories of consumer behavior and revealed preferences. Namely, that the best indicator of consumer preference is consumer purchasing habits (Wong, 2006).

2. Theories of Consumer Behavior and Revealed Preferences

The theory of consumer behavior is a multi-faceted, fundamental theory of micro economics which posits that a consumer a consumer attempts to allocate his or her limited income among available goods and services as to maximize his or her satisfaction. The particular strain of consumer behavior theory that this research focuses on is known as the *homo economicus* (economic man) approach, and assumes that consumers are highly rational and able to intelligently engage in economic transactions in a manner that is beneficial and of self-interest (Tyagi, 2004). The term *homo economicus* was first used in the late 19th century in regards to a person who was most inclined to maximize his utility whilst expending minimum effort (Persky, 1995). This theory further assumes that these rational consumer actors are knowledgeable of alternative options in their purchasing decisions and are aware of the advantages and disadvantages of each option (Kahle and Close, 2006).

The central question in exploring the issue of *homo economicus* is if human sociality plays a part in his economic behavior, or is it more strictly his individuality and pursuit of self-interest and pleasure. Adam Smith's pioneering work, The Wealth of Nations, describes this discourse in terms of

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

the individual while his Theory of Social Sentiments frames it from his sociality (Smith, 1976a; Smith, 1976b). Other renowned economists have reached differing opinions as to this question as well. Gaeter and Fehr were critical of framing economic consumer behavior from a purest perspective commenting that "a sizable portion of economic actors act on considerations of reciprocity." (Fehr et al., 2000). Whereas Ostrom warns that past policy initiatives intended to encourage collective action and strengthen community sentiments of reciprocity, trust, fairness and cooperation may have been misdirected given the rational egoist's (*homo economicus*) proclivity to maximize his own utility (Ostrom, 2000). In his opinion, this central tendency for man to consistently seek his own individual good may result in in more authentic cooperative behavior rather than a policy-oriented one. Regardless of one's personal persuasion as to the generic nature of man, it is important that with any socio-economic activity to get the core concepts correct. This reality is best informed by data rather than rhetoric.

The most prevalent model for this consumer decision making process was called the 'Utility Theory' wherein consumers are purported to make their decisions based on the expected outcomes of their choices (Loudon et al., 1993). The utility theory originally focused on the sole act of purchase as the manifestation of self-interest seeking behavior. Since the early work of economists Nicholas Bernoulli, John von Neumann and Oskar Morgenstern, the theory of consumer behavior has since expanded to include a wide range of additional consumption activities such as: need recognition, information search, evaluation of alternatives, building of purchasing intention, consumption, disposal in addition to the act of purchasing (Bray, 2007; Zinkhan, 1992). New methodologies, paradigmatic approaches to examining the consumer purchasing process and a rich spectrum of literature have developed around the subject of consumer behavior. While this research acknowledges the progress across the numerous discernable stages the theory of consumer

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

behavior has undergone in the past century, it harkens back to a fundamental stage of the process, namely the fact of purchase under the *homo economicus* typological approach.

As a producer *homo economicus* generally attempts to maximize his profit, and as a consumer his utility. At his core, he is "a being who inevitably does that by which he may obtain the greatest amount of necessaries, conveniences and luxuries, with the smallest quantity of labor and physical self-denial with which they may be obtained." (Mill, 1874). Though the term did not originate with the 18th century economists Adam Smith and David Ricardo, it is often associated with their philosophies. The famous line from the Wealth of Nations, "It is not from the benevolence of the butcher, the brewer, or the baker that we expect our dinner, but from their regard to their own interest" (Smith, 1776) further illustrates the nature and proclivity of *homo economicus*.

Building off of the theory of consumer behavior *homo economicus*, Paul Samuelson's theory of revealed preferences postulated that a consumer's preference is best shown through the consumer's purchasing habits (Samuelson, 1938). Existing theories of consumer demand were based on the laws of diminishing rates of marginal substitution which operate on the assumption that consumers make consumptions decisions in pursuit of maximizing their utility. While this assumption of utility maximization was not in question, the accurate derivation of utility function was seen as problematic (Wong, 2006). Samuelson attempted to free consumer behavior from any traces of utility, that is from the underlying utility functions which cannot be measured with great certainty due to their subjective nature. In lieu of utility, observing the consumer's behavior was a suitable alternative for revealed preference.

Renowned economist Hals Varian (2012) succinctly explains the theory of revealed preference in the following two-dimensional manner. Assume there

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

are two goods, good A and good B in a budget set C. If it be observed that good B is chosen over good A, then good B is directly revealed preferred to good A. Revealed preference theory is a means to reconcile demand theories with utility functions by replacing utility with observations of consumer spending. A further detailed description of the theory can be represented both graphically and in mathematical notation.

A two-dimensional example of the theory is as follows:

Budget set B is defined for two goods $X = X_1, X_2$ and determined by prices p_1, p_2 income m, then let bundle a be $(x_1, x_2) \in B$ and b be $(y_1, y_2) \in B$. This situation would typically be represented arithmetically by the inequality $p_1X_1 + p_2X_2 \leq m$. Assuming strongly monotonic preferences, one only need to consider bundles that graphically are located on the budget line, i.e. where $p_1x_1 + p_2x_2 = m$ and $p_1y_1 + p_2y_2 = m$. If, in this situation, it is observed that (x_1, x_2) is chosen over (y_1, y_2), we can conclude that (x_1, x_2) is (directly revealed preferred to (y_1, y_2), which can be summarized as the binary relation $(x_1, x_2) \succcurlyeq (y_1, y_2)$ or equivalently $\mathbf{a} \preccurlyeq \mathbf{b}$.

Additionally, the theory of revealed preference is underpinned by the Weak Axiom of Revealed Preference (WARP). In order to make sure that the consumer is consistent with his or her preferences, the WARP criterion must first be satisfied. The WARP says that when the consumer's preferences remain the same, there is no budget set, or circumstances, wherein the consumer prefers b over a, if he or she has first chosen a over b when both are affordable. It is such that he or she will never choose b over a as long as prices remain constant. Formally:

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

<Figure 2-6> Theory of Revealed Preferences

$$x \in C(B', \succcurlyeq) \Leftarrow \begin{cases} x, y \in B \\ x \in C(B, \succcurlyeq) \\ x, y \in B' \\ y \in C(B', \succcurlyeq) \end{cases}$$

where x and y are arbitrary bundles and $C(B, \succcurlyeq) \subset B$ is the set of bundles chosen in budget set B, given preference relation $\succcurlyeq$.

Alternatively, if a is chosen over b in budget set B where both a and b are feasible bundles, but b is chosen over a when the consumer faces some other budget set B', then a is not a feasible bundle in budget set B'. This equivalent statement of WARP can formally and more generally be expressed as

<Figure 2-6> Weak Axiom of Revealed Preferences (WARP)

$$p \cdot x(p', m') \leq m \wedge x(p', m') \neq x(p, m) \Rightarrow p' \cdot x(p, m) > m$$

In application to this research, the theory of revealed preference may be understood in terms of the platform home sharing listing types (entire homes, private rooms, shared rooms) which are more frequently chosen over others. In other words, if entire homes are more frequently rented over private or shared rooms, then it can be said entire homes, without the host present, are revealed preferred to the shared spaces. This would show the consumer behavior of platform home sharing to be consistent with *homo economicus* in that the privacy afforded from renting an entire property optimizes the consumer's preference to luxury and self-seeking pleasure, given the affordable alternative to rent shared spaces. The social and experiential aspects gained from renting shared spaces would therefore not be as strong

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

factors in the consumer's purchasing behavior.

When more than two goods are available, the theory of revealed preferences accommodates them in a multi-dimensional model with an assumption of transitivity. Namely if A is preferred over B, and B is preferred over C, then A is *indirectly revealed preferred* over C. The preference for A is ultimate if it can be shown that given the choice between A and B, a consumer will choose B, and given the choice between B and C, the consumer will choose B. 'A' therefore maximizes the consumer's utility in the given budget constraint. In terms of majorities, the first statement:

$$A > B > C$$

indicates a simple majority, where 'A' may possess less than half of the total preference, but more than any other single good. An absolute majority must fulfill an additional prerequisite, namely:

$$B + C < A$$

This indicates that 'A' possesses more than 50% of the total preference.

The philosophical propositions in the growing literature on the peer-to-peer transactions in sharing economy are such that buyers and sellers may choose to cooperate due to ulterior motives, that "the currency of the new economy is trust" (Botsman, 2017). Although factors of efficiency, environmental protection, community development and trust-building are derivative values of platform home sharing, the vehicle by which the majority of all transactions take place may be rudimentary consumer behavior after all. This research seeks to contribute to the growing body of literature on the sharing economy by framing the consumer behavior primarily in terms of conventional self-seeking, utility maximizing purchases rather than the benevolence of the butcher.It is important to correctly categorize this form of economic activity in an appropriate theoretical framework in order to not misrepresent its presence in a local context or misinterpret its impact.

When a potential platform home sharing consumer starts to make a

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

purchasing decision, he or she selects the desired locale on the platform's user interface, enters in the dates of his or her stay and then browses the available results returned by the server. If a particular locale has a strong presence of platform home sharing listings, there are numerous alternative options to choose from and therefore by browsing the returned results, the consumer is knowledgeable of alternative options to his or her purchasing decision. The criticism against *homo economicus*, that modern consumers do not have sufficient knowledge of all the available options in order to make rational purchase decisions may have credibility from a philosophically macro perspective; however, in regards to information seeking on the platform, consumers are presented with adequate alternative lodging options to optimize their purchase decisions.

The ability to measure and analyze consumer behavior and preferences on the platform is possible only through a compilation of customized web scraping techniques that can compile purchasing data, sophisticated structured query language and geographic information systems.

In an attempt to more deeply understand the sharing economy and all of its derivative forms, the European Union commissioned a report in 2016 to systematically review 430 secondary sources covering the various typologies, rhetoric and idiosyncrasies of peer-to-peer markets (Codagnone and Martens, 2016). After thorough analysis, the conclusion reached was there was not sufficient empirical evidence to develop implications for employment or regulations. There is a gap in the empirical research concerning the scope of home sharing, its pricing and market penetration.

The contrary argument to unregulated home sharing is that the peer-to-peer transactions have an unfair advantage to traditional business-to-consumer due to the lack of regulatory restrictions enjoyed by peer-to-peer markets, as well as the consumer protection business-to-consumer markets have to afford. Given the lack of empirical

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

evidence owing to the difficulty of obtaining reliable data, there are no unambiguous answers in this discourse. As peer-to-peer transactions continue to increase and many sharing economy platforms begin to become household names, the need for robust data collection methodologies and empirical studies becomes all the more relevant to research.

Theoretical implications of the sharing economy abound in recent research, speaking into the plausible sustainability of such practices (Heinrichs, 2013), framing the issue in an economic-historical framework (Frenken, 2017), and disruptive potential to traditional lodging businesses (Zervas, 2016); however, there is a lack of direct empirical studies showing the impact home sharing has made in given local economies. Theoretical ponderings on the possible implications of home sharing, studies on user motivations for participating in the sharing economy, demographic surveys on consumer behavior and alternative approaches to the collaborative economy are all useful in the continuing discourse, yet the underlying question to be asked is how deeply entrenched is home sharing in a specific economy. Without this fact firmly established, the practicality of subsequent studies is debatable.

In order to perform proper impact studies and theoretical evaluations, first a reliable data set must be collected. Subsequently, an analysis of the data can be performed. To this end, data mining decision maps and Classification and Regression Trees (CART) are useful in interpreting and applying the data set(Sutton, 2004). Data mining and web scraping generally yield results with a wide spectrum of variables from which regression analyses can be wrought. Web scraping is a method whereby researchers can recursively search websites and extract data pertinent to their research goals. Because such data is generally not readily available in built data sets as secondary data, web scraping enables a researcher to compile primary data from current industry trends. Given the wide variety of the data, non-linear assumptions are critical in making sense of a data set as a whole. Linear regression models can also

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

be rendered, but with a more selected criterion of variables. Regression trees have been used in Tourism Management to forecast tourism demand (Cankurt, 2016), in establishing relationships between an abundance of environmental relationships (Lehmann et al., 2002), and general tourism marketing strategies (Crouch, 1992).

3. Decision Trees used in Predictive Statistical Modeling

In forming a regression tree analysis to interpret the results of web scraping in creating primary data and data mining that predicts the value of a dependent variable based on the values of multiple independent variables, classification trees and regression trees are instrumental. Classification and regression trees are machine-learning methodologies for constructing prediction models from big data. Classification trees are employed when the target variable is categorical and the decision tree is used to identify which class the variable should be appropriated. Alternatively, regression decision trees are employed where the target variable is continuous and the tree can be used to predict its value (Loh, 2011). Regression and classification both deal with a prediction of a variable y in response to a set of predictor variables x. In this research, both categorical and regression trees were employed to analyze the results of the web scraping.

Classification and regression trees have three major tasks: how to partition the data at each step, how to determine when to stop partitioning the data, and how to predict the target variable in each partition (Loh, 2008).These steps and their nodal thresholds are detailed in the data analysis of this research. A Chi-squared Automatic Interaction Detector (CHIAD), originally proposed by Kass (1980), is one of the oldest and most thoroughly used decision tree and was therefore employed for this research. The advantage of

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

a CHIAD tree is that it will accommodate multiple branches attaching to a single root node in order to make more complicated statements. This method has been particularly popular in researching market segmentations and is now being applied to larger data sets in tourism research since it is restricted by its sample size criteria (Perez, 2016 Chung et al., 2004 Kim et al., 2011 do Valle et al., 2012). Due its complex nature in comparison to conventional ANOVA and MANOVA forms of multivariate regression analysis (McCarty and Hastak, 2007), CHIAD has seldom been used in tourism studies, despite its strengths in classifying tourism behavior into appropriate categories (Chen, 2003 Chung et al., 2004 Diaz-Perez et al., 2005).

It is worth mentioning some of the strengths of a CHIAD analysis in determining tourism accommodation characteristics of demand. First, Chi-Square is a form non-parametric statistic. This means that the independent variables do not have to fit a normal distribution and can rely on ordinal and categorical data rather than continuous variable exclusively. Despite this, continuous variables can be chosen as criterion variables, allowing CHIAD to handle a diverse data set. Finally, CHIAD can help researchers establish a set of criterion variables according to the objectives of the researcher (Perez, 2016). This is instrumental in describing effective predictors of a dependent variable such as number of reviews of a particular type of accommodation.

4. Airbnb on Jeju

The leading platform home sharing company in the world, Airbnb, has a strong presence and well-established presence on Jeju Island. As the country's premiere tourism destination, Jeju island government has begun to work with the platform to create a "world-class tourism destination" (Airbnb Citizen Korea, 2017). According to the platform, there are more than 4,000

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

active listings and nearly 200,000 travelers stayed in platform home sharing accommodation over the past year (2016-2017). This research seeks to explore more deeply the characteristics of these listings and the travelers consumption patterns through an open data analysis.

The Jeju provincial government has become the third province to endorse Airbnb in South Korea, after Gangwon Province and Chungnam Province. The national decision on platform home sharing, as well as Seoul's government, however, remains ambiguous and could have far-reaching implications for regional development once it has been made.

5. Literature End Note

Much research has already been done on the demand-side of Airbnb, qualitatively analyzing consumer motivation for using the platform and aspects of the rising demand for it (Guttentag, 2016; Roberts, 2016; Yuija, 2015; Finley, 2013), but little research has been done to quantitatively analyze the supply-side of the home-sharing unicorn. The supply-side of platform businesses is particularly relevant in understanding its impact. This difficulty is largely due to the difficulty of obtaining reliable and holistic data from Airbnb, which has chosen not to disclose its hosts'information and resists attempts to extract data from its servers. Therefore, there are few studies in the field of Hospitality and Tourism Management which have deeply analyzed the overall distribution and characteristics of Airbnb in a specifically bounded area (Zervas, 2016).

By simply using the company's website, one can glean only a limited scope of market penetration of Airbnb, and risks severely underestimating its impact and therefore implications for local policy makers and businesses to consider. The following section describes the data the research was able to scrape from the platform.

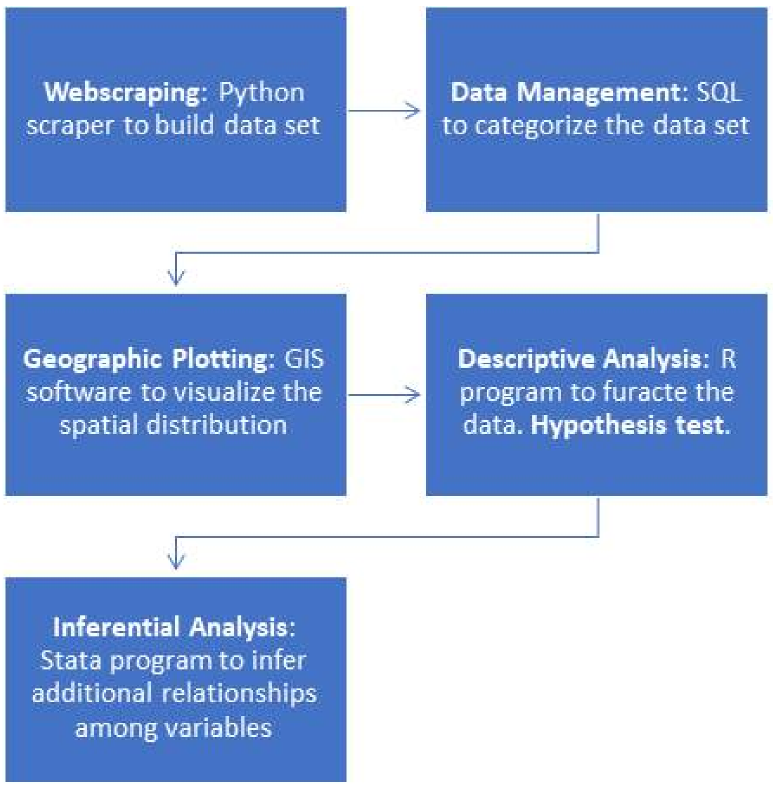
제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

Ⅲ. METHODOLOGY

1. Research Analysis

In order to understand the presence of platform home sharing on Jeju Island and the behavior exhibited by consumers of platform home sharing, this research applied an interdisciplinary approach of computer science, database management and economic theory to obtain and analyze data. The data was periodically collected over 7 months in order to establish price regularity. Specifically, this research sought to analyse which type of listing, be it a shared space with the host or an entirely separate dwelling, was most frequently purchased, thereby revealing the consumers' preference. The analysis of research is further illustrated in the following <Figure 3-1>.

<Figure 3-1> Research Analysis


Webscraping: Python scraper to build data set
Data Management: SQL to categorize the data set
Geographic Plotting: GIS software to visualize the spatial distribution
Descriptive Analysis: R program to furacte the data. Hypothesis test.
Inferential Analysis: Stata program to infer additional relationships among variables

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

2. Hypothesis

Based on the literature review, the following hypothesis is applied to platform home sharing on Jeju Island.

Platform home sharing consumers' preferences on Jeju Island will be revealed as an absolute majority of entire homes rather than shared spaces.

1) Research Model

The research hypothesis may be formally illustrated through a series of inequalities showing assumptions and statements:

if	$A > B > C$ (revealed preference)
then	$A > C$ (transitivity assumption)
Hypothesis	$B + C < A$ (absolute majority of 'A')

'A' represents entire home listings. 'B' represents private room listings. 'C' represents shared room spaces. Room listings are categorically ranked in the analysis in terms of number of reviews as an indicator for how often they are purchased. Therefore, if the sum number of reviews of private rooms and shared rooms is less than the aggregate number of reviews for entire homes, entire homes are categorically observed to have an absolute majority of reviews and thereby consumer preference for entire homes is revealed. Private rooms and shared rooms are both categorically shared spaces, because the guest stays with the host present in the dwelling.

This hypothesis is starkly juxtaposed with the philosophical rhetoric and general academic discourse of the sharing economy, which tend to emphasize the social and experiential aspects of exchange while deemphasizing the basic economic forces driving the phenomenon. The Theory of Revealed Preferences is employed to test this hypothesis, namely, if consumers exhibit the proclivity to purchase one bundle of goods over another, assuming the

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

affordability and availability of both bundles, they can be said to prefer the first and are more highly motivated by its value propositions. The consumer behavior to purchase entire homes over shared spaces would reveal that platform home sharing guests are motivated by notions of privacy, space and cost rather than sociocultural exchange and communal living. Moreover, given a ternary distribution of listing types, a particular category of listing possessing an absolute majority of preference rather than a simple majority, is a stronger indicator of consumer behavior.

3. Data

Consumer behavior is commonly measured by motivation-based surveys on a limited sample to make inferences of a population. This research proposes an alternative to measuring consumer behavior in regards to platform home sharing by examining consumer purchases, in alignment with the theory of revealed preferences.

In order to paint a clearer picture of the scope and implications of platform home sharing on Jeju Island from the supply-side in the local context, alternative methods of data collection from conventional surveys were employed. Web scraping is a technique by which a researcher can design a computer script to systematically contact a website to collect and store specific data. This is in order to establish an open primary data set from which further research may be done and policies may be formed.

For this research, Python 3.5 was used to design a tool that periodically scrape Airbnb's servers in order to obtain the necessary data for analysis. This included the overall satisfaction of the listing, the room type, the property type, the guest capacity of the listing, the number of reviews a listing has received, the host identification number attached to the listing, the room identification number associated with the listing, the precise geographic

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

location of the listing and the price per night of the listing. With this information trended over time, a foundation for further analysis can be established. The description of each data variable is detailed below.

1) Description of Data

(1) Overall Satisfaction

One of the metrics that is publically available and calculable from Airbnb's listings is the reported overall level of satisfaction guests who stay at a particular listing. A guest's level of satisfaction is determined by the guest after the stay experience and is factored into the previous guests'reported overall levels of satisfaction. The total satisfaction metric that the guest reports on is divided by Airbnb into six categories: accuracy, communication, cleanliness, location, check-in and value (Airbnb, 2017). Each of these categories is evaluated on a 5-point likert scale with 1 being the lowest and 5 being the highest level of satisfaction. The user who booked the accommodation is solely responsible for completing the satisfaction review, not each individual guest who stayed there. The "overall satisfaction"is calculated by averaging the six categories of satisfaction. For the purposes of this research, only listings with three or more reviews were considered for analysis in order to mitigate the risk of skewed results based on extraordinary circumstances.

Airbnb markets itself as a "trusted community marketplace"(Airbnb, 2017) whose mission is to match travelers with local hosts. Featuring the individual reviews as well as reciprocal levels of satisfaction shows much platform transparency and is crucial to how the system is based on mutual trust between strangers. Satisfaction levels, however, tend to be positively skewed when compared with traditional tourism accommodation providers' ratings. In arecent study of more than 600,000 listings, nearly 95% of all user-generated

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

reviews were higher than 4.5 out of 5 stars. Overall levels of satisfaction play an important role in ranking and user selection in the platform's ecosystem, so the positively skewed perceptions may feed into a higher mean.

(2) Room Type

The platform delineates its listings into three basic categories: shared rooms, private rooms and entire homes. The listing property is further divided into a number of types, which will be detailed below (Airbnb, 2017)

<Table 3-1> Room Types

Shared Room	Guests share the entire space with you or others and don't have a private room to themselves.
Private Room	Guests share some common areas with you, like the kitchen, living room, or bathroom, but they have their own private room for sleeping.
Entire Home	Guests rent the entire unit and don't have to share the space with you or with anyone else.

Shared rooms tend to result in a more intimate experience with the host and other travelers. They are often the least expensive listing types. Dormitory-style guest houses as well as floor sleeping arrangement in a studio apartment are common 'shared rooms'. Private rooms on the other hand are more suitable for couples and are the archetype of Airbnb's message. Some interaction with the host is expected, but privacy boundaries exist for both parties as well. Finally there is the possibility to rent the entire home. This listing type is generally most expensive and has the widest variety of property types, including some more eclectic listings such as igloos, treehouses, castles and private islands. For the most part, however, 'entire homes'feature whole apartments or single-family homes.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

(3) Property Type

In addition to the basic three room types, there are a plethora of property types to choose from. Property types are the particular variety of accommodation that is being listed for rent, including but not limited to: apartments, bed and breakfasts, boutiques, bungalows, cabins, chalets, condominiums, dormitories, guest suites, guest houses, hostels, houses, huts, in-laws, lofts, pensions, serviced apartments, tents, timeshares, townhouses, treehouses, vacation homes, villas and 'other'types. *At the time of writing, the express search filter and exhaustive list of available property types has been disabled by the platform; however, the data is still available through the API of the platform's server and the URL text of a particular listing. The aforementioned property types are limited to those which the author encountered in this research.

(4) Capacity

The carrying capacity of a listing is determined by the host. This figure represents how many guests may potentially stay in a given listing. Listings are not necessarily always filled to capacity, but are generally not over the stated capacity. This figure is therefore not synonymous with occupancy, but could be used to gauge the potential impact Airbnb listings may have on a locality. Shared rooms tend to have lower capacities in the range of one or two guests while entire properties may have up to 16 guests stay in them. If the 'extra fee per additional guest' feature is included in a listing, the number

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

of guests may potentially exceed the maximum capacity of the listing (Airbnb, 2016). This presents potential safety issues regarding occupancy as well as noise complaints for particularly large parties.

(5) Reviews

How many times a particular listing has been rented may be represented in the number of reviews it has. After a guest has completed his or her stay and completed the satisfaction survey, commented on the experience and the host has commented on the guest's behavior, the platform tags an additional 'review' to the listing's description. This is an important factor in how prevalent a particular listing appears while searching a locality for availabilities on the platform. A listing that has at least one review is much more likely to be rented than a listing that has none. Moreover, although many listings have zero reviews and are therefore excluded in several forms of analysis, they are not entirely disregarded because they show a more holistic market penetration of a locality even if they have not been rented.

(6) Host Identification

Every host on the platform has a unique identification number associated with their account. In order to be approved as a suitable host on the Airbnb platform, the host must take a photo of a government issued identification document (passport, driver's license etc.) as well as a picture of themselves. The self photo must then be matched with the government issued identification document photo for the host to be considered "verified"(Airbnb, 2017). Host identification is an important metric when discussing the legality of hosts listing multiple properties on the platform to establish unofficial accommodation businesses. Municipal governments in Scotland, Canada and

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

the United States have recently begun to limit the number of properties an individual may legally rent out without living in them (Hamada, 2017). One area of consideration, however, remains the host's ability to create multiple identities on the platform from which he or she manage multiple listings and not arouse suspicion from regulatory authorities. Nevertheless, host identification as a variable remains a unique way to monitor the prevalence of multiple property listers.

(7) Room Identification

Each individual listing on Airbnb has its own unique room identification number. Although a host identification number may have multiple room identification numbers associated with it, the room identification number is a singular occurrence. A common metric in estimated the total number of listings of a locality, the room identification number embedded in the URL (e.g. "https://www.airbnb.com/rooms/4789513") the numbers following 'rooms' constitute the room identification) allows researchers to monitor the fluctuation of the platform over a period of time. Room identification also ensures that a listing is not double counted in analytics. Other aspects of the listing may change such as the number of guests it can accommodate, its price and satisfaction, but the room identification for the property is a constant.

(8) Latitudinal and Longitudinal Coordinates

In addition to specific host and room identification numbers, this research was able to obtain the particular latitudinal and longitudinal coordinates for each listing. These coordinates place the research in real space and offer a wide range of practical implications. First, the researcher was able to plot the

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

coordinates into a geographic information system (GIS) interface to more easily represent and analyze the spatial distribution of the platform across a locality. This is especially pertinent to multi-layered GIS interfaces against which municipal jurisdictions can be conjoined with listings' locations. Perhaps even more useful, however, is the coordinates allow researchers to monitor the growth of the platform in particular locations. In order to protect the privacy of the host, the specific coordinates are within a hundred meters of the actual property's location. This margin of error is narrow enough for the purposes of this research, which is to measure the growth and prevalence of the platform on a more macro perspective.

(9) Price

Price is one of the most dynamic variables of the dataset. Prices are displayed in Korean won and are represented as the cost per night to rent the property. The price per night per listing may fluctuate if the host changes his or her price model, or if the host is using some automated revenue management service (Airdna, 2017). Additional fees such as cleaning, service and extra costs per additional guest are not reflected in the price figure scraped from the listing.

In comparison with traditional accommodation, Airbnb listings tend to be priced more competitively for a similar or even greater value proposition. This may be due to lower overhead costs and capital investments that Airbnb hosts need to expend in order to start their businesses. Moreover, Airbnb hosts are generally not subjected to the same scrutiny of permits, taxes and regulations that traditional accommodation providers are, and this may be reflected in the lower price per night.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

4. Data Collection

In order to get a more accurate picture of the scope and scale of Airbnb on Jeju, the data described above was collected each week. The schedule for data collection covering the whole of Jeju island was every six days, shifting back one day of the week in each successive survey.

Data collection schedule: n-1

For example, if a survey is run on Saturday, May 20th, the next survey was conducted on Friday, May 26th. This ensured an unbiased data collection, taking into account the data and prices available on any given day of the week. The web scraping technique utilized in this research works, in essence, by taking a snapshot of the targeted information identified by the API on a website and cataloguing it, iterating the process thousands of times. If the data were collected on the same day of the week, biased booking trends and prices will inevitably skew the results. This n - 1 method ensured a more random as well as thorough data sampling to account for any potential price elasticity of supply.

This data collection methodology is based off of a robust scraping technique designed by Dr. Tom Slee (Slee, 2016), which the researcher programmed specifically for Jeju Island. The accuracy in terms of total listings of such technique is consistently within 10 percent of the parametric data held by Airbnb (Slee, 2016; Airbnb, 2015). After collecting the data through the Python-designed scraper, and designating it to a local server, the data was queried and structured using PostgreSQL 9.6, a powerful object-relational database management server.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

1) Geographic Parameters

Jeju Island is an ideal area for this web scraping technique. Due to its insular nature, the geographic boundaries are clearly defined and therefore there is little ambiguity in delineating the search area. Located approximately 140 kilometers south from the Korean mainland, Jeju Island is easily bounded by four longitudinal and latitudinal points.

A Bounding Box of Jeju Island:
N 33°34′09″N 33°11′39″E 126°09′36″E 126°58′28“

In comparison to other locales, Jeju Island is well suited for this boundary box technique because its borders do not conjoin with neighboring cities and a simple rectangle can capture its entire market. Boundaries for cities are often uncertain, and without publically available GIS interfaces, determining the precise metropolitan area of a locale is difficult. The outlying Chuja islands, although technically part of the Jeju provincial area, are excluded from this boundary box due to the substantially increased area needed to be searched. Furthermore, at the time of writing there were no listings on the Chuja islands.

The web scraping technique employed in this research is first bound by a specific geographic area. Jeju was bounded by the parameters above through the following code.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

<Table 3-2> Bounding Box

```
UPDATE search_area
SET bb_n_lat = NN.NNN,
bb_s_lat = NN.NNN,
bb_e_lng = NN.NNN,
bb_w_lng = NN.NNN
WHERE search_area_id = NNN
```

The bounding box defines the search area for the script, which recursively searches the parameters by breaking the larger area down into smaller rectangles at different zoom levels searching for listings. Like many internet startups today, Airbnb relies on Google Maps' API to host its map features and run its searches. Therefore, a bounding box derived from Google Maps'interface sufficed for this research. The researcher programmed the zoom function to execute 8 times, narrowing rectangular area under examination to listings until additional new listings were not found. This insured that no listings in the entire bounding box were missed during a single scraping instance.

2) Scraping

The scraping script, henceforth known by its functional name "airbnb.py" was configured to recursively run within the specified geographic parameters. To facilitate the scraping function, several open source, prepackaged program modules and tools were imported into the configuration file, airbnb.config:

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

<Table 3-3> Imported Modules to the Script

Logging
Psycopg2 and Psycopg2.errorcodes
Os
Configparser
Sys
datetime

(1) Configuring the scraper

Each program's role in the scraping process will be briefly described.

Logging is a highly flexible and functional program whose purpose is to assist any Python module in logging application messages an integrating them with third-party modules and libraries (Python Documentation, 2017a). In airbnb.config, logging allowed error messages and key functions to be saved and recalled by the researcher. This was instrumental in developing a working application which interfaced with Airbnb's hidden API. Logging allows researchers to see the tracebacks and error messages when a portion of their script does not execute properly.

Psycopg2 is the second edition of the most popular PostgreSQL adapter for Python and allows efficient interoperability between Python modules and PostgreSQL. This is known as a wrapper and enables the researcher to write and read code in Python while accessing third-party databases (such as Airbnb and PostgreSQL) without necessarily knowing the specific language or terminology of the third-party script (Python Documentation, 2017b). Given the scope of the research and ability of the researcher, such wrappers were instrumental in facilitating timely data collection. Psycopg2.errorcodes allowed the researcher to further diagnose faulty code and parameters by referencing a compendium of error messages and tracing back the malfunction script.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

Importing osis a convenient way for programmers to access the operating system of the computer they are running a script on, and can designate or manipulate file paths where data can be stored (Python Documentation, 2017c). For simpler interoperability, this researcher chose to run all scripts on a Linux operating system, thereby forgoing the need to import additional wrappers. The purpose of os for this research was to change the file path directory in storing the scraped data directly onto the computer, rather than on cloud. Oswas instrumental in creating folders, and changing working directories when interacting with various Python modules. Additionally, os facilitated standard time operations so that the researcher could more effectively track intercode operations.

Configparser module was imported to apply the researcher's personal configuration settings (geographic parameters, price levels, zoom functions etc.) to the airbnb.py module.

Sys is short for "system" and is used to access stored variables and functions through command line arguments. It is a standard operating module in most scripts. Functionally this module was used to retrieve the scraped data from a stdout (standard out) into a comma-separated values, or 'csv' file for analysis. Additionally, this module allowed the airbnb.py module to make changes and deposit data in the PostgreSQL program.

Finally, datetime module was imported to interface with the logging module in order to chronicle the exact time and date the data was scraped, and when it was last updated.

In configuring the scraper, the researcher had to keep in mind the potential for Airbnb to perceive a distributed denial-of-service (DDoS) attack. This occurs when Airbnb's servers receive too many requests in rapid succession and the allocated bandwidth cannot accommodate such high traffic. This can cause Airbnb's servers to fail, which, among other things, would terminate the scraping session. The inherent nature of web scraping is such that it

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

copies webpage data far faster than a human can; however, Airbnb's listing pages are designed for human user interface and such rapid requests are easily differentiated from ordinary human queries. Therefore, in order to avoid an Internet Protocol (IP) address ban from Airbnb, and to not overload the Airbnb servers with requests, this researcher implemented a "sleep"function between requests, allowing the web scraper to pause for 2 seconds between each URL scrape. For further information regarding the configuration files, see the Configuration section in the appendix.

(2) Implementing the scraper

In addition to the aforementioned modules, the scraper itself, airbnb.py, imported: argparse, requests, lxml and webbrowser modules.

Argparse is a user-friendly module used to write command-line interfaces which are defined within the program itself (Python Documentation, 2017d). Since airbnb.py references many internally sourced arguments and code, argparse was instrumental in calling those self-defined arguments. For example, the config, survey and listing code are all self-coded arguments and not part of Python's distributed libraries. Argparse module generated error and help messages to assist in implementing the scraper.

Requests is a Python module colloquially known as "http for humans" (Python Documentation, 2017e). Hypertext Transfer Protocol (http) is an application protocol for distributed, collaborative and hypermedia information systems and is the foundation for the Worldwide Web (Fielding et al., 1999). Functionally, HTTP serves as a request-response protocol in the client-server communication exchange model. In this model, a client application sends a request to a server, which in turn sends a response in the form of HTML files among other content. HTTP is designed for work on behalf of a user to facilitate intermediary communication between client (that is web application) and server and is primarily machine-to-machine communication. The nature

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

of this research, however, required human-to-machine communication in order to effectively mine information from servers. Therefore requests was imported to allow simpler application protocol interface (API) communication between the researcher and Airbnb's servers, making the HTTP request more obvious to the researcher and decoding responses from the server in more intelligible language.

Requests is the most commonly used library, or a collection of pre-configured selection of routines, functions and operations that a program can use (Python Documentation, 2013), for communicating with servers and extracting information from websites, such as Airbnb URLs. Lxml was used in conjunction with requests to parse the HTML structure of any given listing's web page and extract the necessary data for this research.

Finally, the webbrowser module was imported to the airbnb.py script. Webbrowser provided a useful interface to display Web-based documents, such as the tabulated room listing information on Airbnb's individual listing pages. Individual listing's URLs were accessed through webbrowser and subsequently scrapped for information using lxml and requests.

(3) Adding a Survey and Collecting Room Data

Following the connection test, some internal functions were executed to add a new survey shell to the scraper to populate. This entailed adding a survey identification number, which followed consecutively from the previous survey. Structured Query Language (SQL) was instrumental in rendering the survey functions into intelligible matrices. In total, 30 surveys with the identical Jeju Island bounding box were run.

The airbnb.py's survey function is coded with the possibility of searching by neighborhood, zip code, city or bounding box; however, due to the geographic isolation of Jeju Island, the additional add survey functionality was

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

not necessary to implement. For further information regarding the multifunctionality of adding survey instances, see Appendix - Airbnb.py, code lines 135-281.

After establishing a survey for airbnb.py to populate, the scraper continued to harvest listing information from Airbnb. The module webbrowser opened a browser to a predefined URL containing thefirst listing within the search parameters. Airbnb URLs are generic in nature particular listing information is accessible by inputting the 7-digit 'room_id' after "rooms/" (e.g. https://www.airbnb.com/rooms/#######).

From the bounding box specified URL, which references a unique listing, airbnb_listing.py then collected the room pertinent data. A note of consideration is appropriate here. Airbnb_listing.py is able to take a snapshot of the information at a specific moment in time, and only represents the data as it was displayed at the moment it was scraped. The host may change the listing's price, availability or carrying capacity thereafter; however, over multiple surveys, according to the central limit theorem, the surveyresults for a particular listing should converge to an accurate representation of the current home sharing situation on the island. Each listing was scraped more than thirty times, so the sample size was ample enough even on the level of individual listings.

Among the scrapable elements on an individual listing's page, the following were designated to be stored in config.airbnb.py. The elements listed below were derived from a self-defined function "self.(element) = None", self referring to the data listed in the particular URL and the element to the API variable on the page as determined by the HTML of the page. If information matching the API variable was found, "None" was replaced with the appropriate float, integer or string.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

<Table 3-4> Scrapable Variables

Table of variables	
self.room_id = room_id	self.bathrooms = None
self.host_id = None	self.price = None
self.room_type = room_type	self.deleted = None
self.country = None	self.minstay = None
self.city = None	self.latitude = None
self.neighborhood = None	self.longitude = None
self.address = None	self.survey_id = survey_id
self.reviews = None	self.coworker_hosted = None
self.overall_satisfaction = None	self.extra_host_languages = None
self.accommodates = None	self.name = None
self.bedrooms = None	self.property_type = None

Due to the liberal nature of Web 2.0, platform-based ecosystems and user discretion, a host is entitled to post as little or as much information about his or her listing as seen fit. Therefore, not all the above listed variables were populated with data, and the research was limited to the least common denominator of information present across all listings, as referenced in the Description of Data section. If the threshold for values was not reached for a particular listing, it was discarded from consideration. See lines 92-96 in appendix airbnb_listing.py for further information regarding this culling process. In order to mitigate the risk of double counting listing information, should the survey ever fail to complete or was truncated short of completion, the each value was stored and checked with a boolean "True" if the listing is previously saved in the database or "False" is the listing value already existed. This rollback feature insured data collection integrity.

For each listing, the following information was stored in an SQL relational

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

database, referring to a particular room_id and their associated values: host_id, room_type, country, city, neighborhood, address, reviews, overall_satisfaction, accommodates, bedrooms, bathrooms, price, deleted, minstay, latitude, longitude, survey_id, coworker_hosted, extra_host_languages, name, property_type, currency, rate_type.

(4) Exporting Room Information

After an initial survey, the information was exported and stored in an SQL file. The data was stored on a private server database, under two layers of encryption to prevent external tampering. The scraped variables were accessed through an SQL schema, or collection of database objects associated with the database username. Based on the 'True' booleans from the requested data in a given listing, tables were generated, with additional columns added as necessary to accommodate multiple variables. For example, if the host claimed to speak additional languages other than Korean, the code <add column extra_host_languages character varying(255)> facilitated the creation of an additional column to display this.

In creating these tables for each survey, much attention was paid to appropriating the correct information into the designated column. An initial "check: room table has no room_id column" was performed to establish a unique room row. In standard cross tabulation form, the row information was designated to room instances, while the columns were present variables in a particular listing.

For further information regarding these processes, the reader may refer to the appendices located at the end of this document.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

5. Methods of Analysis

In order to analyze the data through descriptive statistics, this research employed the statistics program R version 3.4.2. Through R, the raw data was compiled and common statistics such as mean, median, standard deviations and percentages were derived and used in further analysis. This information may be found in the results section of this research. Furthermore, using Stata version 13, this research employed a wide range of inferential and regression analyses to derive meaningful results from the data. In order to accommodate the numerous variables and fit the data in a regression model, a stepwise regression method was selected.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

Ⅳ. ANALYSIS OF DATA

1. Descriptive Analysis

After appropriately populating the dataset obtained through recursive web scrapes, the following results were observed.

The total number platform home sharing listings amounted to 5,566 on Jeju Island. Of these, 2,981 had three reviews or more. The mere presence of 5,566 listings on Jeju may overestimate its presence on the island, therefore it is appropriate to keep in mind the listings which have generated income in addition to the total number of listings.

Of the total number of listings of the population, 2,825 were entire homes, 2,112 were private rooms within a property and 629 were listed as shared rooms. The abnormal distribution of reviews and their densities is shown on the opposite side <Figure 4-1>. This reflects the propensity for newer listings in a growing market and further highlights the phenomenal nature of platform home sharing.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

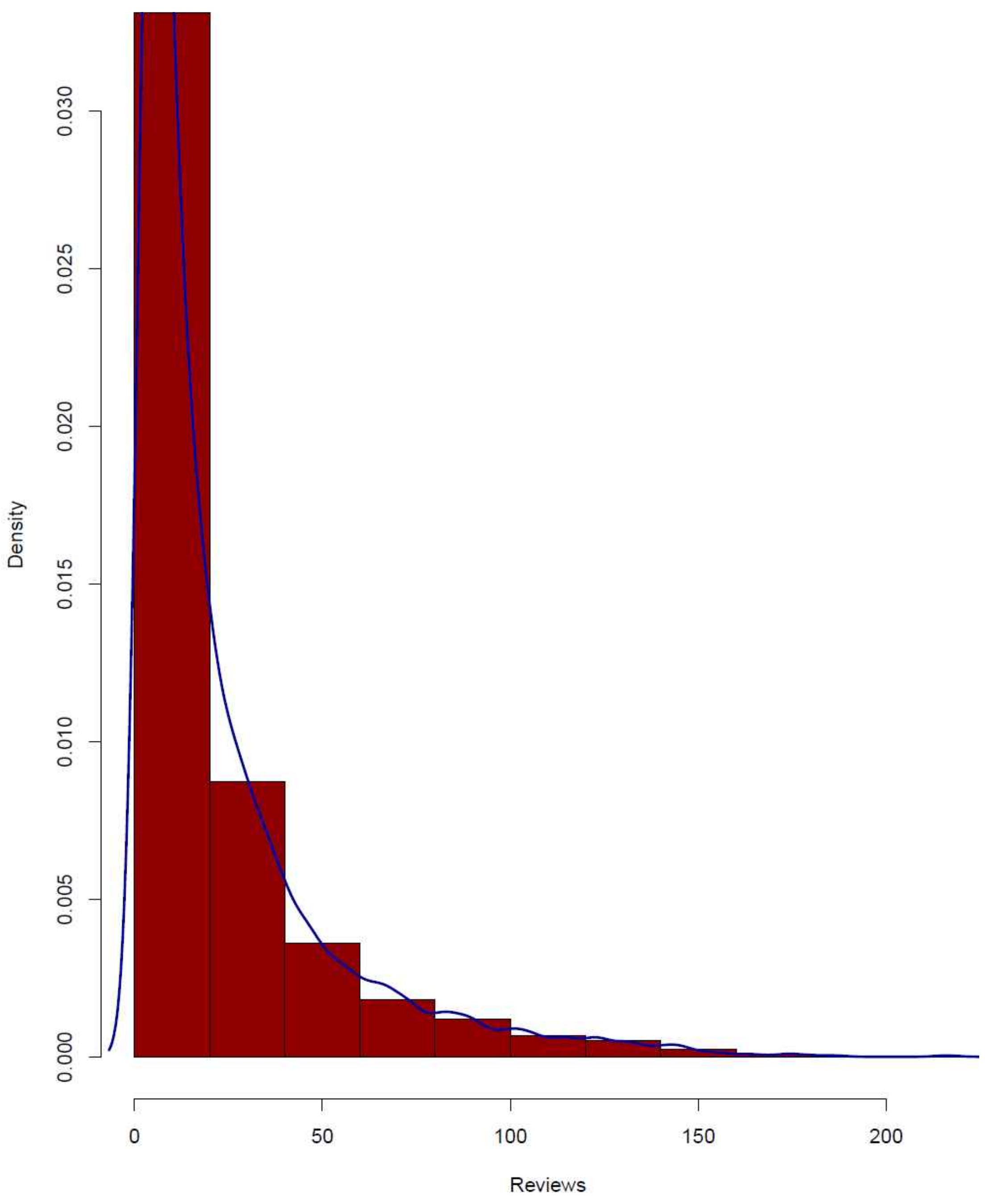


Figure 4-1 All Reviews' Density Distribution

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

<Table 4-1> All Listings

Entire population (5,568 listings)				
	Price (KRW)	No. of Reviews	Satisfaction (out of 5)	Accommodates
Min	10,500	0	0	1
1st Quarter	51,350	0	0	2
Median	78,190	3	4.0	4
3rd Quarter	123,700	13	5.0	6
Max	1,021,619	216	5.0	16
Mean	100,484.50	12.45	2.526	4.6

<Figure 4-2> Price Distribution of Entire Population on Opposite Side

The most popular property, an 'entire home' listing, had a total of 216 reviews and 50% of all the listings were priced between 10,500 and 78,190 KRW per night. The average price night was slightly higher than the median, at 100,484 KRW per night, revealing a right-tailed skew toward the higher end of the market and more luxurious accommodations. The highest priced property was listed as 1,021,691 KRW per night and accommodated 2 people, although the average number of accommodation was 4.6 guests. The satisfaction metric warrants some interpretation, given that mean and median are quite different. Overall satisfaction among the listings with fewer than three reviews is defaulted to zero (overall satisfaction is an average function of three reviews), and since nearly half of all the listings (2,587) have fewer than three reviews this skews the results to the left in the total population mean satisfaction of 2.5/5. This is starkly juxtaposed with the median of 4/5.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

<Table 4-2> Sample of Listings with Review Threshold

Listings with 3 or more reviews (2,981 listings)				
	Price (KRW)	No. of Reviews	Satisfaction (out of 5)	Accommodates
Min	11,714	3	0	1
1st Quarter	47,848	5	4.5	2
Median	72,356	12	5	4
3rd Quarter	117,870	29	5	6
Max	995,650	216	5	16
Mean	92,515	22.77	4.72	4.5

<Figure 4-3> Price Distribution of Listings with 3 or More Reviews on Opposite Side

The median number of reviews was three. This is a logical boundary to analyze the listings which have had an impactful presence on the island (Slee, 2017). Additionally, in order to calculate an overall satisfaction metric of a given listings, a minimum of three reviews was taken as a threshold to mitigate potentially biased reviews. Of the 2,981 listings which had three reviews or more 1,615 were listed as entire homes. Private rooms accounted for 1,096 listings and 270 shared rooms were listed with three or more reviews.

By eliminating 2,587 listings with under three reviews, one is able to see a clearer picture of the properties which are being booked on the island. The median and mean prices decrease slightly from the entire population, whereas the minimum price slightly increases. Unsurprisingly, the most dynamic statistic comparing the total population with the ‘three reviews and above’ listings is in regards to the reviews. In the latter, the mean number of reviews was 22.7 and in the former, this came to roughly half that, at 12.4 reviews. If one measures by median, the statistic in the latter is four times greater than the entire population.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

From the recursive web scraping, it was possible to determine not only the number and characteristics of the platform home sharing listings, but also how many of the hosts listed multiple properties. By subtracting the entire sample from the number of unique host instances, it can be seen that over half of all the listings belong to hosts with multiple listings (3,334).

<Table 4-3> Difference Between Population and Sample

Room type				
	Entire population (5,568)	Unique Hosts (2,234)	Listings with 3 or more reviews (2,981)	Total reviews (69,321)
Shared room	629 (11.31%)	299	270 (9.06%)	4,257
Private room	2112 (37.95%)	865	1,096 (36.76%)	23,635
Entire home	2825 (50.74%)	1,476	1,615 (54.18%)	41,429

<Figure 4-4> Distribution of Listings by Review Threshold on Opposite Side

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

1) Hypothesis test

The research hypothesis stated:

Platform home sharing consumers' preferences on Jeju Island will be revealed as an absolute majority of entire homes rather than shared spaces.

$$\begin{array}{lll} \text{if} & A > B > C & \text{(revealed preference)} \\ \text{then} & A > C & \text{(transitivity assumption)} \\ \text{Hypothesis} & B + C < A & \text{(absolute majority of 'A')} \end{array}$$

According to the data, entire homes categorically possess 60% (41,429 reviews) of the total number of reviews. This is greater than >50% threshold required to constitute an absolute majority in preference. Therefore, the hypothesis is adopted. Formally, this proof may be illustrated in terms of percentages of the total number of reviews by the following inequalities:

$$60\% \text{ (entire homes)} > 34\% \text{ (private rooms)} > 6\% \text{ (shared rooms)}$$
$$60\% \text{ (entire homes)} > 6\% \text{ (shared rooms)}$$
$$34\% \text{ (private rooms)} + 6\% \text{ (shared rooms)} < 60\% \text{ (entire homes)}$$

Platform home sharing consumers reveal their absolute majority preference for entire homes over shared dwellings. This preference is further accentuated by the observation that entire homes have the highest mean price value of 122,398 KRW per night <Table 4-6> compared with the mean values of private rooms and shared rooms, 64,723 KRW <Table 4-5> 26,529 KRW <Table 4-4> respectively. The price notwithstanding, platform home sharing consumers are more highly motivated by amenities of privacy and larger spaces as afforded by entire home listings.

From a population standpoint, half of all listings are entire homes, and if one assumes the three reviews and above threshold, the proportion of entire

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

homes increases by 4%. It is notable that although the overall proportion of the total listings do not significantly change by adopting the three reviews and more threshold, the relative number of shared room visits decreases.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

<Table 4-4> Distribution of Shared Rooms (Sample)

Shared rooms with 3 or more listings (270 listings)				
	Price (KRW)	No. of Reviews (4,062)	Satisfaction (out of 5)	Accommodates
Min	11,714	3	3	1
1st Quarter	19,035	4	4.5	4
Median	22,256	8	4.5	4
3rd Quarter	26,648	17	5	6
Max	174,532	144	5	16
Mean	26,590	15.04	4.656	5.5

<Figure 4-5> Distribution of Shared Rooms with 3 or More Reviews on Opposite Side

With 270 total listings having three reviews or more, shared rooms occupy less than 10% of the population. The mean and median prices for shared rooms are rather similar, revealing an even distribution across the mean. This notwithstanding, the max price for a shared room on Jeju is an outlier of 174,532 KRW. In the first quarter of the listings, the prices range from approximately 11,000 – 19,000 KRW per night for a shared room. This is makes sense when compared to private rooms, which are in a higher price bracket. Over half of the shared room listings are priced below 20,000 KRW, and 75% are priced under 25,000 KRW per night. Satisfaction as a metric is still positively skewed in this case. This is the most affordable listing type on average.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

<Table 4-5> Distribution of Private Rooms (Sample)

Private rooms with 3 or more listings (1096 listings)				
	Price (KRW)	No. of reviews (23,048 total)	Satisfaction (out of 5)	Accommodates
Min	15,171	3	3	1
1st Quarter	40,846	5	4.5	2
Median	57,184	12	4.75	2
3rd Quarter	80,525	26	5	4
Max	339,605	165	5	16
Mean	64,723	21.03	4.69	2.9

<Figure 4-6> Distribution of Private Rooms with 3 or More Reviews on Opposite Side

Private rooms are shared in the same dwelling as the host, however, as the name implies, they are separate sleeping chambers. Since they still entail interaction with the host, they are considered shared spaces for analytical purposes. In terms of number of reviews, the distribution density and volume is quite similar to that of the shared rooms. Some metrics do standout, nonetheless. In comparison with the shared rooms, the private rooms accommodate fewer guests and generally cost more. With an average of accommodating 2.6 fewer guests per listing than shared rooms, the private rooms seem to be targeting couples. Also, the median and the mean prices are within 10,000 KRW of each other, showing a relatively even distribution across the mean. Three quarters of the private rooms are priced under 80,000 KRW per night, and the mean is considerably less than shared rooms at 64,723 KRW per night. The satisfaction rating is ever problematic by showing a high propensity to positive skewedness.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

<Table 4-6> Distribution of Entire Homes (Sample)

Entire homes with 3 or more reviews (1615 listings)				
	Price (KRW)	No. of reviews (40,762 total)	Satisfaction (out of 5)	Accommodates
Min	15,171	3	0	1
1st Quarter	70,022	6	4.5	4
Median	102,698	13	5	4
3rd Quarter	154,619	33	5	6
Max	995,650	216	5	16
Mean	122,398	25.24	4.75	5.4

<Figure 4-7> Distribution of Entire Homes with 3 or More Reviews on Opposite Side

The entire home listings are the most prevalent and most highly visited on the island. In comparison to the shared spaces listings, the entire homes are highly differentiated. In terms of prices, about half of the listings with three reviews are over are under 100,000 KRW per night and about 75% are priced below 150,000 KRW per night. The price difference in the third quarter of the distribution is highly skewed due to the higher end properties which command a higher price. The most expensive listing on Jeju Island is an entire home listing, accommodating up to four guests and has four reviews, priced at 995,650 KRW per night. In terms of how many guests the entire home listings accommodate, the average of 5.4 guests per listing is slightly higher than the median of four guests. The number of listings notwithstanding, although they are more than the shared rooms and private rooms combined, the reviews metric is where the entire home listings stand out. With an average of 25.42 reviews and 40,762 total reviews, the entire home listings are clearly the most popular type of platform home sharing listing on Jeju Island. The satisfaction metric is also the highest when considering entire home listings. Although this metric is consistently inflated across all listings, in the entire home category it appears slightly higher at 4.75/5.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

<Table 4-7> Geographic Areas of Concentration

Concentration	Total Reviews	Entire homes	Private rooms	Shared rooms
1. Jeju City	8641	232	316	115
2. Hamdeok	3417	138	73	35
3. Bonggae	1558	73	58	7
4. Seongsan	10312	362	284	104
5. Pyoseon	5207	180	139	38
6. Seogwipo	8892	250	270	81
7. Jungmun	3379	166	104	20
8. GEC	1391	62	19	0
9. Sanbangsan	1618	48	54	7
10. Hallim	4968	303	135	38
11. Aewol	5486	254	206	27
12. Susan	1561	109	51	7

<Figure 4-8> Areas with high Concentrations of listings on Opposite Side

Finally, a brief analysis of the geographic areas of listings concentration yielded a few insights. Twelve geographic areas of high listing concentration were selected to determine which areas were most frequented and to give more consumer behavior data in more localized geographic contexts. The twelve areas do not cover the entire island, but were visually selected based on higher concentrations of platform home sharing listings.

The area surrounding Seongsan on the eastern coast is the most highly concentrated area, followed by Seogwipo and Jeju City. Inland the concentrated areas quickly disperse and the western part of the island remains relatively desolate of listings.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

2. Inferential Analysis

From the descriptive statistics above it is apparent that entire homes are the most popular type of platform home sharing listing on Jeju Island. In order to further understand the intricacies of the associated variables made available from the web scraping, linear and non-linear regression analytical techniques were applied to yield inferential statistical results. The results are entirely data driven, without subjective bias to reveal trends and interdependencies among the variables.

1) Stepwise Regression Analysis

First, in order to explore the relationships between property characteristics and the number of reviews the researcher started with a stepwise multiple regression of the number of the number of reviews on all other variables. The majority of them are statistically significant predictors of the number of reviews (p-value<0.1).

Among the various platform home sharing listings here were 31 different property types available on Jeju Island. Each of the property types was coded as a variable "prop_type#" and may be seen in the chart below.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

<Table 4-8> List of Property Types

Property Type Variable	Property Description
prop_type1	Apartment
prop_type2	Bed & Breakfast
prop_type3	Boutique Hotel
prop_type4	Bungalow
prop_type5	Cabin
prop_type6	Camper/RV
prop_type7	Castle
prop_type8	Chalet
prop_type9	Condominium
prop_type10	Dorm
prop_type11	Earth House
prop_type12	Guest Suite
prop_type13	Guesthouse
prop_type14	Hostel
prop_type15	House
prop_type16	Hut
prop_type17	Igloo
prop_type18	In-Law
prop_type19	Loft
prop_type20	Nature Lodge
prop_type21	Other
prop_type22	Pension
prop_type23	Pension (Korean)
prop_type24	Serviced Apartment
prop_type25	Tent
prop_type26	Timeshare
prop_type27	Townhouse
prop_type28	Treehouse
prop_type29	Vacation Home
prop_type30	Villa
prop_type31	Yurt

When it came to the host speaking additional languages after Korean, the following languages were associated with an increase in number of reviews: English, Russian, Indonesian and French. Apartment (property type 1), was used as a reference category and only house (property type 15) slightly outperformed it in number of reviews.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

<Figure 4-9> Stepwise Regression Output for Reviews

```
stepwise, pr(0.1) pe(0.05) : regress reviews accommodates price latitude longitude
en es zh ja ru id bn de fr hi it room_type2 room_type3 prop_type2 prop_type3
prop_type4 prop_type5 prop_type6 prop_type7 prop_type8 prop_type9 prop_type10
prop_type11 prop_type12 prop_type13 prop_type14 prop_type15 prop_type16
prop_type17 prop_type18 prop_type19 prop_type20 prop_type21 prop_type22
prop_type23 prop_type24 prop_type25 prop_type26 prop_type27 prop_type28
prop_type29 prop_type30 prop_type31
    Source       SS          df      MS              Number of obs       5,568
-------------+----------------------------------     F(26, 5541)         25.18
      Model  308090.526       26  11849.6356         Prob > F            0.0000
   Residual  2607277.49    5,541  470.542771         R-squared           0.1057
-------------+----------------------------------     Adj R-squared        0.1015
      Total  2915368.02    5,567  523.687447          Root MSE           21.692

------------------------------------------------------------------------------
    reviews |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
 prop_type19   4.423079  2.646537     1.67   0.095    -.7651723    9.61133
 price        -.0000479  4.06e-06   -11.82   0.000    -.0000559    -.00004
 latitude      7.624782  2.783987     2.74   0.006     2.167077   13.08249
 prop_type29  -8.977324  2.972862    -3.02   0.003     -14.8053   -3.149348
 en            9.308498    .68837    13.52   0.000     7.959023   10.65797
 es           -6.412797  3.234521    -1.98   0.047    -12.75373   -.0718683
 zh           -4.152603  1.129772    -3.68   0.000    -6.367399   -1.937806
 prop_type11 -13.2964    3.548487    -3.75   0.000    -20.25282   -6.339971
 ru           11.96615   3.934528     3.04   0.002     4.252932   19.67937
 id           19.24357   6.858835     2.81   0.005     5.797567   32.68958
 prop_type23  -7.386523  1.090559    -6.77   0.000    -9.524446   -5.248599
 de           15.35112   5.034286    -3.05   0.002    -25.22029   -5.481945
 fr            4.988075  2.946131     1.69   0.090    -.7874965  10.76365
 prop_type27  -9.118331  1.850278    -4.93   0.000     -12.7456   -5.491061
 prop_type18  -4.714467  2.585234    -1.82   0.068     -9.78254    .3536054
 room_type2   -5.587133   .7103761   -7.87   0.000    -6.979749   -4.194517
 room_type3  -11.33203   1.111872   -10.19   0.000    -13.51173   -9.152321
 prop_type13  -5.318245  1.223459    -4.35   0.000    -7.716704   -2.919787
 prop_type3   -6.701705  2.237382    -3.00   0.003    -11.08785   -2.315558
 prop_type14  -7.146288  2.364807    -3.02   0.003    -11.78224   -2.510338
 prop_type5  -10.67633   2.564267    -4.16   0.000     -15.7033   -5.64936
 prop_type12 -10.13792   4.665059    -2.17   0.030    -19.28327    -.9925775
 prop_type15   1.299005   .7761311    1.67    0.094    -.2225162    2.820527
 prop_type20 -12.62915   6.400683    -1.97   0.049      -25.177    -.0812999
 prop_type9   -6.006711  2.108586    -2.85   0.004    -10.14037   -1.873055
 prop_type30  -2.666075  1.541765    -1.73   0.084    -5.688538     .3563883
 cons       -234.7387   92.98033    -2.52    0.012    -417.0166  -52.46074
------------------------------------------------------------------------------
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

The first regression included all platform home sharing listings in the analysis. After the first regression, overall satisfaction as added as an additional predictor to control for the quality of guests' experience. The results marginally changed, and some of the negative coefficients in the first regression may be affected by the perceived quality of the guest's stay. However, due to the previously explained skewed nature of satisfaction as a metric, this may not be a reliable predictor in the regression.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

<Figure 4-10> Stepwise Regression Output for Reviews with Satisfaction>0

```
stepwise, pr(0.1) pe(0.05) : regress reviews overall_satisfaction accommodates price
latitude longitude en es zh ja ru id bn de fr hi it room_type2 room_type3 prop_type2
prop_type3 prop_type4 prop_type5 prop_type6 prop_type7 prop_type8 prop_type9
prop_type10 prop_type11 prop_type12 prop_type13 prop_type14 prop_type15 prop_type16
prop_type17 prop_type18 prop_type19 prop_type20 prop_type21 prop_type22 prop_type23
prop_type24 prop_type25 prop_type26 prop_type27 prop_type28 prop_type29 prop_type30
prop_type31 if overall_satisfaction>0
```

```
      Source |       SS         df       MS         Number of obs        2,980
-------------+----------------------------------   F(19, 2960)         20.07
    Model      254494.435       19  13394.4439      Prob > F             0.0000
    Residual  1975929.48     2,960   667.543742     R-squared            0.1141
-------------+----------------------------------   Adj R-squared        0.1084
    Total     2230423.91     2,979   748.715647     Root MSE            25.837

--------------------------------------------------------------------------------
           reviews      Coef.   Std. Err.     t    P>|t|     [95% Conf. Interval]
---------------------+----------------------------------------------------------
overall_satisfaction    9.105673   1.399659    6.51   0.000     6.361271    11.85008
prop_type3            -12.66133    3.496722   -3.62   0.000   -19.51758    -5.805077
price                   -.0000665  7.77e-06   -8.56   0.000     -.0000817  -.0000513
latitude                9.763724    4.55079    2.15   0.032      .840691  18.68676
prop_type14           -12.60063    3.947139   -3.19   0.001   -20.34005    -4.861217
en                      9.59419     1.046703    9.17   0.000     7.541851  11.64653
prop_type11           -22.78714     5.85636   -3.89   0.000    -34.27009  -11.30419
zh                     -7.270575    1.646025   -4.42  0.000    -10.49804   -4.043106
prop_type30            -4.521318    2.357268   -1.92   0.055    -9.143368    .1007319
ru                     11.5957       5.872827    1.97  0.048      .0804574 23.11093
prop_type13            -6.462819    2.113058    -3.06  0.002    -10.60603   -2.319607
prop_type27            -8.865737    3.613702    -2.45  0.014    -15.95136   -1.780115
prop_type5            -11.83393     4.542692    -2.61  0.009    -20.74109   -2.926779
prop_type21             8.411798    2.340519     3.59  0.000      3.822589  13.00101
prop_type23           -10.38028     1.79992     -5.77  0.000    -13.9095    -6.851058
prop_type29           -15.56624     5.98413     -2.60  0.009    -27.29972   -3.832761
room_type2             -7.053129    1.129625    -6.24  0.000     -9.26806   -4.838199
room_type3            -15.35555     1.95184     -7.87  0.000    -19.18265  -11.52845
prop_type9             -9.310061    3.339402    -2.79  0.005    -15.85785   -2.762277
cons                 -336.9983    152.3848      -2.21  0.027   -635.7892   -38.20741
--------------------------------------------------------------------------------
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

Some of the additional languages including Spanish, German and Chinese are seen as to have negative effects on the number of reviews. This may be due to visitor expectations of fluency from the host speaking in the purported language, and the host not being able to communicate effectively with the guest.

The second regression was designed to be more selective due to the abundance of listings with zero reviews (1,549 listings) and a metric of zero satisfaction (2,589 listings). Overall satisfaction is an average of three reviews' satisfaction. Therefore, the listings with zero overall satisfaction and fewer than three reviews were culled and the regression analysis was performed once again.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

<Figure 4-11> Stepwise Regression Output for 3+ Reviews

```
stepwise, pr(0.1) pe(0.05) : regress reviews overall_satisfaction accommodates price
latitude longitude en es zh ja ru id bn de fr hi it room_type2 room_type3 prop_type2
prop_type3 prop_type4 prop_type5 prop_type6 prop_type7 prop_type8 prop_type9
prop_type10 prop_type11 prop_type12 prop_type13 prop_type14 prop_type15 prop_type16
prop_type17 prop_type18 prop_type19 prop_type20 prop_type21 prop_type22 prop_type23
prop_type24 prop_type25 prop_type26 prop_type27 prop_type28 prop_type29 prop_type30
prop_type31 if overall_satisfaction>0&reviews>=3
```

```
     Source       SS          df       MS             Number of obs      2,980
------------+----------------------------------      F(19, 2960)        20.07
   Model      254494.435       19  13394.4439        Prob > F            0.0000
   Residual  1975929.48     2,960   667.543742       R-squared           0.1141
------------+----------------------------------      Adj R-squared       0.1084
   Total      2230423.91     2,979   748.715647       Root MSE           25.837

------------------------------------------------------------------------------
           reviews |     Coef.   Std. Err.     t    P>|t|    [95% Conf. Interval]
-------------------+----------------------------------------------------------
overall_satisfaction   9.105673   1.399659     6.51   0.000    6.361271   11.85008
prop_type3           -12.66133    3.496722    -3.62   0.000  -19.51758    -5.805077
price                  -.0000665  7.77e-06    -8.56   0.000   -.0000817    -.0000513
latitude               9.763724  4.55079      2.15   0.032     .840691   18.68676
prop_type14          -12.60063   3.947139    -3.19   0.001   -20.34005   -4.861217
en                     9.59419   1.046703     9.17  0.000     7.541851  11.64653
prop_type11          -22.78714   5.85636     -3.89   0.000   -34.27009  -11.30419
zh                    -7.270575  1.646025    -4.42   0.000   -10.49804   -4.043106
prop_type30           -4.521318  2.357268    -1.92   0.055    -9.143368     .1007319
ru                    11.5957    5.872827     1.97   0.048      .0804574  23.11093
prop_type13           -6.462819  2.113058    -3.06   0.002   -10.60603   -2.319607
prop_type27           -8.865737  3.613702    -2.45   0.014   -15.95136   -1.780115
prop_type5           -11.83393   4.542692    -2.61   0.009   -20.74109   -2.926779
prop_type21            8.411798  2.340519     3.59  0.000     3.822589  13.00101
prop_type23          -10.38028   1.79992     -5.77  0.000    -13.9095    -6.851058
prop_type29          -15.56624   5.98413     -2.60  0.009    -27.29972   -3.832761
room_type2            -7.053129  1.129625    -6.24  0.000     -9.26806   -4.838199
room_type3           -15.35555   1.95184     -7.87  0.000    -19.18265  -11.52845
prop_type9            -9.310061  3.339402    -2.79   0.005    -15.85785   -2.762277
cons                -336.9983   152.3848     -2.21   0.027    -635.7892  -38.20741
------------------------------------------------------------------------------
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

The results are substantially the same as the previous regression with the notable difference in satisfaction having a more positive coefficient effect on the number of reviews.

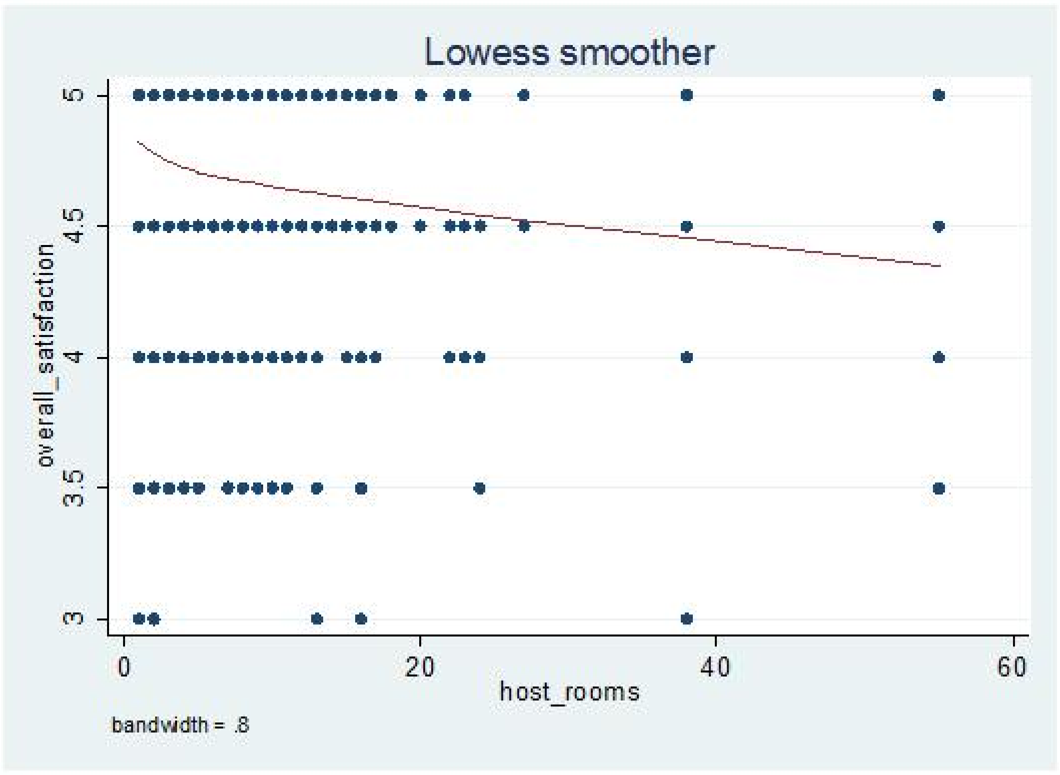
When considering all the variables in a large model such as the one shown above, the r-squared value, or goodness of fit, is quite low at approximately 11%. Nonetheless, it can be seen that hosts who speak Russian, English, are latitudinally positioned on the northern part of the island and have high levels of satisfaction from previous guests are predicted to have a higher number of reviews.

One of the concerns with platform home sharing from a policy standpoint is the growing tendency for single hosts to list multiple properties and act as small hotel chains. Therefore, this research tested to see if hosts with multiple listings had an effect on the number of reviews for each listing corresponding to their host_id. Two dummy variables were created indicating how many properties were offered by the host (host_rooms) and the corresponding binary indicator (host_mult=1 if host_rooms>1). After analysis, however, it was revealed to have no significant impact on the number of reviews when other variables were controlled for.

The same test was performed again testing for levels of satisfaction to see if hosts with multiple listings received lower levels satisfaction, operating under the assumption that a single host may not give as much attention to multiple listings.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

<Figure 4-12> Satisfaction and Host Rooms



With each additional property corresponding to the host, the overall satisfaction decreased by, on average, 0.01 points out of 5. The decrease in satisfaction is statistically significant at $p<0.01$.

<Figure 4-13> Satisfaction as a Function of Multi-Listers

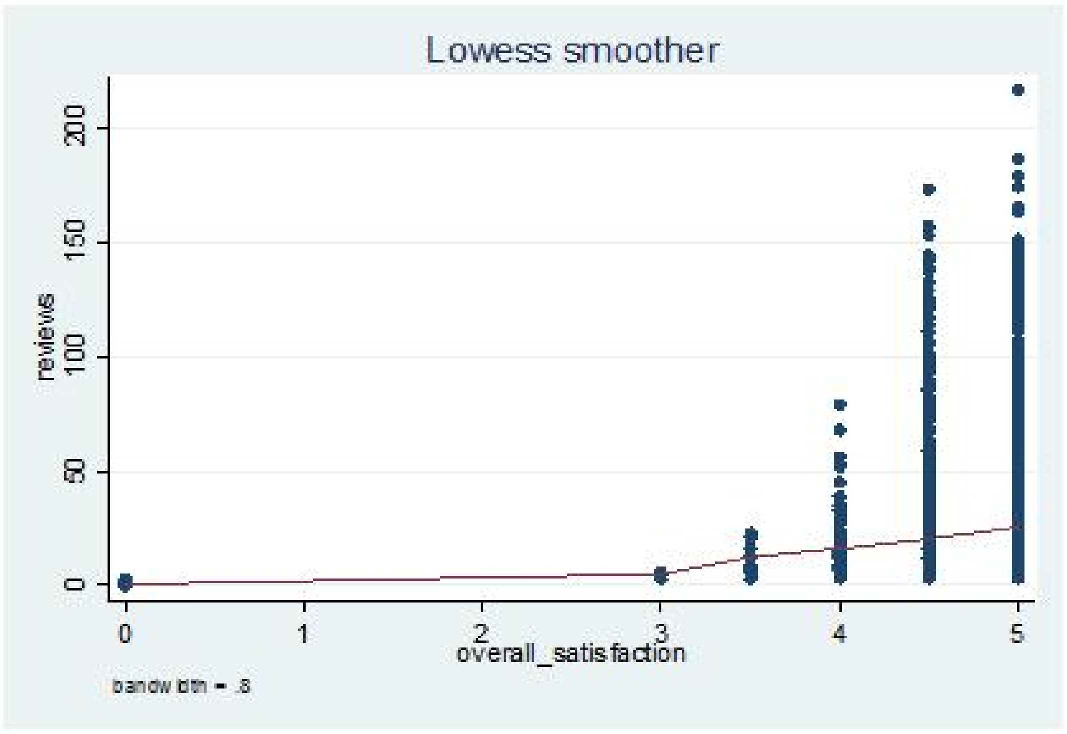
```
reg overall_satisfaction host_rooms if overall_satisfaction>0

      Source       SS           df       MS           Number of obs     2,980
-------------------------------------------------     F(1, 2978)       109.31
    Model       12.8128507        1   12.8128507      Prob > F          0.0000
    Residual   349.065924     2,978    .117214884     R-squared         0.0354
-------------------------------------------------     Adj R-squared      0.0351
    Total       361.878775     2,979    .121476595     Root MSE          0.34237

------------------------------------------------------------------------------
overall_sa~n      Coef.   Std. Err.      t      P>|t|     [95% Conf. Interval]
------------------------------------------------------------------------------
  host_rooms     -.0106755   .0010211  -10.46   0.000    -.0126776   -.0086734
  cons           4.777478    .008406   568.34   0.000     4.760996    4.79396
------------------------------------------------------------------------------
```

As shown in previously built regressions, lower levels of satisfaction are

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

associated with fewer reviews. This is further illustrated by the locally weighted regression plot below, supporting the hypothesis that multiple listers tend to receive lower levels of satisfaction from guests.

<Figure 4-14> Lower Satisfaction Associated with Fewer Reviews



2) CHIAD Regression Trees

Linear regression analysis has its limitations in that it is difficult to account for the potential nonlinearity of the geographic location's impact on the dependent variable. That is why regression trees were used as well. These account for the nonlinearity of the data, as well as the existence of certain interactions among variables that have a combined effect on the number of reviews. More specifically, they split explanatory variables automatically in such a way that provides the highest differentiation of the

제주대학교 중앙도서관 JEJU NATIONAL UNIVERSITY LIBRARY

number of reviews across the resulting segments. The results obtained from these regression trees generally agree with the results of the linear regression analyses, but also give some specific, actionable results based on the combinations of variables.

The CHIAD tree sets up a predictive analysis based on a criterion variable associated with the rest of the variables and configures the segmentation, or splitting, as a result from the dependency demonstrated from the most significant chi-squares (Perez, 2016). The number of resulting independent variables is a factor of how significant the Chi-square test is and can lead to slimmer or more “branchy” trees. The most significant factor is the always the first node of classification. A node (or branch) terminates when where is no significant relationship between the independent and dependent variables. Three CHIAD trees were constructed, each specializing on a different attribute of the listing data. Although each tree yielded similar conclusions, that entire homes in “house”-type properties which are moderately priced result in the highest number of reviews, a deeper analysis of each tree will articulate unique nuances apparent in the data.

In the first tree <Figure 4-14> the focus was given to exploring the effect that hosts who listed multiple properties had on the number of reviews. Although this variable was not the strongest determinant of number of reviews with all other variables considered, node 17 shows that hosts with fewer than 3.5 listings have the third highest predicted number of reviews of 27.2. The highest predicted number of reviews in this tree is observed when property type is “house”, room type is “entire home” and the price does not exceed 61,269 KRW per night (mean number of reviews=50.65). The lowest average number of reviews in this tree had a predicted number of reviews of 2.18. It was observed for objects of a type other than entire home/apt, where none of the extra languages are spoken by the host, where the price exceeds 81,943 and the property can accommodate more than three guests.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

<Figure 4-15> Property Type CHIAD Tree on opposite page

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

Additionally, a sensitivity analysis was performed to deduce what “other languages” means in context. First of all, it ought be noted that English is most prevalent language aside from Korean. Other languages are less important in terms of number of reviews, however, it can be seen from this tree that Japanese achieves the statistical threshold to increase the popularity of node 20 <Figure 14-5>. When considering Japanese into the analysis, the third highest predicted number of reviews of 29.6 is achieved (node 34). These are properties wherein the host speaks Japanese, the listing is categorized as a bed and breakfast, and the latitude is approximate with 33.5 degrees north. While the average number of reviews is 12.45, it is 18.75 when the host speaks extra languages (extra host lang: none=0) - 18.5, while only 9.35, when no other languages are spoken. This reveals a tendency for guests to favor hosts who speak additional languages, and may imply an internationally sourced customer base. In further analysis of this CHIAD tree, it may be seen that house-type properties which are the entire home and not shared spaces, with a price less than or equal to 61,268 KRW per night and where the host speaks multiple languages have the highest predicted number of reviews: 50.65. This is significantly cheaper than comparable value at a five-star hotel.

<Figure 4-16> Additional Languages Sensitivity Analysis on following page

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

Finally, an isolated geographic analysis, when only longitude, latitude and property type were included in the regression was conducted. As seen in the CHIAD tree <Figure 4-16>, comparatively longitude does not play as much of a role in predicting the number of reviews as latitude. The highest predicted value was centered around City Hall, or about 33.5 degrees north and 126.527 degrees east. Node 20 corresponds to this area with 29.4 predicted reviews on average. The lowest predicted number of reviews may be seen in node 25, with 2.75 reviews on average. These properties have a longitude of approximately 126.78 and are shared rooms in the Bonggae bounded area (see Table 4-7). For further information, <Figure 4-16> provides more detailed information.

<Figure 4-17> Geographic Position CHIAD Tree on following page

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

3) Summary of Results

The six main results from the regression analysis may be summarized in the following manner.

<Table 4-9> Summary of Inferential Results

1. Latitude is more important than longitude: the higher the latitude the larger number of reviews.
2. Knowing English and Russian are perceived to be assets.
3. Entire home type listings which are houses are the most popular type of accommodation with the highest number of reviews
4. Earth houses are the least popular types of property (lowest coefficient indicating fewer reviews compared with apartments)
5. Multiple listings ascribed to a single host does not significantly affect the listings' number of reviews, but it does affect the level of satisfaction.
6. Price and number of accommodates are negative factors.

First result: Within the predefined geographic bounding box encompassing Jeju Island, latitude plays a more significant role than longitude in the dispersion of platform home sharing listings. The majority of the listings are located in the northern part of the island; therefore, a higher latitudinal coordinate means more listings and by extension reviews.

Second result: English is the most commonly spoken international language. A possible explanation for Russian being useful, however, is that while most foreign tourists speak some English, Russian tourists generally prefer if the host speaks Russian and, therefore, it is useful to know some Russian to be able to host tourists from Russian speaking countries.

Third result: In accord with the descriptive statistics, listings which are entire homes and categorized as houses receive, on average, the highest number of reviews and therefore are considered the most popular. This supports the research hypothesis that platform home sharing guests are less motivated by the social and experiential aspects of the practice, and more motivated by the pragmatic and economic aspects of the accommodation offerings.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

Fourth result: Earth houses, or clay brick houses, on Jeju Island are comparatively less popular than conventional houses. Of the 42 “Earth Houses” listed on Jeju, 31 are entire homes and 11 are private rooms. Regardless, this shows consumer preference to choose houses above shared spaces.

Fifth result: The rising trend for hosts to use platform home sharing as a small business is an area of concern for policy makers. Although being a multiple lister does not seem to affect the number of reviews a listing receives, it does marginally decrease the overall satisfaction of each listing. A possible explanation may be that hosts managing multiple listings cannot appropriate sufficient time and energy to properly tend to all of them.

Sixth result: Listings which have higher levels of accommodation tend to have fewer reviews. Additionally, listings which are priced higher tend to have fewer reviews. Platform home sharing guests seem to not want to share a space with many others, and are motivated by lower prices.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

Ⅴ. CONCLUSION

1. Conclusion

1) Findings

Starting from a discussion on the emergence of platform home sharing on Jeju Island as a significant and new form of unregulated tourism accommodation, this research traced the literature review of the sharing economy at large and platform home sharing in particular. The lack of empirical studies in this field has resulted in an ambiguous discourse as to what motivates consumers and suppliers to participate in this new economy. While some researchers have posited that consumers are primarily motivated by experiential, social or even environmental notions, this research hypothesized that suppliers are primarily exhibiting rent-seeking behavior and consumers exhibit utility-maximizing behavior by preferring entire homes to any shared space. The test of such a fundamental hypothesis was made possible only by obtaining consumer purchase data and supply characteristics through a robust methodology of web scraping and data mining techniques. The data analysis revealed that the majority of platform home sharing hosts supply entire properties. The majority of consumers on Jeju Island likewise rent entire properties which are categorized as “houses”, are competitively priced below 144,000 KRW per night and are not shared spaces with the host. These properties afford the guest more space and amenities than a hotel at a lower price. The consumer behavior on Jeju Island reveals that this is the digital expansion of the market economy. Therefore, this facet of the sharing economy should be clearly distinguished from what is commonly

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

known as “sharing”.

The descriptive analysis proved the research hypothesis that entire homes would possess an absolute majority of preference in terms of the total number of reviews; formally, $B + C < A$. This reaffirms the tendency for platform home sharing consumers to maximize their utility through amenities such as privacy and space above social interaction and cost reduction.

The analysis further revealed that English and Russian languages were prime assets in describing a listing and Japanese was a secondary asset. Finally, the research showed that hosts who operate multiple listings have an aggregate lower level of satisfaction than hosts who have a single listing.

2) Discussion

The Sharing Economy has matured into a real source of income for many residents of Jeju Island through platform home sharing. On a deeper level, whether or not this new source of income is disrupting traditional accommodation providers is left to be seen and merits further discussion and research. This increasingly relevant sector of the economy must be reliably measured and enter policy makers’ decision frameworks in considering future tourism development on the island.

That the results show that platform home sharing consumers on Jeju Island are more likely to rent a listing that maximizes their utility at a lower price, rather than engaging in social interaction and forming communities is a novel discovery. This consumer behavior in line with the centuries old *homo economicus* paradigm portends utilitarian consumption over brand loyalty and exclusivity to a given platform company. The fact that the platform Airbnb provided this purchase opportunity should not be overly scrutinized, as consumers and suppliers can easily switch platforms should more favorable conditions present themselves. In an access economy, the consumers seem to

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

be less interested in trying to identify which brand or platform is more "them" and more interested in the best economic value. Most marketing practices are based on the linear supply chain model of building a community around a brand and capitalizing on consumers then being able to appreciate a shared identity with their peers with products and services that identify who they are. Platform home sharing consumers, however, appear to not be primarily seeking social value out of renting spaces from strangers. Framing the discussion in terms of community and social exchange may be misleading for policy makers and academics alike, because these are not the benefits that consumers expect to receive be participating in this economy.

Alternatively, the practice of platform home sharing in particular and the sharing economy in general could adopt different names which more accurately depict their character. The sharing economy is too often depicted as an altruistic endeavor, wherein the "sharing" aspect is emphasized and the "economy" aspect is laid to the side. The business models are entirely for-profit and the consumer behavior is consistent with *homo economicus*, therefore the compassionate and goodwill-natured semantics associated with "sharing" may have larger consequences in framing the issue and public perception. The term "platform economy" (JPMorgan and Chase, 2016) may be better suited to address the growing trend; it does not carry any sentiments with it nor skew consumers' perceptions of the businesses under the term.

(1) Satisfaction and Accommodation

On the subject of satisfaction, from the results reveal a universal positive skewedness across all listing types. This has been the case from earlier studies on platform home sharing reviews (Zervas, 2015; Overgoor, 2012), and the consensus is that the host as well as the guest benefit mutually from giving and receiving highly satisfactory reviews. The host receives a higher

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

ranking on the platform if he or she consistently receives high satisfaction, and the guests are accepted by more hosts if they have high reviews from previous stays. The vicious cycle of peer-to-peer inflated vetting has become more entrenched, so that on Airbnb every stay is 'above average'.

According to the mean level of guests each listing type can accommodate, it is worth highlighting that on average entire homes accommodate fewer guests than shared rooms. This may be due to the tendency for shared rooms listings to operate as dormitory guest houses and fit as many guests as possible.

The primary data made available from the web scraping methodology is limited, but one may also estimate revenue for an area that was generated through platform home sharing. Based on the number of reviews as a proxy for the relative number of visits, one can calculate how much a given listing has earned. While this is not exact enough for absolute values, it is useful for comparative analyses for whole cities or larger neighborhoods to determine which areas are earning more money from platform home sharing. If the ratio of reviews to visits is kept constant, this may be a valuable tool for policy makers to measure the efficacy of platform home sharing in geographically bounded areas.

(2) Multi-listers

After recognizing the profit seeking, economic utility maximizing characteristics of platform home sharing on Jeju, one of the concerns policy makers may have with it is the tendency for individual hosts to proactively rent out multiple properties, effectively becoming a small hotel chain. Although the data analysis revealed through linear regression that being a multi-lister does not affect the demand of a particular listing, the legal implications of running a potentially untaxed business are still present. One policy suggestion may be to limit the listing ration to one listing per host.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

As of October, 2017, approximately half of the listings on Jeju Island were allocated to multi-listers.

The theory of disruptive innovation needs to be updated to include forms platform disruption in a consistent manner. Furthermore, platform home sharing has not yet been proven through empirical studies to disrupt the incumbent hotel industry. These studies need to be performed in local contexts to determine if disruption occurs. Platforms derive their effectiveness and disruptive potential from the scale and scope of their customer networks; both producers and consumers and both must be considered in tandem when analyzing their propensity to disrupt. Too often disruptive innovation studies focus entirely on the demand and forget the supply.

2. Implications

This research contributes to the field of Tourism Management in a number of unique ways. First, in terms of theory, it grounds platform home sharing as an economic endeavor, and not social or experiential endeavor. This had not yet been definitively demonstrated due to the lack of user data and purchasing behavior. Second, it develops and introduces a unique way to collect data on an otherwise inaccessible sector of the tourism accommodation economy. Third, it provides not only an application of regression and classification tree analyses, but also further develops the theoretical discourse of such techniques in the Tourism and Hospitality sector. Fourth, it introduces an empirically, data-driven study based on theory of revealed preferences to analyze an increasingly relevant actor in Jeju Island's tourism sector. This research can contribute to the constructive policy development of Jeju's municipal government as it determines how best to address platform home sharing. It also provides hotel industry professionals market applications in how to better understand a potential source of local competition.

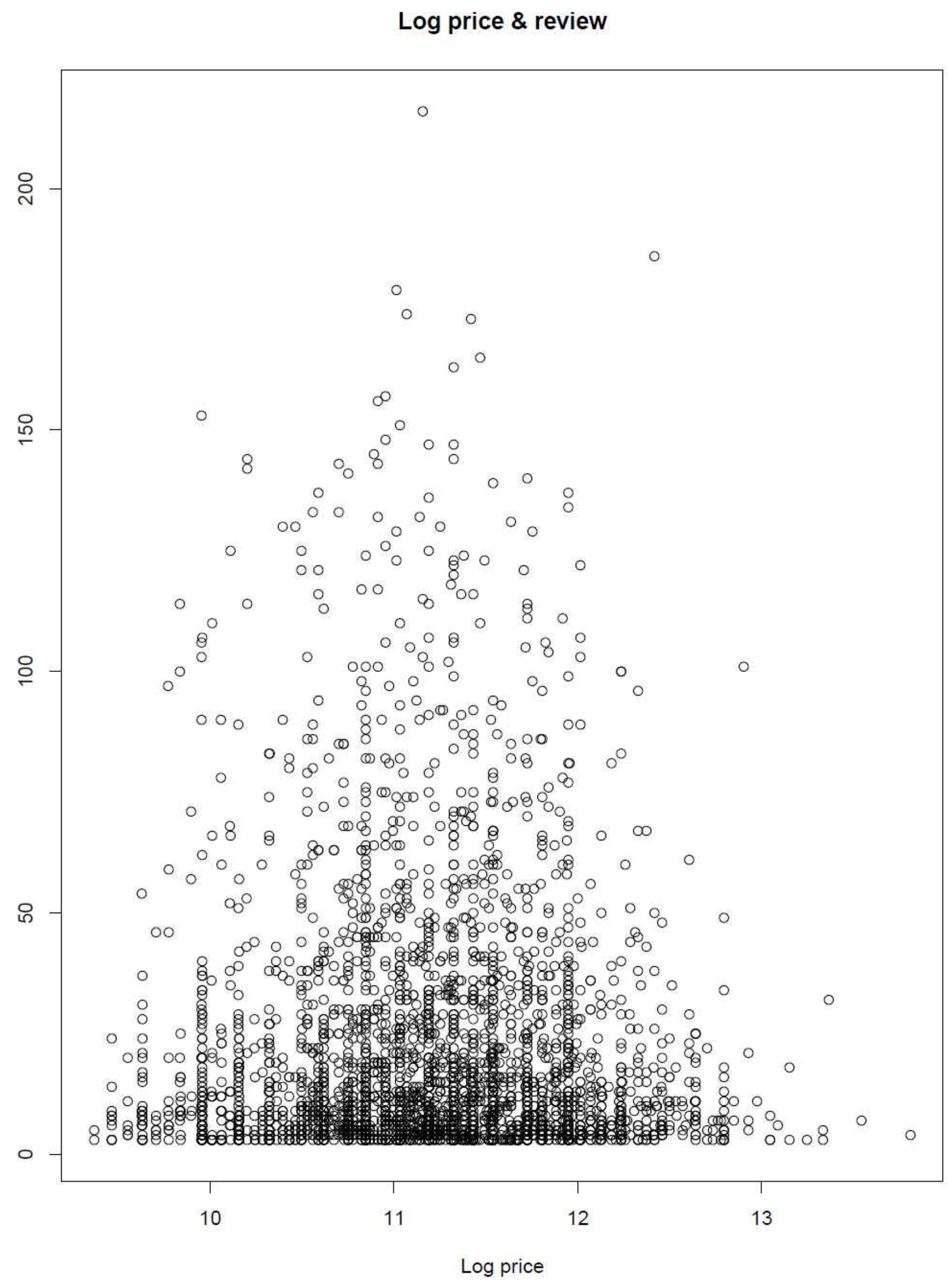
제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

Pragmatically, this research opens up new avenues of research and application for tourism practitioners, academics and policymakers to assess regulatory measures and further integrate community structures through open data. Now that researchers can conceptually and empirically assess the various impacts of platform home sharing in terms of market penetration and consumer behavior, it is possible to evaluate the responses to a variety of regulatory measures implemented by local, national, supra-national institutions as well as the platforms themselves. One key aspect of the current platform home sharing market to keep in mind is that the market is still growing and should continued to be monitored as it becomes more of a mainstay in tourism accommodation. As seen in <Figure 5-1> the price and relative number of reviews for platform home sharing on Jeju are still low, with prices converging around 60,000 KRW per night and very few reviews. As the market grows in scope and use, the data may segment into more discernably different groups.

Local economies are complex systems, made up of energy systems, food systems, sociocultural systems and housing systems among others. In a way, they are a system of systems. As the tourism continues to grow, the local systems more strongly affected by tourism business converge more closely in the local economy (Cohen et al., 2017). This has created challenges on the local government level when more and more citizens are engaging in entrepreneurial activities to diversify household income and leverage underutilized assets, such as spare rooms. Renowned urbanist, Richard Florida, has demonstrated that cities which manage to keep their residents by mitigating emigration are more likely to survive in the future than those which do not (Chesbrough, 2003). This entails encouraging an active technology scene, supporting tolerance for diversity and retaining entrepreneurial talent. The collaborative economic phenomenon embodied in platform home sharing on the local economic level is well-positioned to fulfill

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

these three ideals.



<Figure 5-1> Log Price-Review Distribution

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

In previous confrontations with Airbnb, lawmakers have tried to curtail the extent to which hosts may rent their properties, or place specific registration requirements on active hosts (Wachsmuth, 2017). These attempts, however, have been met by opposition from an energized home sharing community and ultimately failed. This is what has become known as Travis' Law, named after Travis Kalanick, the former CEO of Uber, which states that if a product appears so superior to the status quo in a society where the government must be responsive to the people, the people will defend and demand its right to exist (Isaacson, 2017). As of the time of writing, the local Jeju government is favorably disposed to cooperatively working with the platform to develop Jeju's tourism sector; however, additional transparency in the platform's dynamics and access to open data may change its perception by policy makers.

Hotel incumbents have largely ignored new market entrants when they are not perceived as a real threat to profits. Then, as the new entrant begins to expand its network of users, policy makers and incumbents tend to dismiss and make little of the entrant. Eventually, the actually scale and shifting paradigm in demand becomes apparent and rash measures are applied to curb its growth. However, once the network has reached a critical mass of users, it tends to succeed despite localized limitations.

3. Limitations and Future Research

All studies have their limitations, and this is no exception. The methodology developed in this research could have more meaningful impact if it were scaled to a larger geographic area and comparative analyses of Korean centers such as Busan and Seoul were carried out. Jeju Island was an ideal testing ground for this research given its geographic isolation,

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

tourism dependent economy and abundant supply of accommodation. The fact that platform home sharing has garnered such a reception on the island despite the abundant supply of traditional tourism accommodation serves as evidence for its consumer appeal.

In the future, Jeju policy makers should adopt a method to periodically collect and analyze data on the platform home sharing market. As the market continues to grow, the need to monitor the economic activity becomes more pertinent. This research proposes one methodology to accomplish this, but a more automated system could be developed with further research. Additionally, this methodology primarily focuses on the characteristics of supply, however, market segmentation studies of demand and demographic surveys could provide a deeper understanding of the forces at work in platform home sharing.

This research methodology could be applied in the following areas in further studies:

Measure the potential impact of platform home sharing on local hotels. This would require hotel occupancy and revenue data in located in areas which have particularly high concentrations of listings.

Determine the efficacy of platform home sharing in alleviating the strain on accommodation providers during large scale international events. This was the original purpose of Airbnb from its inception and would make an interesting study covering the period of the 2018 Olympic Games.

Analyze neighborhood gentrification effects of platform home sharing and increasing rents due to platform home sharing. Some of the undesired spillover effects of concentrated platform home sharing are potentially outpricing local residents and dividing neighborhoods into richer and poorer sides. Recent studies have revealed that a heavy platform home sharing in certain neighborhoods may lead to increased rent prices and gentrification (Lee, 2016; Wachsmuth, 2017). This research provides the first step in

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

approaching a deeper analysis on the subject.

Compare the growth of platform home sharing and characteristics of supply across other free tourism economic zones in South Korea such as Busan and Gangwon province. These areas have been granted relaxed regulations and provided legal status for home sharing (Yoon, 2016). As the Republic of Korea decides whether and how to share its market with newcomers from mobile or online platforms (Joongan Ilbo, 2016), this research would provide local administrative organizations more insight into how to make those decisions.

This is not an exhaustive list of possible continuing research, but rather applications of the data collection techniques developed in this study and implications for the relationships derived from them. Further relationships among the variables derived from this data collection methodology may also be tested through iterations of regression trees in continuing research.

As the presence of platform home sharing continues to grow on Jeju Island, and in the international tourism industry at large, Schopenhauer's famous words are worth remembering (Hassert, 1913):

"Ein jedes Problem durchläuft bis zu seiner Anerkennung drei Stufen: In der ersten erscheint es lächerlich, in der zweiten wird es bekämpft und in der dritten gilt es als selbstverständlich". – Arthur Schopenhauer

Government bureaus and hoteliers alike need to understand the scope and implications of a growing sharing economy. The platform wave can longer be considered a negligible phenomenon, its presence occupies an ever-greater portion commerce in local economies. It changes the way consumers make purchases, but not why they make them. Policy makers should be aware of the small business and individual entrepreneurial investments into this sector in order to mitigate ineffectual responses to its growth.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

REFERENCES

Adedoyin, F. (2017). Smart Tourism vis Digital Governance: A Case for Jeju Volcanic Island and Lava Tubes. *Journal of Tourism, Hospitality and Sports* 31.

Airbnb. (2017, October 21). What does the room type of a listing mean? Airbnb Help Articles. Retrieved from https://goo.gl/9r4cWx

Airbnb a. (2015, February 25). Paris Leads the World on Home Sharing. Airbnb Public Policy. Retrieved from https://goo.gl/jBu1MY

Airbnb a. (2016, April 21). Korean Government Partners with Airbnb to Supercharge Rural Economy. Airbnb Citizen. Retrieved from https://goo.gl/AMFfHc

Airbnb b. (2017, October 21). Why do I have to complete a verification process to host an experience? Airbnb Help Articles. Retrieved from https://goo.gl/z2sydw

Airbnb c. (2016, September 1). Airbnb Law Enforcement Transparency Report. Airbnb Citizen. Retrieved from https://goo.gl/eLLhzR

Airbnb d. (2017, October 21). How it works. Airbnb Help Articles. Retrieved from https://goo.gl/pZGerQ

Airbnb Citizen Korea. (2017). Airbnb Partners with Jeju Provincial Government Partners to Promote Tourism. *Airbnb Citizen*. Retrieved from https://goo.gl/u3Q8Ru

Airdna. (2017). Market Minder. Retrieved from https://goo.gl/Rd7JuJ

Armitage, C. & Conner, M. (2010). Efficacy of the Theory of Planned Behaviour: A Meta-Analytic Review. *British Journal of Social Psychology*, 40, 471 - 499. https://goo.gl/yki4UL

Azjen, I. (2005). Attitudes, Personality and Behavior. McGraw-Hill International.

Barnes, P. (2013, April 8). Rasmar Recognizes Jeju's Watery Riches. *The Jeju Weekly*. News and Travel.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

Beaudry, C. (2009). Who's right, Marshall or Jacobs? The localization versus urbanization debate. Research Policy, 38(2), 318 - 337.

Bellin, H. (2017). Some Managerial Thinking About the Sharing Economy. *Marketing Channel Insights* 24(1). 97-99.

Bensinger, G. (2017, March 9). Airbnb valued at $31 billion after new funding round. Wall Street Journal. Retrieved from https://goo.gl/FQ5wCK

Berger et al., (2017). Uber drivers of disruption. Oxford Martin School. January 2017. 1-11. Retrieved from https://goo.gl/j1ZBCF

Boncheck, M. & Choudray, S. P. (2013). Three elements of a successful platform. Harvard Business Review. Retrieved from https://goo.gl/UMWjfp

Botsman, R. (2017). Thinking. Retrieved from https://goo.gl/iVEckV

Botsman, R. (2013). The sharing economy lacks a shared definition. Fast Company. November 21, 2013. Retrieved from https://goo.gl/PSK1Mz

Bower, J.L. (1995). Disruptive Technologies: Catching the Wave. Harvard Business Review. January and February, 1995.

Bower, J.L. and Christensen, C.M. (1995, January-February). Disruptive technologies: Catching the wave. Harvard Business Review 73 (1), 51.

Cankurt, S. (2016). Proceedings from Intelligent Systems (IS): IEEE 8th International Conference. Retrieved from https://goo.gl/zbZDJu

Bray, J. (2007). Consumer behavior theory: Approaches and models. Bournemouth University Press. Poole, United Kingdom.

Carr, (2014). Airbnb unveils a major rebranding effort that paves the way for sharing more than homes. Fast Company. July 16, 2014. Retrieved from https://goo.gl/ydeqKd

Case, S. (2016). The third wave: An entrepreneur's vision of the future. New York: Simon and Schuster.

Chen, J. (2003). Market segmentation by tourists' sentiments. Annals of Tourism Research, 30(1), 178 - 193

Chesbrough, H. W. (2003). Open innovation: The new imperative for creating and

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

profiting from technology. Boston, Mass: Harvard Business School Press.

Choudary, S. (2016). Pipelines, Platforms, and the New Rules of Strategy. Harvard Business Review. April 2016.

Christensen, C. M. (1997). The Innovator's Dilemma. New York: Harper Business

Christensen. C. M. (2006). The ongoing process of building a theory of disruption. Journal of Product Innovation Management 23 (1), 39-55.

Christensen, C. M., McDonald, R., & Raynor, M. E. (2015). What is disruptive innovation? Harvard Business Review. December, 2015.

Chung, S.Y. Oh, S.S. Kim, S.Y. Han. (2004). Three representative market segmentation methodologies for hotel guest room customers. Tourism Management 25 (4) (2004), 429 - 444

Coase, R. H. (1937). The nature of the firm. Economica 4 (16), 386-405.

Codagnone, C. & Martens, B. (2016). Scoping the sharing economy: Origins, definitions, impact, and regulatory issues. Joint Research Centre of the European Commission Working Paper. Retrieved from https://goo.gl/o67ocK

Cohen et al. (2017). The City as a Lab: Open Innovation Meets the Collaborative Economy. California Management Review 59 (1) 5-13.

Crouch, G. (1992). Marketing international tourism to Australia: A regression analysis. Tourism Management 13(2), 196-208.

Dahlander, L. Gann, D. (2010). How open is innovation? Research Policy 39(6), 699-709

Das, S. (2017, February 12). The sharing economy creates a Dickensian world for workers—it masks a dark problem in the labour market. Independent. Retrieved from https://goo.gl/WUSu5w

do Valle, P. Pintassilgo, A. Matias, F. André. (2012). Tourist attitude towards an accommodation tax earmarket for environmental protection: A survey in Algarve. Tourism Management, 33 (2012), 1408 - 1416

Doh. (2015, October). From the editor: Why we need phenomenon-based research in international business. Journal of World Business 50, 609-611.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

Deleneuville, M. (2016). Le “smart tourism,” une aubaine pour les start-up de la smart city. Journal du Net. Retrieved from https://goo.gl/Sh7kfF

Deleneuville, M. (2017). Open data: les transporteurs craigngent pout leurs secret industriels. Journal du Net. April 3, 2017. Retrieved from https://goo.gl/rg7yyh

Eckhardt, G. M. & Bardhi, F. (2015, January 28). The sharing economy isn’t about sharing at all. The Harvard Business Review. Retrieved from https://goo.gl/sUfgVv

The Economist. (2017, May 27). Among private tech firms, Airbnb has pursued a distinct strategy. Retrieved from https://goo.gl/uCjN2y

The Economist. (2011, June 30). Aiming high. Retrieved from https://goo.gl/xeqMh6

Elder, D. (2017, October 18). Population of Jeju-si expected to surpass half a million next year. The Jeju Weekly. Retrieved from https://goo.gl/AC7wZS

Fehr, E. & Gächter, S. (2000). Fairness and retaliation: The economics of reciprocity. Journal of Economic Perspectives 14 (3), 158-181.

Fielding, R. T., Gettys, J., Mogul, Jeffrey C.; Nielsen, Henrik Frystyk; Masinter, Larry; Leach, Paul J.; Berners-Lee, Tim (June 1999). Hypertext Transfer Protocol – HTTP/1.1. IETF. The Internet Society. RFC 2616.

Foroohar, R. (2016). How the gig economy could save capitalism. Time. Retrieved from https://goo.gl/6gjrxS

Frenken, K. (2017). Putting the sharing economy into perspective. Environmental Innovation and Societal Transitions 23. 3-10

Fuchs C., Hofkirchner W., Schafranek M., Raffl C., Sandoval M., & Bichler R. (2010). Theoretical foundations of the web: Cognition, communication, and co-operation. Towards an understanding of web 1.0, 2.0, 3.0. Future Internet 2 (1), 41-59.

Furchtgott-Roth, H. (2016). The myth of ‘sharing’ in a sharing economy. Forbes. Retrieved from https://goo.gl/umEcfa

Finley, K. (2013). Trust in the sharing economy: An exploratory study. University of Warwick Press. Master’s Dissertation. Retrieved from https://goo.gl/WZwPkp

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

Freitag, J. 2017. Airbnb & Hotel Performance: Ana analysis of proprietary data in 13 global markets. Smith Travel Research. https://goo.gl/fEXyFj

Furr, N. and Zhu, F. (2016, April). Products to platforms making the leap. Harvard Business Review. Retrieved from https://goo.gl/1Dg31a

Gansky, L. (2010). The Mesh: Why the future is sharing. New York: Penguin Group.

Gerwe, O. & Silva, R. (2016). Business model innovation and substitution: Effect of Airbnb entry on the hotel industry. Working paper. Retrieved from https://goo.gl/N6WUMV

Granovetter, M. S. (1973, May). The strength of weak ties. American Journal of Sociology 78 (6), 1360-1380.

Gravari-Barbas, M. & Guinand, S. (2017). Tourism and gentrification in contemporary metropolises: International perspectives. Oxford: Routledge.

Gretzel, U., Koo, C., Sigala, M., & Xiangm C. (2015). Smart tourism: Foundations and developments. Electronic Markets 25.

Guttentag, D. (2016). Why tourists choose Airbnb: A motivation-based segmentation study underpinned by innovation concepts (Doctoral dissertation). UWSpace Waterloo's Institutional Repository. Retrieved from https://goo.gl/DdD1uX

Ikkala, T. & Lampinen, A. (2015). Monetizing network hospitality: hospitality and sociability in the context of Airbnb. Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing, ACM, Vancouver, 1033-44.

Isaacson, W. (2017, June 19). How Uber and Airbnb became poster children for the disruption economy. New York Times. Retrieved from https://goo.gl/89SvB2

Hamada, R. (2017, August 17). Concern as Airbnb properties “snowball” across Scotland. The Ferret. Retrieved from https://goo.gl/qc1SaL

Hambrick, D. (2007). Upper Echelons Theory: an update. The Academy of Management Review 32(2). 334-343.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

Hassert, K. (1913). Allgemeine Verkehrsgeographie. Berlin

Helft, M. (2017, September 21). How IoT is making workplaces more efficient. Forbes. Retrieved from https://goo.gl/kCr9Ms

Heinrichs, H. (2013). Sharing Economy: A potential new pathway to sustainability. GAIA 22(4). 228-231

Höjer, M., & Wangel, J. (2015). Smart Sustainable Cities: Definition and Challenges. In L. M. Hilty & B. Aebischer (Eds.), ICT Innovations for Sustainability, Advances in Intelligent Systems and Computing 333 - 344 New York: Springer.

Hwang, J., Park, H. Y., & Hunter,W. C. (2015). Constructivism in smart tourism research: Seoul destination image. Asia Pacific Journal of Information Systems 25 (1), 163 - 178.

Moazed, A. Johnson, D. (2016, February 27). Why Clayton Christensen is wrong about Uber and disruptive innovation. Retrieved from https://goo.gl/3eFSQU

Moazed, A. (2016). *Modern Monopolies: What it takes to dominate the 21st Century Economy*. St. Martin's Press. New York.

Joongang Ilbo. (2016, February 19). We need the sharing economy. Korea JoongAng Daily. Retrieved from https://goo.gl/PavAjm

Kahle L. R. & Close, A. (2006). Consumer behavior knowledge for effective sports and event marketing. New York: Taylor & Francis.

Kessler, Andy. (2014, January 14). Brian Chesky: The Sharing Economy and its enemies. The Wall Street Journal.

JPMorgan and Chase. (2016). The online platform economy: Has growth peaked? Retrieved from https://goo.gl/DtTwx6

Kass, G. (1980). An Exploratory Technique for Investigating Large Quantities of Categorical Data. Journal of the Royal Statistical Society. Series C (Applied Statistics). Vol. 29, No. 2 (1980), 119-127

Kim, D.J. Timothy, J. Hwang (2011). Understanding Japanese tourists' shopping preferences using the decision tree analysis method. Tourism Management, 32 (3)

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

(2011), 544 - 554

Kim, H. (2015). Percieved Destination Personality Based on Visitors' Experience: A Case Study of Jeju Island. *Tourism Travel and Research Association: Advancing Tourism Research Globally*. 2015 TTRA International Conference.

King, A. A. & Baartartogtokh, B. (2015, September 15). How useful is the theory of disruptive innovation? MIT Sloan Management Review. Retrieved from https://goo.gl/qC6TPy

Kokalitcheva, K. (2016, August 6). How Airbnb is preparing for the Rio Olympics. Fortune. Retrieved from https://goo.gl/QXNbxE

Kuhn, T. S. (1996). The structure of scientific revolutions. Chicago, IL: University of Chicago Press.

Lampinen, A. (2016). Why we need to examine multiple social network sites. Communication and the Public 1 (4). 489-493

Lee, D. (2016). How Airbnb short-term rentals exacerbate Los Angeles's affordable housing crisis: Analysis and policy recommendations. Harvard Law and Policy Review 229.

Leathwick, J. R., Lehmann, A., & McCoverton, J. (2002). GRASP: Generalized regression analysis and spatial prediction. Ecological Modelling 157 (2-3), 189-207.

Lepore, J. (2014, June 23). The disruption machine: What the gospel of innovation gets wrong. New Yorker. Retrieved from https://goo.gl/N3dxkG

Saunders, A. (2014, September 29). Is disruptive innovation dead? Management Today. Retrieved from https://goo.gl/fVm1Ha

Loudon, D. L., et al., (1993). Consumer behavior concepts and applications. 4th ed.: McGraw Hill.

Loh, W. Y. (2011). Classification and regression trees. WIREs: Data Mining and Knowledge Discovery 1, 14-23.

Loh, W. Y. (2008). Classification and regression tree models. In Ruggeri, Kennet, and Faltin (Eds), Encyclopedia of Statistics in Quality and Reliability (315-323).

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

New Jersey: Wiley.

McCarty, J; Hastak, M. (2007). Segmentation approaches in data-mining: A comparison of RFM, CHAID, and logistic regression. Journal of Business Research, 60 (2007), 656 - 662

McRae, H. (2016, September 14). A lack of jobs isn't the issue in the UK – it's the quality of them. The Independent. Retrieved from https://goo.gl/yZqKDM

Mill, J. S. (1836, October). On the definition of political economy, and on the method of investigation proper to it. Library of Economics and Liberty. England

Mill, J.S. (1874). Essays on Some Unsettled Questions of Political Economy, 2nd ed. London: Longmans, Green, Reader & Dyer.

Oksam, J. (2015). Airbnb: The future of networked hospitality businesses. Journal of Tourism Futures 2 (1), 22-42.

O'Reilly, T. (2005). What is Web 2.0. In H. M. Donelan, K. Kear, & M. Ramage (Eds.). Online communication and collaboration: A reader. Routledge. New York.

Ostrom, E. (2000). Collective action and the evolution of social norms. Journal of Economic Perspectives 14 (3), 137-158.

Overgoor, J., Potts, C., & Wulczyn, E. (2012). Trust propagation with mixed-effects models. Stanford University Working Paper. Retrieved from https://goo.gl/1UwgYG

Peek, G. (2014). City in Transition: Urban Open Innovation Environments as a Radical Innovation. Rotterdam University Press. Retrieved from https://goo.gl/1UwgYG

Perez, F; Bethencourt-Cejas, M. (2016). CHAID algorithm as an appropriate analytical method for tourism market segmentation. Journal of Destination and Marketing Management Volume 5(3), 275-282.

Perez, 2005. F.M. Díaz-Pérez, M. Bethencourt-Cejas, J.A. Álvarez-González. The segmentation of Canary island tourism markets by expenditure: Implication for tourism policy Tourism Management, 26 (6) (2005), 961 - 964

Persky, J. (1995). The ethology of Homo economicus. The Journal of Economic

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

Perspectives 9 (2), 221-231.

Python Documentation. (2013). Retrieved from https://goo.gl/VfRaUG

Python Documentation. (2017a). Retrieved from https://goo.gl/i3Ef5P

Python Documentation. (2017b). Retrieved from https://goo.gl/7vswLM

Python Documentation. (2017c). Retrieved from https://goo.gl/K2oR1d

Python Documentation. (2017d). Retrieved from https://goo.gl/DkdK1g

Python Documentation. (2017e). Retrieved from https://goo.gl/QiMQnr

Raynor, M. (2013). Three rules for making a company truly great. Harvard Business Review. April, 2013.

Reich, R. (2015, February 2). The share-the-scraps economy. Retrieved from https://goo.gl/pwUceT

Rick, T. (2013, July 26). Will collaborative consumption disrupt traditional business models. Retrieved from https://goo.gl/95oPzT

Roucha, R. (2015). On the ethics of web scraping. Retrieved from https://goo.gl/7auibd

Vazquez Sampere, J. P. (2016, April 8). Why platform disruption is so much bigger than product disruption. Harvard Business Review. Retrieved from https://goo.gl/mYxLpb

Sameulson, P. (1938). A note on the pure theory of consumer's behavior. Economica 5 (17), 61-71.

Satell, G. (2017, June 21). The 4 types of innovation and the problems they solve. Harvard Business Review. Retrieved from https://goo.gl/r2hC2g

Satell, G. (2013, June 26). 4 ways open innovation can drive your business forward. Digital Toronto. Retrieved from https://goo.gl/b6Z8HG

Shaughnessy, H. (2016). Platform wave disruption: A new theory of disruption and the eclipse of American power. New York.

Shead, S. (2016). Airbnb is looking at ways of increasing the amount of interaction between hosts and guests. Business Insider. Retrieved from https://goo.gl/os8BcH

Slee, T. (2016). Airbnb data collection methodology and accuracy. Retrieved from

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

https://goo.gl/s3wpZ9

Slee, T. (2017). What's Yours is Mine. OR Publications. Ontario.

Smith, A. (1776). The wealth of nations, Books I-III. New York: Penguin Classics, 1986, page 119

Southcott, D. (2015, September 10). Smart Support for Jeju Entrepreneurs. *Jeju Weekly*. Community

Staab, S., & Werthner, H. (2002, November-December). Intelligent systems for tourism. IEEE Intelligent Systems, 53 – 55.

Stors, N. (2016). Motives for using Airbnb in metropolitan tourism – why do people sleep in the bed of a stranger?. Regions – The voice of the membership. 299. 17-19.

Sundararajan, A. (2003). Network Effects. New York University Publications. Retrieved from https://goo.gl/FB2LQt

Sundararajan, Arun. (2016). The Sharing Economy: The End of Employment and the Rise of Crowd-Based Capitalism. Boston.

Sunil, J., & Noah, Z. (2015). Policymaking for the Sharing Economy: Beyond Whack-A-Mole. Toronto: Mowat Centre, University of Toronto. Retrieved from: https://goo.gl/G4M9sj (20-5-2015)

Sutton, C. (2004). Classification and Regression Trees, Bagging, and Boosting. Handbook of Statistics 24. 303-329.

Ting, D. (2017). Retrieved from https://goo.gl/gc5zPB

Ting, D. (2016). Retrieved from https://goo.gl/WUajm7

Tsang, A. (2017). Retrieved from https://goo.gl/wwgEWN

Tsotsis, A. (2011). Retrieved from https://goo.gl/8UAg4T

Tuttle, B. (2015, July 9). Marriott's CEO just made a pretty good sales pitch for Airbnb? Money.com, Retrieved from https://goo.gl/WcZP2U

Tyagi, C. and Kumar, A. (2004). Consumer behavior. Atlantic Publishers, US

UNWTO (2015). Understanding Tourism: Basic Glossary. Accessed online (May 25, 2015) at https://goo.gl/cCiL88

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

Roberts, H. (2016, November 11). Analysts say that Airbnb is hurting hotels more than predicted. *Business Insider Deutschland*. Retrieved from https://goo.gl/SL75DC

Rothkopf, E. (2014). CCTP-725: remix and dialogic culture. *Georgetown University Press*. Retrieved from https://goo.gl/F28D2Q

Varian, H. (2012). Revealed preference and its applications. *The Economic Journal* 122(560). p 332-338

Wachsmuth, D. (2017). Airbnb and the Rent Gap: gentrification through the sharing economy. *McGill University Press*. Retrieved from https://goo.gl/RUx6g7

Werthner and Ricci. (2004). E-Commerce and Tourism. *Communications of the ACM* 47(12). p 101-105.

West, J. Gallagher, S. (2006). Challenges of open innovation: the paradox of firm investment and open source software. R&D Management 36(3). 319-331.

Whiteman, M. (2014, June 30). The Sharing Economy Isn't Really a Disruption at All. *Aspen Institute College*. Retrieved from https://goo.gl/LrddoM

Williamson, O. (1979). Transaction-Cost Economics: The governance of contractual relations. *Journal of Law and Economics* 22(2). p 233-261

Wong, S. (2006). *Foundations of Paul Samuelson's Revealed Preference Theory*. Routledge. Boston.

Yonghap, P. (2016, June 26). Park Vows to Make Jeju and Island of Smart-Tourism and Energy Self-sufficiency. *Yonghap News Agency*. National Politics and Diplomacy

Yoon, J. Y. (2016, February 10). Government will ease rules to nurture sharing economy. *Korea Times*. Retrieved from https://goo.gl/esnzS8

Yu, H. (2017, February 16). Marriott and Hilton Stay Ahead of the Sharing Economy, Proving thar Airbnb is not the Uber of Hotels. *Forbes*. Retrieved from https://goo.gl/bSxnbG

Yuija, C. (2015, July). Why people choose Airbnb over hotel. Paper presented at 82nd Annual TOSOK International Conference. Pyeongchang, South Korea.

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

Zervas, G. (2016). The Rise of the Sharing Economy: Estimating the Impact of Airbnb on the Hotel Industry. *Boston University Press*. Retrieved from https://goo.gl/rVbeaL

Zervas, G. (2015). A first look at Online Reputation on Airbnb, Where Every Stay os Above Average. *Social Science Research*. Retrieved from https://goo.gl/F7wu6a

Zinkhan, G. M. (1992). Human nature and models of consumer decision making. Journal of Advertising 21 (4), ii

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

APPENDIX A

Platform Home Sharing Scraping Tool “airbnb.py”

This tool is the which functionally retrieves the data from Airbnb’s listing pages.

```
# ============================================================================
# Airbnb web site scraper, for analysis of Airbnb listings
# Tom Slee, 2013--2017.
#
# function naming conventions:
#   ws_get = get from web site
#   db_get = get from database
#   db_add = add to the database
#
# function name conventions:
#   add = add to database
#   display = open a browser and show
#   list = get from database and print
#   print = get from web site and print
# ============================================================================
import logging
import argparse
import sys
import time
import requests
from lxml import html
from datetime import datetime
import psycopg2
import psycopg2.errorcodes
import webbrowser
from airbnb_config import ABConfig
from airbnb_survey import ABSurvey, ABSurveyByBoundingBox
from airbnb_survey import ABSurveyByNeighborhood, ABSurveyByZipcode
from airbnb_listing import ABListing
import airbnb_ws

# ============================================================================
# CONSTANTS
# ============================================================================


logger = logging.getLogger()


def list_search_area_info(config, search_area):
    try:
        conn = config.connect()
        cur = conn.cursor()
        cur.execute("""
                select search_area_id
                from search_area where name=%s
                """, (search_area,))
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
        result_set = cur.fetchall()
        cur.close()
        count = len(result_set)
        if count == 1:
            print("\nThere is one search area called",
                  str(search_area),
                  "in the database.")
        elif count > 1:
            print("\nThere are", str(count),
                  "cities called", str(search_area),
                  "in the database.")
        elif count < 1:
            print("\nThere are no cities called",
                  str(search_area),
                  "in the database.")
            sys.exit()
        sql_neighborhood = """select count(*) from neighborhood
        where search_area_id = %s"""
        sql_search_area = """select count(*) from search_area
        where search_area_id = %s"""
        for result in result_set:
            search_area_id = result[0]
            cur = conn.cursor()
            cur.execute(sql_neighborhood, (search_area_id,))
            count = cur.fetchone()[0]
            cur.close()
            print("\t" + str(count) + " neighborhoods.")
            cur = conn.cursor()
            cur.execute(sql_search_area, (search_area_id,))
            count = cur.fetchone()[0]
            cur.close()
            print("\t" + str(count) + " Airbnb cities.")
    except psycopg2.Error as pge:
        logger.error(pge.pgerror)
        logger.error("Error code " + pge.pgcode)
        logger.error("Diagnostics " + pge.diag.message_primary)
        cur.close()
        conn.rollback()
        raise
    except Exception:
        logger.error("Failed to list search area info")
        raise


def list_surveys(config):
    try:
        conn = config.connect()
        cur = conn.cursor()
        cur.execute("""
            select survey_id, to_char(survey_date, 'YYYY-Mon-DD'),
                    survey_description, search_area_id, status
            from survey
            where survey_date is not null
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
            and status is not null
            and survey_description is not null
            order by survey_id asc""")
        result_set = cur.fetchall()
        if len(result_set) > 0:
            template = "| {0:3} | {1:>12} | {2:>50} | {3:3} | {4:3} |"
            print(template.format("ID", "Date", "Description", "SA", "status"))
            for survey in result_set:
                (survey_id, survey_date, desc, sa_id, status) = survey
                print(template.format(survey_id, survey_date, desc, sa_id, status))
    except Exception:
        logger.error("Cannot list surveys.")
        raise


def db_ping(config):
    try:
        conn = config.connect()
        if conn is not None:
            print("Connection test succeeded")
        else:
            print("Connection test failed")
    except Exception:
        logger.exception("Connection test failed")


def db_add_survey(config, search_area):
    try:
        conn = config.connect()
        cur = conn.cursor()
        # Add an entry into the survey table, and get the survey_id
        sql = """
        insert into survey (survey_description, search_area_id)
        select (name || ' (' || current_date || ')') as survey_description,
        search_area_id
        from search_area
        where name = %s
        returning survey_id"""
        cur.execute(sql, (search_area,))
        survey_id = cur.fetchone()[0]

        # Get and print the survey entry
        cur.execute("""select survey_id, survey_date,
        survey_description, search_area_id
        from survey where survey_id = %s""", (survey_id,))
        (survey_id,
         survey_date,
         survey_description,
         search_area_id) = cur.fetchone()
        conn.commit()
        cur.close()
        print("\nSurvey added:\n" +
              "\n\tsurvey_id=" + str(survey_id) +
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
                "\n\tsurvey_date=" + str(survey_date) +
                "\n\tsurvey_description=" + survey_description +
                "\n\tsearch_area_id=" + str(search_area_id))
    except Exception:
        logger.error("Failed to add survey for " + search_area)
        raise


def db_get_room_to_fill(config, survey_id):
    for attempt in range(config.MAX_CONNECTION_ATTEMPTS):
        try:
            conn = config.connect()
            cur = conn.cursor()
            if survey_id == 0:  # no survey specified
                sql = """
                    select room_id, survey_id
                    from room
                    where deleted is null
                    order by random()
                    limit 1
                    """
                cur.execute(sql)
            else:
                sql = """
                    select room_id, survey_id
                    from room
                    where deleted is null
                    and survey_id = %s
                    order by random()
                    limit 1
                    """
                cur.execute(sql, (survey_id,))
            (room_id, survey_id) = cur.fetchone()
            listing = ABListing(config, room_id, survey_id)
            cur.close()
            conn.commit()
            return listing
        except TypeError:
            logger.info("Finishing: no unfilled rooms in database --")
            conn.rollback()
            del (config.connection)
            return None
        except Exception:
            logger.exception("Error retrieving room to fill from db")
            conn.rollback()
            del (config.connection)
    return None


def ws_get_city_info(config, city, flag):
    try:
        url = config.URL_SEARCH_ROOT + city
        response = airbnb_ws.ws_request_with_repeats(config, url)
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
if response is None:
    return False
tree = html.fromstring(response.text)
try:
    citylist = tree.xpath(
        "//input[@name='location']/@value")
    neighborhoods = tree.xpath(
        "//input[contains(@id, 'filter-option-neighborhoods')]/@value")
    if flag == config.FLAGS_PRINT:
        print("\n", citylist[0])
        print("Neighborhoods:")
        for neighborhood in neighborhoods:
            print("\t", neighborhood)
    elif flag == config.FLAGS_ADD:
        if len(citylist) > 0:
            conn = config.connect()
            cur = conn.cursor()
            # check if it exists
            sql_check = """
                select name
                from search_area
                where name = %s"""
            cur.execute(sql_check, (citylist[0],))
            if cur.fetchone() is not None:
                logger.info("City already exists: " + citylist[0])
                return
            sql_search_area = """insert
                        into search_area (name)
                        values (%s)"""
            cur.execute(sql_search_area, (citylist[0],))
            # city_id = cur.lastrowid
            sql_identity = """select
            currval('search_area_search_area_id_seq')
            """
            cur.execute(sql_identity, ())
            search_area_id = cur.fetchone()[0]
            sql_city = """insert
                    into city (name, search_area_id)
                    values (%s,%s)"""
            cur.execute(sql_city, (city, search_area_id,))
            logger.info("Added city " + city)
            logger.debug(str(len(neighborhoods)) + " neighborhoods")
        if len(neighborhoods) > 0:
            sql_neighborhood = """
                insert into neighborhood(name, search_area_id)
                values(%s, %s)
                """
            for neighborhood in neighborhoods:
                cur.execute(sql_neighborhood, (neighborhood,
                                              search_area_id,))
                logger.info("Added neighborhood " + neighborhood)
        else:
            logger.info("No neighborhoods found for " + city)
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
                conn.commit()
        except UnicodeEncodeError:
            # if sys.version_info >= (3,):
            #     logger.info(s.encode('utf8').decode(sys.stdout.encoding))
            # else:
            #     logger.info(s.encode('utf8'))
            # unhandled at the moment
            pass
        except Exception:
            logger.error("Error collecting city and neighborhood information")
            raise
    except Exception:
        logger.error("Error getting city info from website")
        raise


def display_room(config, room_id):
    webbrowser.open(config.URL_ROOM_ROOT + str(room_id))


def display_host(config, host_id):
    webbrowser.open(config.URL_HOST_ROOT + str(host_id))


def fill_loop_by_room(config, survey_id):
    """
    Master routine for looping over rooms (after a search)
    to fill in the properties.
    """
    room_count = 0
    while room_count < config.FILL_MAX_ROOM_COUNT:
        try:
            if len(config.HTTP_PROXY_LIST) == 0:
                logger.info(
                    "No proxies left: re-initialize after {0} seconds".format(
                        config.RE_INIT_SLEEP_TIME))
                time.sleep(config.RE_INIT_SLEEP_TIME)  # be nice
                config = ABConfig()
            room_count += 1
            listing = db_get_room_to_fill(config, survey_id)
            if listing is None:
                return None
            else:
                if listing.ws_get_room_info(config.FLAGS_ADD):
                    pass
                else:  # Airbnb now seems to return nothing if a room has gone
                    listing.save_as_deleted()
        except AttributeError:
            logger.error("Attribute error: marking room as deleted.")
            listing.save_as_deleted()
        except Exception as e:
            logger.error("Error in fill_loop_by_room:" + str(type(e)))
            raise
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
def parse_args():
    """
    Read and parse command-line arguments
    """
    parser = argparse.ArgumentParser(
        description='Manage a database of Airbnb listings.',
        usage='%(prog)s [options]')
    parser.add_argument("-v", "--verbose",
                        action="store_true", default=False,
                        help="""write verbose (debug) output to the log file""")
    parser.add_argument("-c", "--config_file",
                        metavar="config_file", action="store", default=None,
                        help="""explicitly set configuration file, instead of
                        using the default <username>.config""")
    # Only one argument!
    group = parser.add_mutually_exclusive_group()
    group.add_argument('-asa', '--addsearcharea',
                       metavar='search_area', action='store', default=False,
                       help="""get and save the name and neighborhoods
                       for search area (city)""")
    group.add_argument('-asv', '--addsurvey',
                       metavar='search_area', type=str,
                       help="""add a survey entry to the database,
                       for search_area""")
    group.add_argument('-dbp', '--dbping',
                       action='store_true', default=False,
                       help='Test the database connection')
    group.add_argument('-dh', '--displayhost',
                       metavar='host_id', type=int,
                       help='display web page for host_id in browser')
    group.add_argument('-dr', '--displayroom',
                       metavar='room_id', type=int,
                       help='display web page for room_id in browser')
    group.add_argument('-f', '--fill', nargs='?',
                       metavar='survey_id', type=int, const=0,
                       help='fill details for rooms collected with -s')
    group.add_argument('-lsa', '--listsearcharea',
                       metavar='search_area', type=str,
                       help="""list information about this search area
                       from the database""")
    group.add_argument('-lr', '--listroom',
                       metavar='room_id', type=int,
                       help='list information about room_id from the database')
    group.add_argument('-ls', '--listsurveys',
                       action='store_true', default=False,
                       help='list the surveys in the database')
    group.add_argument('-psa', '--printsearcharea',
                       metavar='search_area', action='store', default=False,
                       help="""print the name and neighborhoods for
                       search area (city) from the Airbnb web site""")
    group.add_argument('-pr', '--printroom',
                       metavar='room_id', type=int,
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
                        help="""print room_id information
                        from the Airbnb web site""")
    group.add_argument('-ps', '--printsearch',
                        metavar='survey_id', type=int,
                        help="""print first page of search information
                        for survey from the Airbnb web site""")
    group.add_argument('-psn', '--printsearch_by_neighborhood',
                        metavar='survey_id', type=int,
                        help="""print first page of search information
                        for survey from the Airbnb web site,
                        by neighborhood""")
    group.add_argument('-psz', '--printsearch_by_zipcode',
                        metavar='survey_id', type=int,
                        help="""print first page of search information
                        for survey from the Airbnb web site,
                        by zipcode""")
    group.add_argument('-psb', '--printsearch_by_bounding_box',
                        metavar='survey_id', type=int,
                        help="""print first page of search information
                        for survey from the Airbnb web site,
                        by bounding_box""")
    group.add_argument('-s', '--search',
                        metavar='survey_id', type=int,
                        help='search for rooms using survey survey_id')
    group.add_argument('-sn', '--search_by_neighborhood',
                        metavar='survey_id', type=int,
                        help='search for rooms using survey survey_id')
    group.add_argument('-sb', '--search_by_bounding_box',
                        metavar='survey_id', type=int,
                        help="""search for rooms using survey survey_id,
                        by bounding box
                        """)
    group.add_argument('-sz', '--search_by_zipcode',
                        metavar='survey_id', type=int,
                        help="""search for rooms using survey_id,
                        by zipcode""")
    group.add_argument('-V', '--version',
                        action='version',
                        version='%(prog)s, version ' +
                        str(SCRIPT_VERSION_NUMBER))
    group.add_argument('-?', action='help')

    args = parser.parse_args()
    return (parser, args)

def main():
    (parser, args) = parse_args()
    ab_config = ABConfig(args)
    try:
        if args.search:
            survey = ABSurveyByNeighborhood(ab_config, args.search)
            survey.search(ab_config.FLAGS_ADD)
        elif args.search_by_neighborhood:
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
            survey = ABSurveyByNeighborhood(ab_config, args.search_by_neighborhood)
            survey.search(ab_config.FLAGS_ADD)
        elif args.search_by_zipcode:
            survey = ABSurveyByZipcode(ab_config, args.search_by_zipcode)
            survey.search(ab_config.FLAGS_ADD)
        elif args.search_by_bounding_box:
            survey = ABSurveyByBoundingBox(ab_config, args.search_by_bounding_box)
            survey.search(ab_config.FLAGS_ADD)
        elif args.fill is not None:
            fill_loop_by_room(ab_config, args.fill)
        elif args.addsearcharea:
            ws_get_city_info(ab_config, args.addsearcharea, ab_config.FLAGS_ADD)
        elif args.addsurvey:
            db_add_survey(ab_config, args.addsurvey)
        elif args.dbping:
            db_ping(ab_config)
        elif args.displayhost:
            display_host(ab_config, args.displayhost)
        elif args.displayroom:
            display_room(ab_config, args.displayroom)
        elif args.listsearcharea:
            list_search_area_info(ab_config, args.listsearcharea)
        elif args.listroom:
            listing = ABListing(ab_config, args.listroom, None)
            listing.print_from_db()
        elif args.listsurveys:
            list_surveys(ab_config)
        elif args.printsearcharea:
            ws_get_city_info(ab_config, args.printsearcharea, ab_config.FLAGS_PRINT)
        elif args.printroom:
            listing = ABListing(ab_config, args.printroom, None)
            listing.ws_get_room_info(ab_config.FLAGS_PRINT)
        elif args.printsearch:
            survey = ABSurveyByNeighborhood(ab_config, args.printsearch)
            survey.search(ab_config.FLAGS_PRINT)
        elif args.printsearch_by_neighborhood:
            survey = ABSurveyByNeighborhood(ab_config, args.printsearch_by_neighborhood)
            survey.search(ab_config.FLAGS_PRINT)
        elif args.printsearch_by_bounding_box:
            survey = ABSurveyByBoundingBox(ab_config, args.printsearch_by_bounding_box)
            survey.search(ab_config.FLAGS_PRINT)
        elif args.printsearch_by_zipcode:
            survey = ABSurveyByZipcode(ab_config, args.printsearch_by_zipcode)
            survey.search(ab_config.FLAGS_PRINT)
        else:
            parser.print_help()
    except (SystemExit, KeyboardInterrupt):
        sys.exit()
    except Exception:
        logger.exception("Top level exception handler: quitting.")
        sys.exit(0)

if __name__ == "__main__": main()
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

APPENDIX B

Platform Home Sharing Scraper Configuration Tool “airbnb_config.py”

This tool is programmed to configure the web scraper located in appendix A. It uses several third-party modules to effectively call functions.

```
# ============================================================================
# Airbnb Configuration module, for use in web scraping and analytics
# ============================================================================
import logging
import psycopg2
import psycopg2.errorcodes
import os
import configparser
import sys
from datetime import datetime

logger = logging.getLogger()
logger.info("Logger got")

class ABConfig():

    def __init__(self, args=None):
        """ Read the configuration file <username>.config to set up the run
        """
        self.config_file = None
        self.log_level = logging.INFO
        if args is not None:
            self.config_file=args.config_file
            try:
                if args.verbose:
                    self.log_level = logging.DEBUG
                else:
                    self.log_level = logging.INFO
            except:
                self.log_level = logging.INFO
        self.connection = None
        self.FLAGS_ADD = 1
        self.FLAGS_PRINT = 9
        self.FLAGS_INSERT_REPLACE = True
        self.FLAGS_INSERT_NO_REPLACE = False
        self.URL_ROOT = "https://www.airbnb.com/"
        self.URL_ROOM_ROOT = self.URL_ROOT + "rooms/"
        self.URL_HOST_ROOT = self.URL_ROOT + "users/show/"
        self.URL_SEARCH_ROOT = self.URL_ROOT + "s/"
        self.URL_API_SEARCH_ROOT = self.URL_ROOT + "search/search_results"
        self.SEARCH_AREA_GLOBAL = "UNKNOWN"  # special case: sample listings globally
        self.SEARCH_RECTANGLE_EDGE_BLUR = 0.1
        self.SEARCH_BY_NEIGHBORHOOD = 'neighborhood'  # default
        self.SEARCH_BY_ZIPCODE = 'zipcode'
        self.SEARCH_BY_BOUNDING_BOX = 'bounding box'
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
        self.SEARCH_LISTINGS_ON_FULL_PAGE = 18
        self.HTTP_PROXY_LIST = []
        self.HTTP_PROXY_LIST_COMPLETE = []

        try:
            config = configparser.ConfigParser()

            if self.config_file is None:
                # look for username.config on both Windows (USERNAME) and Linux (USER)
                if os.name == "nt":
                    username = os.environ['USERNAME']
                else:
                    username = os.environ['USER']
                self.config_file = username + ".config"
            logging.info("Reading configuration file " + self.config_file)
            if not os.path.isfile(self.config_file):
                logging.error("Configuration file " + self.config_file + " not found.")
                sys.exit()
            config.read(self.config_file)

            # database
            try:
                self.DB_HOST     =     config["DATABASE"]["db_host"]     if     ("db_host"     in
config["DATABASE"]) else None
                self.DB_PORT = config["DATABASE"]["db_port"]
                self.DB_NAME = config["DATABASE"]["db_name"]
                self.DB_USER = config["DATABASE"]["db_user"]
                self.DB_PASSWORD = config["DATABASE"]["db_password"]
            except Exception:
                logger.error("Incomplete database information in " + config_file + ": cannot
continue.")
                sys.exit()
            # network
            try:
                self.HTTP_PROXY_LIST = config["NETWORK"]["proxy_list"].split(",")
                self.HTTP_PROXY_LIST = [x.strip() for x in self.HTTP_PROXY_LIST]
            except Exception:
                logger.warning("No proxy_list in " + config_file + ": not using proxies")
                self.HTTP_PROXY_LIST = []
            self.HTTP_PROXY_LIST_COMPLETE = list(self.HTTP_PROXY_LIST)
            logging.info("Complete          proxy          list          has          {p}
proxies".format(p=len(self.HTTP_PROXY_LIST_COMPLETE)))
            try:
                self.USER_AGENT_LIST = config["NETWORK"]["user_agent_list"].split(",,")
                self.USER_AGENT_LIST = [x.strip() for x in self.USER_AGENT_LIST]
                self.USER_AGENT_LIST = [x.strip('"') for x in self.USER_AGENT_LIST]
            except Exception:
                logger.info("No user agent list in " + username +
                            ".config: not using user agents")
                self.USER_AGENT_LIST = []
            self.MAX_CONNECTION_ATTEMPTS = \
                int(config["NETWORK"]["max_connection_attempts"])
            self.REQUEST_SLEEP = float(config["NETWORK"]["request_sleep"])
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
            self.HTTP_TIMEOUT = float(config["NETWORK"]["http_timeout"])

            # survey
            self.FILL_MAX_ROOM_COUNT = int(config["SURVEY"]["fill_max_room_count"])
            self.ROOM_ID_UPPER_BOUND = int(config["SURVEY"]["room_id_upper_bound"])
            self.SEARCH_MAX_PAGES = int(config["SURVEY"]["search_max_pages"])
            self.SEARCH_MAX_GUESTS = int(config["SURVEY"]["search_max_guests"])
            self.SEARCH_MAX_RECTANGLE_ZOOM = int(
                config["SURVEY"]["search_max_rectangle_zoom"])
            self.RE_INIT_SLEEP_TIME = float(config["SURVEY"]["re_init_sleep_time"])

        except Exception:
            logger.exception("Failed to read config file properly")
            raise

    def connect(self):
    # get a database connection
        """ Return a connection to the database"""
        try:
            if (not hasattr(self, "connection") or
                self.connection is None or self.connection.closed != 0):
                cattr = dict(
                    user=self.DB_USER,
                    password=self.DB_PASSWORD,
                    database=self.DB_NAME
                )
                if self.DB_HOST is not None:
                    cattr.update(dict(
                                host=self.DB_HOST,
                                port=self.DB_PORT,
                                ))
                self.connection = psycopg2.connect(**cattr)
                self.connection.set_client_encoding('UTF8')
            return self.connection
        except psycopg2.OperationalError as pgoe:
            logger.error(pgoe.message)
            raise
        except Exception:
            logger.error("Failed to connect to database.")
            raise
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

APPENDIX C

Platform Home Sharing Website Requests “airbnb_ws.py”

This tool imports key m,odules to handle requests from Airbnb's website.

```
import logging
import sys
import random
import time
import requests
from airbnb_config import ABConfig

# Set up logging
logger = logging.getLogger(__name__)


def ws_request_with_repeats(config, url, params=None):
    # Return None on failure
    for attempt_id in range(config.MAX_CONNECTION_ATTEMPTS):
        try:
            response = ws_request(config, url, attempt_id, params)
            if response is None:
                continue
            elif response.status_code == requests.codes.ok:
                return response
        except (SystemExit, KeyboardInterrupt):
            raise
        except AttributeError:
            logger.exception("AttributeError retrieving page")
        except Exception as ex:
            logger.error("Failed to retrieve web page " + url)
            logger.exception("Exception retrieving page: " + str(type(ex)))
            # Failed
    return None


def ws_request(config, url, attempt_id, params=None):
    """
    Individual web request: returns a response object or None on failure
    """
    try:
        # wait
        sleep_time = config.REQUEST_SLEEP * random.random()
        logger.debug("sleeping " + str(sleep_time)[:7] + " seconds...")
        time.sleep(sleep_time)  # be nice

        timeout = config.HTTP_TIMEOUT

        # If a list of user agent strings is supplied, use it
        if len(config.USER_AGENT_LIST) > 0:
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
            user_agent = random.choice(config.USER_AGENT_LIST)
            headers = {"User-Agent": user_agent}
        else:
            headers = {'User-Agent': 'Mozilla/5.0'}

        # If there is a list of proxies supplied, use it
        http_proxy = None
        logger.debug("Using " + str(len(config.HTTP_PROXY_LIST)) + " proxies.")
        if len(config.HTTP_PROXY_LIST) > 0:
            http_proxy = random.choice(config.HTTP_PROXY_LIST)
            proxies = {
                'http': http_proxy,
                'https': http_proxy,
            }
            logger.debug("Requesting page through proxy " + http_proxy)
        else:
            proxies = None

        # Now make the request
        response = requests.get(url, params, timeout=timeout,
                                headers=headers, proxies=proxies)
        if response.status_code == 503:
            if http_proxy:
                logger.warning("HTTP 503 error from web site: IP address {a} blocked"
                    .format(a=http_proxy))
                if len(config.HTTP_PROXY_LIST) > 0:
                    # randomly remove the proxy from the list, with probability 50%
                    if random.choice([True, False]):
                        config.HTTP_PROXY_LIST.remove(http_proxy)
                        logger.warning(
                            "Removing {http_proxy} from proxy list; {n} of {p} remain."
                            .format( http_proxy=http_proxy,
                                n=len(config.HTTP_PROXY_LIST),
                                p=len(config.HTTP_PROXY_LIST_COMPLETE)))
                    else:
                        logger.warning(
                            "Not removing {http_proxy} from proxy list this time; still {n} of
{p}."
                            .format( http_proxy=http_proxy,
                                n=len(config.HTTP_PROXY_LIST),
                                p=len(config.HTTP_PROXY_LIST_COMPLETE)))
                if len(config.HTTP_PROXY_LIST) == 0:
                    # fill proxy list again, wait a long time, then restart
                    logger.warning("No proxies remain. Resetting proxy list and waiting {m}
minutes."
                        .format(m=(config.RE_INIT_SLEEP_TIME / 60.0)))
                    config.HTTP_PROXY_LIST                                        =
list(config.HTTP_PROXY_LIST_COMPLETE)
                    time.sleep(config.RE_INIT_SLEEP_TIME)
                    config.REQUEST_SLEEP += 1.0
                    logger.warning("Adding one second to request sleep time. Now {s}"
                        .format(s=config.REQUEST_SLEEP))
            else:
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
                logger.warning("HTTP 503 error from web site: IP address blocked. Waiting {m} minutes."
                        .format(m=(config.RE_INIT_SLEEP_TIME / 60.0)))
                time.sleep(config.RE_INIT_SLEEP_TIME)
                config.REQUEST_SLEEP += 1.0
        return response
    except (SystemExit, KeyboardInterrupt):
        raise
    except requests.exceptions.ConnectionError:
        # For requests error and exceptions, see
        # http://docs.python-requests.org/en/latest/user/quickstart/
        # errors-and-exceptions
        logger.warning("Network request exception {a}: connectionError".format(a=attempt_id))
        return None
    except requests.exceptions.HTTPError:
        logger.error("Network request exception {a}: invalid HTTP response".format(a=attempt_id))
        return None
    except requests.exceptions.Timeout:
        logger.warning("Network request exception {a}: timeout".format(a=attempt_id))
        return None
    except requests.exceptions.TooManyRedirects:
        logger.error("Network request exception {a}: too many redirects".format(a=attempt_id))
        return None
    except requests.exceptions.RequestException:
        logger.error("Network request exception {a}: unidentified requests".format(a=attempt_id))
        return None
    except Exception as e:
        logger.exception("Network request exception: type " + type(e).__name__)
        return None
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

APPENDIX D

Platform Home Sharing Listing Scraping Tool “airbnb_listing.py”

This tool is designed to populate the fields of a single Airbnb listing, or room_id.

```
import logging
import re
from lxml import html
import psycopg2
import json
import airbnb_ws

logger = logging.getLogger()


class ABListing():
    """
    # ABListing represents an Airbnb room_id, as captured at a moment in time.
    # room_id, survey_id is the primary key.
    # Occasionally, a survey_id = None will happen, but for retrieving data
    # straight from the web site, and not stored in the database.
    """
    def __init__(self, config, room_id, survey_id, room_type=None):
        self.config = config
        self.room_id = room_id
        self.host_id = None
        self.room_type = room_type
        self.country = None
        self.city = None
        self.neighborhood = None
        self.address = None
        self.reviews = None
        self.overall_satisfaction = None
        self.accommodates = None
        self.bedrooms = None
        self.bathrooms = None
        self.price = None
        self.deleted = None
        self.minstay = None
        self.latitude = None
        self.longitude = None
        self.survey_id = survey_id
        #  extra fields added from search json:
        # coworker_hosted (bool)
        self.coworker_hosted = None
        # extra_host_languages (list)
        self.extra_host_languages = None
        # name (str)
        self.name = None
        # property_type (str)
        self.property_type = None
        # currency (str)
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
        self.currency = None
        # rate_type (str) - "nightly" or other?
        self.rate_type = None
        """ """

    def status_check(self):
        status = True  # OK
        # if sufficient of the values are None or don't exist, the room
        # entry was not properly parsed and we may as well throw the whole
        # thing away.
        unassigned_values = {key: value
                             for key, value in vars(self).items()
                             if not key.startswith('__') and
                             not callable(key) and
                             value is None
                             }
        if len(unassigned_values) > 9:  # just a value indicating deleted
            logger.info("Room " + str(self.room_id) + ": marked deleted")
            status = False  # probably deleted
            self.deleted = 1
        else:
            for key, val in unassigned_values.items():
                if (key == "overall_satisfaction" and "reviews" not in
                        unassigned_values):
                    if val is None and self.reviews > 2:
                        logger.debug("Room " + str(self.room_id) + ": No value for " + key)
                elif val is None:
                    logger.debug("Room " + str(self.room_id) + ": No value for " + key)
        return status

    def get_columns(self):

        # columns = [attr for attr in dir(self) if not
        # callable(attr) and not attr.startswith("__")]
        columns = ("room_id", "host_id", "room_type", "country",
                   "city", "neighborhood", "address", "reviews",
                   "overall_satisfaction", "accommodates", "bedrooms",
                   "bathrooms", "price", "deleted", "minstay",
                   "latitude", "longitude", "survey_id", "last_modified",)
        return columns

    def save_as_deleted(self):
        try:
            logger.debug("Marking room deleted: " + str(self.room_id))
            if self.survey_id is None:
                return
            conn = self.config.connect()
            sql = """
                update room
                set deleted = 1, last_modified = now()::timestamp
                where room_id = %s
                and survey_id = %s
            """
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
            cur = conn.cursor()
            cur.execute(sql, (self.room_id, self.survey_id))
            cur.close()
            conn.commit()
        except Exception:
            logger.error("Failed to save room as deleted")
            raise

    def save(self, insert_replace_flag):
        """
        Save a listing in the database. Delegates to lower-level methods
        to do the actual database operations.
        Return values:
            True: listing is saved in the database
            False: listing already existed
        """
        try:
            rowcount = -1
            if self.deleted == 1:
                self.save_as_deleted()
            else:
                if insert_replace_flag == self.config.FLAGS_INSERT_REPLACE:
                    rowcount = self.__update()
                if (rowcount == 0 or
                        insert_replace_flag == self.config.FLAGS_INSERT_NO_REPLACE):
                    try:
                        self.__insert()
                        return True
                    except psycopg2.IntegrityError:
                        logger.debug("Room " + str(self.room_id) + ": already collected")
                        return False
        except psycopg2.OperationalError:
            # connection closed
            logger.error("Operational error (connection closed): resuming")
            del(self.config.connection)
        except psycopg2.DatabaseError as de:
            self.config.connection.conn.rollback()
            logger.erro(psycopg2.errorcodes.lookup(de.pgcode[:2]))
            logger.error("Database error: resuming")
            del(self.config.connection)
        except psycopg2.InterfaceError:
            # connection closed
            logger.error("Interface error: resuming")
            del(self.config.connection)
        except psycopg2.Error as pge:
            # database error: rollback operations and resume
            self.config.connection.conn.rollback()
            logger.error("Database error: " + str(self.room_id))
            logger.error("Diagnostics " + pge.diag.message_primary)
            del(self.config.connection)
        except (KeyboardInterrupt, SystemExit):
            raise
        except UnicodeEncodeError as uee:
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
            logger.error("UnicodeEncodeError Exception at " +
                         str(uee.object[uee.start:uee.end]))
            raise
        except ValueError:
            logger.error("ValueError for room_id = " + str(self.room_id))
        except AttributeError:
            logger.error("AttributeError")
            raise
        except Exception:
            self.config.connection.rollback()
            logger.error("Exception saving room")
            raise

    def print_from_web_site(self):
        """ What is says """
        try:
            print_string = "Room info:"
            print_string += "\n\troom_id:\t" + str(self.room_id)
            print_string += "\n\tsurvey_id:\t" + str(self.survey_id)
            print_string += "\n\thost_id:\t" + str(self.host_id)
            print_string += "\n\troom_type:\t" + str(self.room_type)
            print_string += "\n\tcountry:\t" + str(self.country)
            print_string += "\n\tcity:\t\t" + str(self.city)
            print_string += "\n\tneighborhood:\t" + str(self.neighborhood)
            print_string += "\n\taddress:\t" + str(self.address)
            print_string += "\n\treviews:\t" + str(self.reviews)
            print_string += "\n\toverall_satisfaction:\t"
            print_string += str(self.overall_satisfaction)
            print_string += "\n\taccommodates:\t" + str(self.accommodates)
            print_string += "\n\tbedrooms:\t" + str(self.bedrooms)
            print_string += "\n\tbathrooms:\t" + str(self.bathrooms)
            print_string += "\n\tprice:\t\t" + str(self.price)
            print_string += "\n\tdeleted:\t" + str(self.deleted)
            print_string += "\n\tlatitude:\t" + str(self.latitude)
            print_string += "\n\tlongitude:\t" + str(self.longitude)
            print_string += "\n\tminstay:\t" + str(self.minstay)
            print_string += "\n\tcoworker_hosted:\t" + str(self.coworker_hosted)
            print_string += "\n\tlanguages:\t" + str(self.extra_host_languages)
            print_string += "\n\tproperty_type:\t" + str(self.property_type)
            print(print_string)
        except Exception:
            raise

    def print_from_db(self):
        """ What it says """
        try:
            columns = self.get_columns()
            sql = "select room_id"
            for column in columns[1:]:
                sql += ", " + column
            sql += " from room where room_id = %s"
            conn = self.config.connect()
            cur = conn.cursor()
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
            cur.execute(sql, (self.room_id,))
            result_set = cur.fetchall()
            if len(result_set) > 0:
                for result in result_set:
                    i = 0
                    print("Room information: ")
                    for column in columns:
                        print("\t", column, "=", str(result[i]))
                        i += 1
                return True
            else:
                print("\nNo room", str(self.room_id), "in the database.\n")
                return False
            cur.close()
        except Exception:
            raise

    def ws_get_room_info(self, flag):
        """ Get the room properties from the web site """
        try:
            # initialization
            logger.info("-" * 70)
            logger.info("Room " + str(self.room_id) +
                        ": getting from Airbnb web site")
            room_url = self.config.URL_ROOM_ROOT + str(self.room_id)
            response = airbnb_ws.ws_request_with_repeats(self.config, room_url)
            if response is not None:
                page = response.text
                tree = html.fromstring(page)
                self.__get_room_info_from_tree(tree, flag)
                return True
            else:
                return False
        except (KeyboardInterrupt, SystemExit):
            raise
        except Exception as ex:
            logger.exception("Room " + str(self.room_id) +
                             ": failed to retrieve from web site.")
            logger.error("Exception: " + str(type(ex)))
            raise

    def __insert(self):
        """ Insert a room into the database. Raise an error if it fails """
        try:
            logger.debug("Values: ")
            logger.debug("\troom_id: {}".format(self.room_id))
            logger.debug("\thost_id: {}".format(self.host_id))
            conn = self.config.connect()
            cur = conn.cursor()
            sql = """
                insert into room (
                    room_id, host_id, room_type, country, city,
                    neighborhood, address, reviews, overall_satisfaction,
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
                    accommodates, bedrooms, bathrooms, price, deleted,
                    minstay, latitude, longitude, survey_id,
                    coworker_hosted, extra_host_languages, name,
                    property_type, currency, rate_type

                )
                """
            sql += """
                values (%s, %s, %s, %s, %s, %s, %s, %s, %s,
                %s, %s, %s, %s, %s, %s, %s, %s, %s,
                %s, %s, %s, %s, %s, %s
                )"""
            insert_args = (
                self.room_id, self.host_id, self.room_type, self.country,
                self.city, self.neighborhood, self.address, self.reviews,
                self.overall_satisfaction, self.accommodates, self.bedrooms,
                self.bathrooms, self.price, self.deleted, self.minstay,
                self.latitude, self.longitude, self.survey_id,
                self.coworker_hosted, self.extra_host_languages, self.name,
                self.property_type, self.currency, self.rate_type
                )
            cur.execute(sql, insert_args)
            cur.close()
            conn.commit()
            logger.debug("Room " + str(self.room_id) + ": inserted")
            logger.debug("(lat, long) = ({lat:+.5f}, {lng:+.5f})".format(lat=self.latitude,
lng=self.longitude))
        except psycopg2.IntegrityError:
            # logger.info("Room " + str(self.room_id) + ": insert failed")
            conn.rollback()
            cur.close()
            raise
        except:
            conn.rollback()
            raise

    def __update(self):
        """ Update a room in the database. Raise an error if it fails.
        Return number of rows affected."""
        try:
            rowcount = 0
            conn = self.config.connect()
            cur = conn.cursor()
            logger.debug("Updating...")
            sql = """
                update room
                set host_id = %s, room_type = %s,
                    country = %s, city = %s, neighborhood = %s,
                    address = %s, reviews = %s, overall_satisfaction = %s,
                    accommodates = %s, bedrooms = %s, bathrooms = %s,
                    price = %s, deleted = %s, last_modified = now()::timestamp,
                    minstay = %s, latitude = %s, longitude = %s,
                    coworker_hosted = %s, extra_host_languages = %s, name = %s,
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
                        property_type = %s, currency = %s, rate_type = %s
                    where room_id = %s
                    and survey_id = %s"""
                update_args = (
                    self.host_id, self.room_type,
                    self.country, self.city, self.neighborhood,
                    self.address, self.reviews, self.overall_satisfaction,
                    self.accommodates, self.bedrooms, self.bathrooms,
                    self.price, self.deleted,
                    self.minstay, self.latitude,
                    self.longitude,
                    self.coworker_hosted, self.extra_host_languages, self.name,
                    self.property_type, self.currency, self.rate_type,
                    self.room_id,
                    self.survey_id,
                    )
                logger.debug("Executing...")
                cur.execute(sql, update_args)
                rowcount = cur.rowcount
                logger.debug("Closing...")
                cur.close()
                conn.commit()
                logger.info("Room " + str(self.room_id) +
                            ": updated (" + str(rowcount) + ")")
                return rowcount
            except:
                # may want to handle connection close errors
                logger.warning("Exception in __update: raising")
                raise

        def __get_country(self, tree):
            try:
                temp = tree.xpath(
                    "//meta[contains(@property,'airbedandbreakfast:country')]"
                    "/@content"
                    )
                if len(temp) > 0:
                    self.country = temp[0]
            except:
                raise

        def __get_city(self, tree):
            try:
                temp = tree.xpath(
                    "//meta[contains(@property,'airbedandbreakfast:city')]"
                    "/@content"
                    )
                if len(temp) > 0:
                    self.city = temp[0]
            except:
                raise

        def __get_rating(self, tree):
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
        try:
            # 2016-04-10
            s = tree.xpath("//meta[@id='_bootstrap-listing']/@content")
            temp = tree.xpath(
                "//meta[contains(@property,'airbedandbreakfast:rating')]"
                "/@content"
                )
            if s is not None:
                j = json.loads(s[0])
                self.overall_satisfaction = j["listing"]["star_rating"]
            elif len(temp) > 0:
                self.overall_satisfaction = temp[0]
        except IndexError:
            return
        except:
            raise

    def __get_latitude(self, tree):
        try:
            temp = tree.xpath("//meta"
                              "[contains(@property,"
                              "'airbedandbreakfast:location:latitude')]"
                              "/@content")
            if len(temp) > 0:
                self.latitude = temp[0]
        except:
            raise

    def __get_longitude(self, tree):
        try:
            temp = tree.xpath(
                "//meta"
                "[contains(@property,'airbedandbreakfast:location:longitude')]"
                "/@content")
            if len(temp) > 0:
                self.longitude = temp[0]
        except:
            raise

    def __get_host_id(self, tree):
        try:
            # 2016-04-10
            s = tree.xpath("//meta[@id='_bootstrap-listing']/@content")
            temp = tree.xpath(
                "//div[@id='host-profile']"
                "//a[contains(@href,'/users/show')]"
                "/@href"
            )
            if s is not None:
                j = json.loads(s[0])
                self.host_id = j["listing"]["user"]["id"]
                return
            elif len(temp) > 0:
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
                host_id_element = temp[0]
                host_id_offset = len('/users/show/')
                self.host_id = int(host_id_element[host_id_offset:])
            else:
                temp = tree.xpath(
                    "//div[@id='user']"
                    "//a[contains(@href,'/users/show')]"
                    "/@href")
                if len(temp) > 0:
                    host_id_element = temp[0]
                    host_id_offset = len('/users/show/')
                    self.host_id = int(host_id_element[host_id_offset:])
        except IndexError:
            return
        except:
            raise

    def __get_room_type(self, tree):
        try:
            # -- room type --
            # new page format 2015-09-30?
            temp = tree.xpath(
                "//div[@class='col-md-6']"
                "/div/span[text()[contains(.,'Room type:')]]"
                "/../strong/text()"
                )
            if len(temp) > 0:
                self.room_type = temp[0].strip()
            else:
                # new page format 2014-12-26
                temp_entire = tree.xpath(
                    "//div[@id='summary']"
                    "//i[contains(concat(' ', @class, ' '),"
                    " ' icon-entire-place ')]"
                    )
                if len(temp_entire) > 0:
                    self.room_type = "Entire home/apt"
                temp_private = tree.xpath(
                    "//div[@id='summary']"
                    "//i[contains(concat(' ', @class, ' '),"
                    " ' icon-private-room ')]"
                    )
                if len(temp_private) > 0:
                    self.room_type = "Private room"
                temp_shared = tree.xpath(
                    "//div[@id='summary']"
                    "//i[contains(concat(' ', @class, ' '),"
                    " ' icon-shared-room ')]"
                    )
                if len(temp_shared) > 0:
                    self.room_type = "Shared room"
        except:
            raise
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
def __get_neighborhood(self, tree):
    try:
        temp2 = tree.xpath(
            "//div[contains(@class,'rich-toggle')]/@data-address"
            )
        temp1 = tree.xpath("//table[@id='description_details']"
                           "//td[text()[contains(.,'Neighborhood:')]]"
                           "/following-sibling::td/descendant::text()")
        if len(temp2) > 0:
            temp = temp2[0].strip()
            self.neighborhood = temp[temp.find("(")+1:temp.find(")")]
        elif len(temp1) > 0:
            self.neighborhood = temp1[0].strip()
        if self.neighborhood is not None:
            self.neighborhood = self.neighborhood[:50]
    except:
        raise

def __get_address(self, tree):
    try:
        temp = tree.xpath(
            "//div[contains(@class,'rich-toggle')]/@data-address"
            )
        if len(temp) > 0:
            temp = temp[0].strip()
            self.address = temp[:temp.find(",")]
        else:
            # try old page match
            temp = tree.xpath(
                "//span[@id='display-address']"
                "/@data-location"
                )
            if len(temp) > 0:
                self.address = temp[0]
    except:
        raise

def __get_reviews(self, tree):
    try:
        # 2016-04-10
        s = tree.xpath("//meta[@id='_bootstrap-listing']/@content")
        # 2015-10-02
        temp2 = tree.xpath(
            "//div[@class='___iso-state___p3summarybundlejs']"
            "/@data-state"
            )
        if s is not None:
            j = json.loads(s[0])
            self.reviews = \
                j["listing"]["review_details_interface"]["review_count"]
        elif len(temp2) == 1:
            summary = json.loads(temp2[0])
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
                self.reviews = summary["visibleReviewCount"]
            elif len(temp2) == 0:
                temp = tree.xpath(
                    "//div[@id='room']/div[@id='reviews']//h4/text()")
                if len(temp) > 0:
                    self.reviews = temp[0].strip()
                    self.reviews = str(self.reviews).split('+')[0]
                    self.reviews = str(self.reviews).split(' ')[0].strip()
                if self.reviews == "No":
                    self.reviews = 0
            else:
                # try old page match
                temp = tree.xpath(
                    "//span[@itemprop='reviewCount']/text()"
                    )
                if len(temp) > 0:
                    self.reviews = temp[0]
            if self.reviews is not None:
                self.reviews = int(self.reviews)
        except IndexError:
            return
        except Exception as e:
            logger.exception(e)
            self.reviews = None

    def __get_accommodates(self, tree):
        try:
            # 2016-04-10
            s = tree.xpath("//meta[@id='_bootstrap-listing']/@content")
            temp = tree.xpath(
                "//div[@class='col-md-6']"
                "/div/span[text()[contains(.,'Accommodates:')]]"
                "/../strong/text()"
                )
            if s is not None:
                j = json.loads(s[0])
                self.accommodates = j["listing"]["person_capacity"]
                return
            elif len(temp) > 0:
                self.accommodates = temp[0].strip()
            else:
                temp = tree.xpath(
                    "//div[@class='col-md-6']"
                    "/div[text()[contains(.,'Accommodates:')]]"
                    "/strong/text()"
                    )
                if len(temp) > 0:
                    self.accommodates = temp[0].strip()
                else:
                    temp = tree.xpath(
                        "//div[@class='col-md-6']"
                        "//div[text()[contains(.,'Accommodates:')]]"
                        "/strong/text()"
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
                    )
                    if len(temp) > 0:
                        self.accommodates = temp[0].strip()
            if type(self.accommodates) == str:
                self.accommodates = self.accommodates.split('+')[0]
                self.accommodates = self.accommodates.split(' ')[0]
            self.accommodates = int(self.accommodates)
        except:
            self.accommodates = None

    def __get_bedrooms(self, tree):
        try:
            temp = tree.xpath(
                "//div[@class='col-md-6']"
                "/div/span[text()[contains(.,'Bedrooms:')]]"
                "/../strong/text()"
                )
            if len(temp) > 0:
                self.bedrooms = temp[0].strip()
            else:
                temp = tree.xpath(
                    "//div[@class='col-md-6']"
                    "/div[text()[contains(.,'Bedrooms:')]]"
                    "/strong/text()"
                    )
                if len(temp) > 0:
                    self.bedrooms = temp[0].strip()
            if self.bedrooms:
                self.bedrooms = self.bedrooms.split('+')[0]
                self.bedrooms = self.bedrooms.split(' ')[0]
            self.bedrooms = float(self.bedrooms)
        except:
            self.bedrooms = None

    def __get_bathrooms(self, tree):
        try:
            temp = tree.xpath(
                "//div[@class='col-md-6']"
                "/div/span[text()[contains(.,'Bathrooms:')]]"
                "/../strong/text()"
                )
            if len(temp) > 0:
                self.bathrooms = temp[0].strip()
            else:
                temp = tree.xpath(
                    "//div[@class='col-md-6']"
                    "/div/span[text()[contains(.,'Bathrooms:')]]"
                    "/../strong/text()"
                    )
                if len(temp) > 0:
                    self.bathrooms = temp[0].strip()
            if self.bathrooms:
                self.bathrooms = self.bathrooms.split('+')[0]
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
                self.bathrooms = self.bathrooms.split(' ')[0]
            self.bathrooms = float(self.bathrooms)
        except:
            self.bathrooms = None

    def __get_minstay(self, tree):
        try:
            # -- minimum stay --
            temp3 = tree.xpath(
                "//div[contains(@class,'col-md-6')"
                "and text()[contains(.,'minimum stay')]]"
                "/strong/text()"
                )
            temp2 = tree.xpath(
                "//div[@id='details-column']"
                "//div[contains(text(),'Minimum Stay:')]"
                "/strong/text()"
                )
            temp1 = tree.xpath(
                "//table[@id='description_details']"
                "//td[text()[contains(.,'Minimum Stay:')]]"
                "/following-sibling::td/descendant::text()"
                )
            if len(temp3) > 0:
                self.minstay = temp3[0].strip()
            elif len(temp2) > 0:
                self.minstay = temp2[0].strip()
            elif len(temp1) > 0:
                self.minstay = temp1[0].strip()
            if self.minstay is not None:
                self.minstay = self.minstay.split('+')[0]
                self.minstay = self.minstay.split(' ')[0]
            self.minstay = int(self.minstay)
        except:
            self.minstay = None

    def __get_price(self, tree):
        try:
            temp2 = tree.xpath(
                "//meta[@itemprop='price']/@content"
                )
            temp1 = tree.xpath(
                "//div[@id='price_amount']/text()"
                )
            if len(temp2) > 0:
                self.price = temp2[0]
            elif len(temp1) > 0:
                self.price = temp1[0][1:]
                non_decimal = re.compile(r'[^\d.]+')
                self.price = non_decimal.sub('', self.price)
            # Now find out if it's per night or per month
            # (see if the per_night div is hidden)
            per_month = tree.xpath(
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
                "//div[@class='js-per-night book-it__payment-period  hide']")
            if per_month:
                self.price = int(int(self.price) / 30)
            self.price = int(self.price)
        except:
            self.price = None

    def __get_room_info_from_tree(self, tree, flag):
        try:
            # Some of these items do not appear on every page (eg,
            # ratings, bathrooms), and so their absence is marked with
            # logger.info. Others should be present for every room (eg,
            # latitude, room_type, host_id) and so are marked with a
            # warning.  Items coded in <meta
            # property="airbedandbreakfast:*> elements -- country --

            self.__get_country(tree)
            self.__get_city(tree)
            self.__get_rating(tree)
            self.__get_latitude(tree)
            self.__get_longitude(tree)
            self.__get_host_id(tree)
            self.__get_room_type(tree)
            self.__get_neighborhood(tree)
            self.__get_address(tree)
            self.__get_reviews(tree)
            self.__get_accommodates(tree)
            self.__get_bedrooms(tree)
            self.__get_bathrooms(tree)
            self.__get_minstay(tree)
            self.__get_price(tree)
            self.deleted = 0

            # NOT FILLING HERE, but maybe should? have to write helper methods:
            # coworker_hosted, extra_host_languages, name,
            #     property_type, currency, rate_type

            self.status_check()

            if flag == self.config.FLAGS_ADD:
                self.save(self.config.FLAGS_INSERT_REPLACE)
            elif flag == self.config.FLAGS_PRINT:
                self.print_from_web_site()
            return True
        except (KeyboardInterrupt, SystemExit):
            raise
        except IndexError:
            logger.exception("Web page has unexpected structure.")
            raise
        except UnicodeEncodeError as uee:
            logger.exception("UnicodeEncodeError Exception at " +
                             str(uee.object[uee.start:uee.end]))
            raise
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
except AttributeError:
    logger.exception("AttributeError")
    raise
except TypeError:
    logger.exception("TypeError parsing web page.")
    raise
except Exception:
    logger.exception("Error parsing web page.")
    raise
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

APPENDIX E

Platform Home Sharing Survey Tool "airbnb_survey.py"

This tool organizes how surveys are run, whether by bounding box as in this research or through zip code areas or neighborhoods. It gathers the listing information within the designated search areas.

```
# ============================================================================
# - bounding box (-sb)
# ============================================================================
import logging
import sys
import random
import psycopg2
from datetime import date
from airbnb_listing import ABListing
import airbnb_ws

logger = logging.getLogger()


class ABSurvey():

    def __init__(self, config, survey_id):
        self.config = config
        self.survey_id = survey_id
        self.search_area_id = None
        self.search_area_name = None
        self.set_search_area()
        self.room_types = ["Private room", "Entire home/apt", "Shared room"]

        # Set up logging
        logger.setLevel(config.log_level)
        # create a file handler
        logfile = "survey_{survey_id}.log".format(survey_id=self.survey_id)
        filelog_handler = logging.FileHandler(logfile, encoding="utf-8")
        filelog_formatter = logging.Formatter('%(asctime)-15s %(levelname)-8s%(message)s')
        filelog_handler.setFormatter(filelog_formatter)
        # create a console handler
        console_handler = logging.StreamHandler()
        console_handler.setLevel(config.log_level)
        ch_formatter = logging.Formatter('%(levelname)-8s%(message)s')
        console_handler.setFormatter(ch_formatter)

        # logging: set log file name, format, and level
        logger.addHandler(filelog_handler)
        logger.addHandler(console_handler)

        # Suppress informational logging from requests module
        logging.getLogger("requests").setLevel(logging.WARNING)
        logging.getLogger("urllib3").setLevel(logging.WARNING)
        logger.propagate = False
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
    def set_search_area(self):
        try:
            conn = self.config.connect()
            cur = conn.cursor()
            cur.execute("""
                select sa.search_area_id, sa.name
                from search_area sa join survey s
                on sa.search_area_id = s.search_area_id
                where s.survey_id = %s""", (self.survey_id,))
            (self.search_area_id, self.search_area_name) = cur.fetchone()
            cur.close()
        except (KeyboardInterrupt, SystemExit):
            cur.close()
            raise
        except Exception:
            cur.close()
            logger.error("No search area for survey_id " + str(self.survey_id))
            raise

    def update_survey_entry(self, search_by):
        try:
            survey_info = (date.today(),
                           search_by,
                           self.survey_id, )
            sql = """
            update survey
            set survey_date = %s, survey_method = %s
            where survey_id = %s
            """
            conn = self.config.connect()
            cur = conn.cursor()
            cur.execute(sql, survey_info)
            return True
        except psycopg2.Error as pge:
            logger.error(pge.pgerror)
            cur.close()
            conn.rollback()
            return False

    def listing_from_search_page_json(self, result, room_id, room_type):
        try:
            listing = ABListing(self.config, room_id, self.survey_id, room_type)
            # listing
            json_listing = result["listing"]
            listing.host_id = json_listing["primary_host"]["id"] if "primary_host" in json_listing
else None
            listing.address = json_listing["public_address"] if "public_address" in json_listing
else None
            listing.reviews = json_listing["reviews_count"] if "reviews_count" in json_listing else
None
            listing.overall_satisfaction = json_listing["star_rating"] if "star_rating" in json_listing
else None
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
            listing.accommodates = json_listing["person_capacity"] if "person_capacity" in
json_listing else None
            listing.bedrooms = json_listing["bedrooms"] if "bedrooms" in json_listing else None
            listing.latitude = json_listing["lat"] if "lat" in json_listing else None
            listing.longitude = json_listing["lng"] if "lng" in json_listing else None
            listing.coworker_hosted = json_listing["coworker_hosted"] if "coworker_hosted" in
json_listing else None
            listing.extra_host_languages = json_listing["extra_host_languages"] \
                if "extra_host_languages" in json_listing else None
            listing.name = json_listing["name"] if "name" in json_listing else None
            listing.property_type = json_listing["property_type"] if "property_type" in json_listing
else None
            # pricing
            json_pricing = result["pricing_quote"]
            listing.price = json_pricing["rate"]["amount"] if "rate" in json_pricing else None
            listing.currency = json_pricing["rate"]["currency"] if "rate" in json_pricing else None
            listing.rate_type = json_pricing["rate_type"] if "rate_type" in json_pricing else None
            return listing
        except:
            logger.exception("Error in survey.listing_from_search_page_json: returning None")
            sys.exit(-1)
            return None

    def log_progress(self, room_type, neighborhood_id,
                     guests, page_number, has_rooms):
        """ Add an entry to the survey_progress_log table to record the fact
        that a page has been visited.
        This does not apply to search by bounding box, but does apply to both
        neighborhood and zipcode searches, which is why it is in ABSurvey.
        """
        try:
            page_info = (self.survey_id, room_type, neighborhood_id,
                         guests, page_number, has_rooms)
            logger.debug("Search page: " + str(page_info))
            sql = """
            insert into survey_progress_log
            (survey_id, room_type, neighborhood_id,
            guests, page_number, has_rooms)
            values (%s, %s, %s, %s, %s, %s)
            """
            conn = self.config.connect()
            cur = conn.cursor()
            cur.execute(sql, page_info)
            cur.close()
            conn.commit()
            logger.debug("Logging survey search page for neighborhood " +
                         str(neighborhood_id))
            return True
        except psycopg2.Error as pge:
            logger.error(pge.pgerror)
            cur.close()
            conn.rollback()
            return False
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
    except Exception:
        logger.error("Save survey search page failed")
        return False

def fini(self):
    """ Wrap up a survey: correcting status and survey_date
    """
    try:
        logger.info("Finishing survey {survey_id}, for {search_area_name}".format(
            survey_id=self.survey_id, search_area_name=self.search_area_name
        ))
        sql_update = """
        update survey
        set survey_date = (
        select min(last_modified)
        from room
        where room.survey_id = survey.survey_id
        ), status = 1
        where survey_id = %s
        """
        conn = self.config.connect()
        cur = conn.cursor()
        cur.execute(sql_update, (self.survey_id, ))
        cur.close()
        conn.commit()
        return True
    except:
        logger.exception("Survey fini failed")
        return False

def page_has_been_retrieved(self, room_type, neighborhood_or_zipcode,
                            guests, page_number, search_by):
    """
    Used with neighborhood and zipcode logging (see method above).
    Returns 1 if the page has been retrieved previously and has rooms
    Returns 0 if the page has been retrieved previously and has no rooms
    Returns -1 if the page has not been retrieved previously
    """
    conn = self.config.connect()
    cur = conn.cursor()
    has_rooms = 0
    try:
        if search_by == self.config.SEARCH_BY_NEIGHBORHOOD:
            neighborhood = neighborhood_or_zipcode
            # TODO: Currently fails when there are no neighborhoods
            if neighborhood is None:
                has_rooms = -1
            else:
                params = (self.survey_id, room_type, neighborhood, guests,
                        page_number,)
                logger.debug("Params: " + str(params))
                sql = """
                select spl.has_rooms
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
                    from survey_progress_log spl
                    join neighborhood nb
                    on spl.neighborhood_id = nb.neighborhood_id
                    where survey_id = %s
                    and room_type = %s
                    and nb.name = %s
                    and guests = %s
                    and page_number = %s"""
                    cur.execute(sql, params)
                    has_rooms = cur.fetchone()[0]
                    logger.debug("has_rooms = " + str(has_rooms) +
                                 " for neighborhood " + neighborhood)
            else:  # SEARCH_BY_ZIPCODE
                zipcode = int(neighborhood_or_zipcode)
                params = (self.survey_id, room_type, zipcode, guests, page_number,)
                logger.debug(params)
                sql = """
                    select spl.has_rooms
                    from survey_progress_log spl
                    where survey_id = %s
                    and room_type = %s
                    and neighborhood_id = %s
                    and guests = %s
                    and page_number = %s"""
                cur.execute(sql, params)
                has_rooms = cur.fetchone()[0]
                logger.debug("has_rooms = " + str(has_rooms) +
                             " for zipcode " + str(zipcode))
        except Exception:
            has_rooms = -1
            logger.debug("Page has not been retrieved previously")
        finally:
            cur.close()
            return has_rooms


class ABSurveyByBoundingBox(ABSurvey):
    """
    Subclass of Survey that carries out a survey by a quadtree of bounding
    boxes: recursively searching rectangles.
    """

    def __init__(self, config, survey_id):
        super().__init__(config, survey_id)
        self.get_logged_progress()
        self.get_bounding_box()

    def get_logged_progress(self):
        try:
            sql = """
            select room_type, guests, price_min, price_max,
            quadtree_node, median_node
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
            from survey_progress_log_bb
            where survey_id = %s
            """
            conn = self.config.connect()
            cur = conn.cursor()
            cur.execute(sql, (self.survey_id,))
            row = cur.fetchone()
            cur.close()
            conn.commit()
            if row is None:
                logger.info("No progress logged for survey {}".format(self.survey_id))
                self.logged_progress = None
            else:
                logged_progress = {}
                logged_progress["room_type"] = row[0]
                logged_progress["guests"] = row[1]
                logged_progress["price_range"] = [row[2], row[3]]
                logged_progress["quadtree"] = eval(row[4])
                logged_progress["median"] = eval(row[5])
                logger.info( """Retrieved logged progress: {rt}, {g} guests, price
{pmin}-{pmax}""".
                format(rt = logged_progress["room_type"],
                    g=logged_progress["guests"],
                    pmin=logged_progress["price_range"][0],
                    pmax=logged_progress["price_range"][1]))
                logger.info("\tquadtree node {quadtree}"
                        .format(quadtree=repr(logged_progress["quadtree"])))
                logger.info("\tmedian node {median}"
                        .format(median=repr(logged_progress["median"])))
                self.logged_progress = logged_progress
        except Exception:
            logger.exception("Exception in get_progress: setting logged progress to None")
            self.logged_progress = None

    def get_bounding_box(self):
        try:
            # Get the bounding box
            conn = self.config.connect()
            cur = conn.cursor()
            cur.execute("""
                        select bb_n_lat, bb_e_lng, bb_s_lat, bb_w_lng
                        from search_area sa join survey s
                        on sa.search_area_id = s.search_area_id
                        where s.survey_id = %s""", (self.survey_id,))
            # result comes back as a tuple. We want it mutable later, so
            # convert to a list [n_lat, e_lng, s_lat, w_lng]
            self.bounding_box = list(cur.fetchone())
            cur.close()
            # Validate the bounding box
            if None in self.bounding_box:
                logger.error("Invalid bounding box: contains 'None'")
                return
            if self.bounding_box[0] <= self.bounding_box[2]:
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
                logger.error("Invalid bounding box: n_lat must be > s_lat")
                return
            if self.bounding_box[1] <= self.bounding_box[3]:
                logger.error("Invalid bounding box: e_lng must be > w_lng")
                return
            logger.info("Bounding box: " + str(self.bounding_box))
        except Exception:
            logger.exception("Exception in set_bounding_box")
            self.bounding_box = None

    def search(self, flag):
        """
        Initialize bounding box search.
        A bounding box is a rectangle around a city, specified in the
        search_area table. The loop goes to quadrants of the bounding box
        rectangle and, if new listings are found, breaks that rectangle
        into four quadrants and tries again, recursively.
        The rectangles, including the bounding box, are represented by
        [n_lat, e_lng, s_lat, w_lng], because Airbnb uses the SW and NE
        corners of the box.
        """
        try:
            logger.info("=" * 70)
            logger.info("Survey {survey_id}, for {search_area_name}".format(
                survey_id=self.survey_id, search_area_name=self.search_area_name
            ))
            ABSurvey.update_survey_entry(self, self.config.SEARCH_BY_BOUNDING_BOX)
            logger.info("Searching by bounding box, max_zoom={max_zoom}"
                    .format(max_zoom=self.config.SEARCH_MAX_RECTANGLE_ZOOM ))
            # Initialize search parameters
            # quadtree_node holds the quadtree: each rectangle is
            # divided into 00 | 01 | 10 | 11, and the next level down adds
            # on another rectangle.
            price_increments = [0, 40, 60, 80, 100, 120,
                                140, 180, 200, 300, 500,
                                700, 1000, 1500, 50000]
            max_price = {"Private room": 500,
                         "Entire home/apt": 100000,
                         "Shared room": 500}
            # set starting point
            guests_start = 1
            quadtree_node = [] # list of [0,0] etc coordinates
            median_node = [] # median lat, long to define optimal quadrants
            room_types_start_index = 0
            price_start_index = 0
            # set starting point (for survey being resumed)
            if self.logged_progress is not None:
                room_types_start_index                                    =
self.room_types.index(self.logged_progress["room_type"])
                guests_start = self.logged_progress["guests"]
                price_start_index = price_increments.index(self.logged_progress["price_range"][0])
                logger.info("""Restarting   survey   {survey_id}   at   {room_type},   {guests}   guests,
price={price}"""
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
                        .format(survey_id=self.survey_id,
room_type=self.room_types[room_types_start_index],
                        guests=guests_start, price=price_increments[price_start_index]))
            # Starting point set: loop over room types
            for room_type in self.room_types[room_types_start_index:]:
                if room_type in ("Private room", "Shared room"):
                    max_guests = 4
                else:
                    max_guests = self.config.SEARCH_MAX_GUESTS
                # loop over guests
                for guests in range(guests_start, max_guests):
                    # loop over price ranges
                    for i in range(price_start_index, len(price_increments) - 1):
                        price_range = [price_increments[i], price_increments[i+1]]
                        if price_range[1] > max_price[room_type]:
                            continue
                        self.recurse_quadtree(
                            room_type, guests, price_range, quadtree_node,
                            median_node, flag)
                        # reset starting price
                        price_start_index = 0
                # reset the starting point so that (in the event of a resumed
                # survey) the next room type gets all guest counts.
                guests_start = 1
            self.fini()
        except (SystemExit, KeyboardInterrupt):
            raise
        except Exception:
            logger.exception("Error")

    def recurse_quadtree(self, room_type, guests, price_range, quadtree_node,
            median_node, flag):
        """
        Recursive function to search for listings inside a rectangle.
        The actual search calls are done in search_node, and
        this method prints output and sets up new rectangles, if necessary,
        for another round of searching.

        To match Airbnb's use of SW and NE corners, quadrants are divided
        like this:

                    [0,1] (NW)   |   [0,0] (NE)
                    ---------------------------
                    [1,1] (SW)   |   [1,0] (SE)

        The quadrants are searched in the order [0,0], [0,1], [1,0], [1,1]
        """
        try:
            if self.subtree_previously_completed(quadtree_node):
                # go to the next subtree
                #TODO: use the same technique as the loop, below
                if quadtree_node[-1] == [0,0]:
                    quadtree_node[-1] = [0,1]
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
                elif quadtree_node[-1] == [0,1]:
                    quadtree_node[-1] = [1,0]
                elif quadtree_node[-1] == [1,0]:
                    quadtree_node[-1] = [1,1]
                elif quadtree_node[-1] == [1,1]:
                    del quadtree_node[-1]
                return

            # Only search this node if it has not been previously searched
            if (self.logged_progress is None or
                len(quadtree_node) >= len(self.logged_progress["quadtree"])):
                (new_rooms, page_count, median_leaf) = self.search_node(room_type, guests,
price_range,
                                                                 quadtree_node,
                                                                 median_node, flag)
                # we are off and searching: set logged_progress to None so
                # future guests, prices etc don't get truncated
                self.logged_progress = None
            else:
                median_leaf = self.logged_progress["median"][-1]
                logger.debug("Node                    previously                    searched:
{quadtree}".format(quadtree=quadtree_node))
                # if the logged_progress has more depth, recurse
                if len(self.logged_progress["quadtree"]) >= len(quadtree_node):
                    page_count = self.config.SEARCH_MAX_PAGES
            # The max zoom is set in config, but decrease it by one for each guest
            # so that high guest counts don't zoom in (which turns out to generate
            # very few new rooms but take a lot of time)
            zoomable          =          len(quadtree_node)          <          max(1,
(self.config.SEARCH_MAX_RECTANGLE_ZOOM - 2 * (guests - 1)))
            # zoomable = len(quadtree_node) < self.config.SEARCH_MAX_RECTANGLE_ZOOM
            # If (new_rooms > 0 or page_count == self.config.SEARCH_MAX_PAGES) and
zoomable:
            # zoom in if the search returned a full set of SEARCH_MAX_PAGES pages even
            # if no rooms were new, as there may still be new rooms that show up at
            # higher zoom levels.
            if page_count == self.config.SEARCH_MAX_PAGES and zoomable:
                # append a node to the quadtree for a new level
                quadtree_node.append([0,0])
                median_node.append(median_leaf)
                for int_leaf in range(4):
                    # append a node to the quadtree for a new level
                    quadtree_leaf = [int(i)
                            for i in str(bin(int_leaf))[2:].zfill(2)]
                    quadtree_node[-1] = quadtree_leaf
                    new_rooms = self.recurse_quadtree(room_type, guests, price_range,
                                                quadtree_node, median_node, flag)
                # the search of the quadtree below this node is complete:
                # remove the leaf element from the tree and return to go up a level
                del quadtree_node[-1]
                del median_node[-1]
            logger.debug("Returning from recurse_quadtree for {}".format(quadtree_node))
            if flag == self.config.FLAGS_PRINT:
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
                # for FLAGS_PRINT, fetch one page and print it
                sys.exit(0)
        except (SystemExit, KeyboardInterrupt):
            raise
        except TypeError as te:
            logger.exception("TypeError in recurse_quadtree")
            logger.error(te.args)
            raise
        except:
            logger.exception("Error in recurse_quadtree")
            raise

    def search_node(self, room_type, guests, price_range, quadtree_node,
            median_node, flag):
        """
            rectangle is (n_lat, e_lng, s_lat, w_lng)
            returns number of *new* rooms and number of pages tested
        """
        try:
            logger.info("-" * 70)
            rectangle = self.get_rectangle_from_quadtree_node(quadtree_node, median_node)
            logger.info(
                ("Searching rectangle: {room_type}, guests = {guests}, "
                 "prices in [{p1}, {p2}], zoom factor = {z}")
                .format(room_type=room_type, guests=guests,
                    p1=price_range[0], p2=price_range[1], z=len(quadtree_node))
            )
            logger.debug("quadtree_node                                                    =
{quadtree_node}".format(quadtree_node=str(quadtree_node)))
            logger.debug("Rectangle: N={n:+.5f}, E={e:+.5f}, S={s:+.5f}, W={w:+.5f}".format(
                n=rectangle[0], e=rectangle[1], s=rectangle[2], w=rectangle[3])
            )
            new_rooms = 0
            room_total = 0
            # median_lists are collected from results on each page and used to
            # calculate the median values, which will be used to divide the
            # volume into optimal "quadrants".
            median_lists = {}
            median_lists["latitude"] = []
            median_lists["longitude"] = []
            for page_number in range(1, self.config.SEARCH_MAX_PAGES + 1):
                room_count = 0
                # set up the parameters for the request
                params = {}
                params["guests"] = str(guests)
                params["page"] = str(page_number)
                params["source"] = "filter"
                params["room_types[]"] = room_type
                params["sw_lat"] = str(rectangle[2])
                params["sw_lng"] = str(rectangle[3])
                params["ne_lat"] = str(rectangle[0])
                params["ne_lng"] = str(rectangle[1])
                params["search_by_map"] = str(True)
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
                params["price_min"] = str(price_range[0])
                params["price_max"] = str(price_range[1])
                # make the http request
                response              =              airbnb_ws.ws_request_with_repeats(self.config,
self.config.URL_API_SEARCH_ROOT, params)
                # process the response
                if response is None:
                    logger.warning("No   response   received   from   request   despite   multiple
attempts: {p}"
                            .format(p=params))
                    continue
                json = response.json()
                for result in json["results_json"]["search_results"]:
                    room_id = int(result["listing"]["id"])
                    if room_id is not None:
                        room_count += 1
                        room_total += 1
                        listing = self.listing_from_search_page_json(result, room_id, room_type)
                        median_lists["latitude"].append(listing.latitude)
                        median_lists["longitude"].append(listing.longitude)
                        if listing is None:
                            continue
                        if listing.host_id is not None:
                            listing.deleted = 0
                            if flag == self.config.FLAGS_ADD:
                                if listing.save(self.config.FLAGS_INSERT_NO_REPLACE):
                                    new_rooms += 1
                            elif flag == self.config.FLAGS_PRINT:
                                print(room_type, listing.room_id)
                # Log page-level results
                logger.info("Page {page_number:02d} returned {room_count:02d} listings"
                        .format(page_number=page_number, room_count=room_count))
                if flag == self.config.FLAGS_PRINT:
                    # for FLAGS_PRINT, fetch one page and print it
                    sys.exit(0)
                if room_count < self.config.SEARCH_LISTINGS_ON_FULL_PAGE:
                    # If a full page of listings is not returned by Airbnb,
                    # this branch of the search is complete.
                    logger.debug("Final page of listings for this search")
                    break
            # Log rectangle-level results
            logger.info(("Results:  {page_count} pages, {new_rooms} new rooms, "
                "{room_type}, {g} guests, prices in [{p1}, {p2}]").format(
                            room_type=room_type, g=str(guests),
                            p1=str(price_range[0]),
                            p2=str(price_range[1]),
                            new_rooms=str(new_rooms),
                            page_count=str(page_number)))
            if len(median_node) == 0:
                median_leaf = "[]"
            else:
                median_leaf = median_node[-1]
            logger.info("Results: rect = {median_leaf}, node = {quadtree_node}"
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
                    .format(quadtree_node=str(quadtree_node), median_leaf=str(median_leaf)))
            # calculate medians
            if room_count > 0:
                median_lat = sorted(median_lists["latitude"])[int(len(median_lists["latitude"])/2)]
                median_lng                                                                          =
sorted(median_lists["longitude"])[int(len(median_lists["longitude"])/2)]
                median_leaf = [median_lat, median_lng]
            else:
                # values not needed, but we need to fill in an item anyway
                median_leaf = [0, 0]
            # log progress
            self.log_progress(room_type, guests, price_range[0],
                    price_range[1], quadtree_node, median_node)
            return (new_rooms, page_number, median_leaf)
        except UnicodeEncodeError:
            logger.error("UnicodeEncodeError: set PYTHONIOENCODING=utf-8")
            # if sys.version_info >= (3,):
            #     logger.info(s.encode('utf8').decode(sys.stdout.encoding))
            # else:
            #     logger.info(s.encode('utf8'))
            # unhandled at the moment
        except Exception:
            logger.exception("Exception in get_search_page_info_rectangle")
            raise

    def get_rectangle_from_quadtree_node(self, quadtree_node, median_node):
        try:
            rectangle = self.bounding_box[0:4]
            for node, medians in zip(quadtree_node, median_node):
                logger.debug("Quadtrees: {q}".format(q=node))
                logger.debug("Medians: {m}".format(m=medians))
                [n_lat, e_lng, s_lat, w_lng] = rectangle
                blur = abs(n_lat - s_lat) * self.config.SEARCH_RECTANGLE_EDGE_BLUR
                # find the mindpoints of the rectangle
                mid_lat = (n_lat + s_lat)/2.0
                mid_lng = (e_lng + w_lng)/2.0
                # mid_lat = medians[0]
                # mid_lng = medians[1]
                # overlap quadrants to ensure coverage at high zoom levels
                # Airbnb max zoom (18) is about 0.004 on a side.
                rectangle = []
                if node==[0,0]: # NE
                    rectangle = [n_lat + blur, e_lng + blur, mid_lat - blur, mid_lng - blur]
                elif node==[0,1]: # NW
                    rectangle = [n_lat + blur, mid_lng + blur, mid_lat - blur, w_lng - blur]
                elif node==[1,0]: # SE
                    rectangle = [mid_lat + blur, e_lng + blur, s_lat - blur, mid_lng - blur]
                elif node==[1,1]: # SW
                    rectangle = [mid_lat + blur, mid_lng + blur, s_lat - blur, w_lng - blur]
            logger.debug("Rectangle calculated: {rect}".format(rect=rectangle))
            return rectangle
        except:
            logger.exception("Exception in get_rectangle_from_quadtree_node")
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
            return None

    def subtree_previously_completed(self, quadtree_node):
        # Return if the child subtree of this node was completed
        # in a previous survey
        subtree_previously_completed = False
        if len(quadtree_node) > 0 and self.logged_progress is not None:
            s_this_quadrant = ''.join(str(quadtree_node[i][j])
                    for j in range(0,2)
                    for i in range(0,len(quadtree_node)))
            s_logged_progress = ''.join(str(self.logged_progress["quadtree"][i][j])
                    for j in range(0,2)
                    for i in range(0,len(quadtree_node)))
            if int(s_this_quadrant) < int(s_logged_progress):
                subtree_previously_completed = True
                logger.debug("Subtree                    previously                    completed:
{quadtree}".format(quadtree=quadtree_node))
        return subtree_previously_completed


    def log_progress(self, room_type, guests, price_min, price_max,
            quadtree_node, median_node):
        try:
            # This upsert statement requires PostgreSQL 9.5
            # Convert the quadrant to a string with repr() before storing it
            sql = """
            insert into survey_progress_log_bb
            (survey_id, room_type, guests, price_min, price_max, quadtree_node,
            median_node)
            values
            (%s, %s, %s, %s, %s, %s, %s)
            on conflict ON CONSTRAINT survey_progress_log_bb_pkey
            do update
                set room_type = %s
                , guests = %s
                , price_min = %s
                , price_max = %s
                , quadtree_node = %s
                , median_node = %s
                , last_modified = now()
            where survey_progress_log_bb.survey_id = %s
            """
            conn = self.config.connect()
            cur = conn.cursor()
            cur.execute(sql, (self.survey_id, room_type,
                guests, price_min, price_max, repr(quadtree_node),
                repr(median_node),
                room_type, guests, price_min, price_max,
                repr(quadtree_node), repr(median_node),
                self.survey_id))
            cur.close()
            conn.commit()
            logger.debug("Progress logged")
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
            return True
        except Exception as e:
            logger.warning("""Progress not logged: survey not affected, but
                    resume will not be available if survey is truncated.""")
            logger.exception("Exception in  log_progress: {e}".format(e=type(e)))
            conn.close()
            return False



class ABSurveyByNeighborhood(ABSurvey):
    """
    Subclass of Survey that carries out a survey by looping over
    the neighborhoods as defined on the Airbnb web site.
    """

    def search(self, flag):
        logger.info("=" * 70)
        logger.info("Survey {survey_id}, for {search_area_name}".format(
            survey_id=self.survey_id, search_area_name=self.search_area_name
        ))
        ABSurvey.update_survey_entry(self, self.config.SEARCH_BY_NEIGHBORHOOD)
        if self.search_area_name == self.config.SEARCH_AREA_GLOBAL:
            # "Special case": global search
            self.__global_search()
        else:
            logger.info("Searching by neighborhood")
            neighborhoods = self.get_neighborhoods_from_search_area()
            # for some cities (eg Havana) the neighborhood information
            # is incomplete, and an additional search with no
            # neighborhood is useful
            neighborhoods = neighborhoods + [None]
            for room_type in self.room_types:
                logger.debug(
                    "Searching for %(rt)s by neighborhood",
                    {"rt": room_type})
                if len(neighborhoods) > 0:
                    self.__search_loop_neighborhoods(neighborhoods,
                                                     room_type, flag)
                else:
                    self.__search_neighborhood(None, room_type, flag)
        self.fini()

    def __search_loop_neighborhoods(self, neighborhoods, room_type, flag):
        """Loop over neighborhoods in a city. No return."""
        try:
            for neighborhood in neighborhoods:
                self.__search_neighborhood(neighborhood, room_type, flag)
        except Exception:
            raise

    def __search_neighborhood(self, neighborhood, room_type, flag):
        try:
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
            if room_type in ("Private room", "Shared room"):
                max_guests = 4
            else:
                max_guests = self.config.SEARCH_MAX_GUESTS
            for guests in range(1, max_guests):
                logger.debug("Searching for %(g)i guests", {"g": guests})
                for page_number in range(1, self.config.SEARCH_MAX_PAGES + 1):
                    if flag != self.config.FLAGS_PRINT:
                        count = self.page_has_been_retrieved(
                            room_type, neighborhood, guests, page_number,
                            self.config.SEARCH_BY_NEIGHBORHOOD)
                        if count == 1:
                            logger.info(
                                "\t...search page has been visited previously")
                            continue
                        elif count == 0:
                            logger.info(
                                "\t...search page has been visited previously")
                            break
                        else:
                            pass
                    room_count = self.__search_neighborhood_page(
                        room_type, neighborhood, guests, page_number, flag)
                    logger.info(("{room_type} ({g} guests): neighborhood {neighborhood}: "
                                 "{room_count} rooms, {page_number} pages").format(
                                     room_type=room_type, g=str(guests),
                                     neighborhood=neighborhood,
                                     room_count=room_count,
                                     page_number=str(page_number)))
                    if flag == self.config.FLAGS_PRINT:
                        # for FLAGS_PRINT, fetch one page and print it
                        sys.exit(0)
                    if room_count < self.config.SEARCH_LISTINGS_ON_FULL_PAGE:
                        logger.debug("Final page of listings for this search")
                        break
        except Exception:
            raise

    def __search_neighborhood_page(self, room_type, neighborhood, guests, page_number, flag):
        try:
            logger.info("-" * 70)
            logger.info(room_type + ", " +
                str(neighborhood) + ", " +
                str(guests) + " guests, " +
                "page " + str(page_number))
            new_rooms = 0
            room_count = 0
            params = {}
            params["page"] = str(page_number)
            params["source"] = "filter"
            params["location"] = self.search_area_name
            params["room_types[]"] = room_type
            params["neighborhoods[]"] = neighborhood
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
            response                    =                    airbnb_ws.ws_request_with_repeats(self.config,
self.config.URL_API_SEARCH_ROOT, params)
            json = response.json()
            for result in json["results_json"]["search_results"]:
                room_id = int(result["listing"]["id"])
                if room_id is not None:
                    room_count += 1
                    listing = self.listing_from_search_page_json(result, room_id, room_type)
                    if listing is None:
                        continue
                    if listing.host_id is not None:
                        listing.deleted = 0
                        if flag == self.config.FLAGS_ADD:
                            if listing.save(self.config.FLAGS_INSERT_NO_REPLACE):
                                new_rooms += 1
                        elif flag == self.config.FLAGS_PRINT:
                            print(room_type, listing.room_id)
            if room_count > 0:
                has_rooms = 1
            else:
                has_rooms = 0
            if flag == self.config.FLAGS_ADD:
                neighborhood_id = self.get_neighborhood_id(neighborhood)
                self.log_progress(room_type, neighborhood_id,
                                  guests, page_number, has_rooms)
            return room_count
        except UnicodeEncodeError:
            logger.error("UnicodeEncodeError: set PYTHONIOENCODING=utf-8")
            # if sys.version_info >= (3,):
            #     logger.info(s.encode('utf8').decode(sys.stdout.encoding))
            # else:
            #     logger.info(s.encode('utf8'))
            # unhandled at the moment
        except Exception:
            raise

    def get_neighborhood_id(self, neighborhood):
        try:
            sql = """
            select neighborhood_id
            from neighborhood nb,
                search_area sa,
                survey s
            where nb.search_area_id = sa.search_area_id
            and sa.search_area_id = s.search_area_id
            and s.survey_id = %s
            and nb.name = %s
            """
            conn = self.config.connect()
            cur = conn.cursor()
            cur.execute(sql, (self.survey_id, neighborhood,))
            neighborhood_id = cur.fetchone()[0]
            cur.close()
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
            conn.commit()
            cur = conn.cursor()
            cur.execute(sql, (self.survey_id, neighborhood,))
            neighborhood_id = cur.fetchone()[0]
            cur.close()
            conn.commit()
            return neighborhood_id
        except psycopg2.Error:
            raise
        except Exception:
            return None

    def get_neighborhoods_from_search_area(self):
        try:
            conn = self.config.connect()
            cur = conn.cursor()
            cur.execute("""
                select name
                from neighborhood
                where search_area_id =  %s
                order by name""", (self.search_area_id,))
            neighborhoods = []
            while True:
                row = cur.fetchone()
                if row is None:
                    break
                neighborhoods.append(row[0])
            cur.close()
            return neighborhoods
        except Exception:
            logger.error("Failed to retrieve neighborhoods from " +
                         str(search_area_id))
            raise


class ABSurveyByZipcode(ABSurvey):
    """
    Subclass of Survey that carries out a survey by looping over
    zipcodes as defined in a separate table
    """

    def search(self, flag):
        logger.info("=" * 70)
        logger.info("Survey {survey_id}, for {search_area_name}".format(
            survey_id=self.survey_id, search_area_name=self.search_area_name
        ))
        ABSurvey.update_survey_entry(self, self.config.SEARCH_BY_ZIPCODE)
        logger.info("Searching by zipcode")
        zipcodes = self.get_zipcodes_from_search_area()
        for room_type in self.room_types:
            try:
                i = 0
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
                for zipcode in zipcodes:
                    i += 1
                    self.__search_zipcode(str(zipcode), room_type, self.survey_id,
                                          flag, self.search_area_name)
            except Exception:
                raise
        self.fini()

    def __search_zipcode(self, zipcode, room_type, survey_id,
                         flag, search_area_name):
        try:
            if room_type in ("Private room", "Shared room"):
                max_guests = 4
            else:
                max_guests = self.config.SEARCH_MAX_GUESTS
            for guests in range(1, max_guests):
                logger.debug("Searching for %(g)i guests", {"g": guests})
                for page_number in range(1, self.config.SEARCH_MAX_PAGES + 1):
                    if flag != self.config.FLAGS_PRINT:
                        # this efficiency check can be implemented later
                        count = self.page_has_been_retrieved(
                            room_type, str(zipcode),
                            guests, page_number, self.config.SEARCH_BY_ZIPCODE)
                        if count == 1:
                            logger.info(
                                "\t...search page has been visited previously")
                            continue
                        elif count == 0:
                            logger.info(
                                "\t...search page has been visited previously")
                            break
                        else:
                            logger.debug("\t...visiting search page")
                    room_count = self.get_search_page_info_zipcode(
                        room_type, zipcode, guests, page_number, flag)
                    if flag == self.config.FLAGS_PRINT:
                        # for FLAGS_PRINT, fetch one page and print it
                        sys.exit(0)
                    if room_count < self.config.SEARCH_LISTINGS_ON_FULL_PAGE:
                        logger.debug("Final page of listings for this search")
                        break
        except Exception:
            raise

    def get_zipcodes_from_search_area(self):
        try:
            conn = self.config.connect()
            cur = conn.cursor()
            # Query from the manually-prepared zipcode table
            cur.execute("""
            select zipcode
            from zipcode z, search_area sa
            where sa.search_area_id = %s
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
            and z.search_area_id = sa.search_area_id
            """, (self.search_area_id,))
            zipcodes = []
            while True:
                row = cur.fetchone()
                if row is None:
                    break
                zipcodes.append(row[0])
            cur.close()
            return zipcodes
        except Exception:
            logger.error("Failed to retrieve zipcodes for search_area" +
                         str(self.search_area_id))
            raise

    def get_search_page_info_zipcode(self, room_type,
                                     zipcode, guests, page_number, flag):
        try:
            logger.info("-" * 70)
            logger.info(room_type + ", zipcode " + str(zipcode) + ", " +
                        str(guests) + " guests, " + "page " + str(page_number))
            room_count = 0
            new_rooms = 0
            params = {}
            params["guests"] = str(guests)
            params["page"] = str(page_number)
            params["source"] = "filter"
            params["location"] = zipcode
            params["room_types[]"] = room_type
            response          =          airbnb_ws.ws_request_with_repeats(self.config,
self.config.URL_API_SEARCH_ROOT, params)
            json = response.json()
            for result in json["results_json"]["search_results"]:
                room_id = int(result["listing"]["id"])
                if room_id is not None:
                    room_count += 1
                    listing = self.listing_from_search_page_json(result, room_id, room_type)
                    if listing is None:
                        continue
                    if listing.host_id is not None:
                        listing.deleted = 0
                        if flag == self.config.FLAGS_ADD:
                            if listing.save(self.config.FLAGS_INSERT_NO_REPLACE):
                                new_rooms += 1
                        elif flag == self.config.FLAGS_PRINT:
                            print(room_type, listing.room_id)
            if room_count > 0:
                has_rooms = 1
            else:
                has_rooms = 0
            if flag == self.config.FLAGS_ADD:
                self.log_progress(room_type, zipcode,
                                  guests, page_number, has_rooms)
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
            else:
                logger.info("No rooms found")
            return room_count
        except UnicodeEncodeError:
            logger.error(
                "UnicodeEncodeError: you may want to set PYTHONIOENCODING=utf-8")
            # if sys.version_info >= (3,):
            #     logger.info(s.encode('utf8').decode(sys.stdout.encoding))
            # else:
            #     logger.info(s.encode('utf8'))
            # unhandled at the moment
        except Exception as e:
            logger.error("Exception type: " + type(e).__name__)
            raise


def ABSurveyGlobal(ABSurvey):
    """
    Special search to randomly choose rooms from a range rather than to
    look at specific areas of the world.
    """

    def search(self, flag, search_by):
        logger.info("-" * 70)
        logger.info("Survey {survey_id}, for {search_area_name}".format(
            survey_id=self.survey_id, search_area_name=self.search_area_name
        ))
        ABSurvey.update_survey_entry(self, self.config.SEARCH_AREA_GLOBAL)
        room_count = 0
        while room_count < self.config.FILL_MAX_ROOM_COUNT:
            try:
                # get a random candidate room_id
                room_id = random.randint(0, self.config.ROOM_ID_UPPER_BOUND)
                listing = ABListing(self.config, room_id, self.survey_id)
                if room_id is None:
                    break
                else:
                    if listing.ws_get_room_info(self.config.FLAGS_ADD):
                        room_count += 1
            except AttributeError:
                logger.error(
                    "Attribute error: marking room as deleted.")
                listing.save_as_deleted()
            except Exception as ex:
                logger.exception("Error in search:" + str(type(ex)))
                raise
        self.fini()
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

APPENDIX F

Platform Home Sharing Survey Exporting Tool “export_spreadsheet.py”

This tool deposits the information gathered into a designated folder.

```
import psycopg2 as pg
import pandas as pd
import argparse
import datetime as dt
import logging
from airbnb_config import ABConfig

LOG_LEVEL = logging.INFO
# Set up logging
LOG_FORMAT = '%(levelname)-8s%(message)s'
logging.basicConfig(format=LOG_FORMAT, level=LOG_LEVEL)
DEFAULT_START_DATE = '2013-05-02'


def survey_df(ab_config, city, start_date):
    sql_survey_ids = """
        select survey_id, survey_date, comment
        from survey s, search_area sa
        where s.search_area_id = sa.search_area_id
        and sa.name = %(city)s
        and s.survey_date > '{start_date}'
        and s.status = 1
        order by survey_id
    """.format(start_date=start_date)
    conn = ab_config.connect()
    df = pd.read_sql(sql_survey_ids, conn,
                    params={"city": city})
    conn.close()
    return(df)


def city_view_name(ab_config, city):
    sql_abbrev = """
    select abbreviation from search_area
    where name = %s
    """
    conn = ab_config.connect()
    cur = conn.cursor()
    cur.execute(sql_abbrev, (city, ))
    city_view_name = 'listing_' + cur.fetchall()[0][0]
    cur.close()
    return city_view_name


def total_listings(ab_config, city_view):
    sql = """select s.survey_id "Survey",
    survey_date "Date", count(*) "Listings"
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
    from {city_view} r join survey s
    on r.survey_id = s.survey_id
    group by 1, 2
    order by 1
    """.format(city_view=city_view)
    conn = ab_config.connect()
    df = pd.read_sql(sql, conn)
    conn.close()
    return df


def by_room_type(ab_config, city_view):
    sql = """select s.survey_id "Survey",
    survey_date "Date", room_type "Room Type",
    count(*) "Listings", sum(reviews) "Reviews",
    sum(reviews * price) "Relative Income"
    from {city_view} r join survey s
    on r.survey_id = s.survey_id
    where room_type is not null
    group by 1, 2, 3
    order by 1
    """.format(city_view=city_view)
    conn = ab_config.connect()
    df = pd.read_sql(sql, conn)
    conn.close()
    return df.pivot(index="Date", columns="Room Type")


def by_host_type(ab_config, city_view):
    sql = """
    select survey_id "Survey",
        survey_date "Date",
        case when listings_for_host = 1
        then 'Single' else 'Multi'
        end "Host Type",
        sum(hosts) "Hosts", sum(listings) "Listings", sum(reviews) "Reviews"
    from (
        select survey_id, survey_date,
        listings_for_host, count(*) hosts,
        sum(listings_for_host) listings, sum(reviews) reviews
        from  (
            select s.survey_id survey_id, survey_date,
            host_id, count(*) listings_for_host,
            sum(reviews) reviews
            from {city_view} r join survey s
            on r.survey_id = s.survey_id
            group by s.survey_id, survey_date, host_id
            ) T1
        group by 1, 2, 3
    ) T2
    group by 1, 2, 3
    """.format(city_view=city_view)
    conn = ab_config.connect()
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
    df = pd.read_sql(sql, conn)
    conn.close()
    df = df.pivot(index="Date", columns="Host Type")
    # df.set_index(["Date"], drop=False, inplace=True)
    return df


def by_neighborhood(ab_config, city_view):
    sql = """select
        s.survey_id, survey_date "Date", neighborhood "Neighborhood",
        count(*) "Listings", sum(reviews) "Reviews"
    from {city_view} r join survey s
    on r.survey_id = s.survey_id
    group by 1, 2, 3
    """.format(city_view=city_view)
    conn = ab_config.connect()
    df = pd.read_sql(sql, conn)
    conn.close()
    df = df.pivot(index="Date", columns="Neighborhood")
    # df.set_index(["Date"], drop=False, inplace=True)
    return df


def export_city_summary(ab_config, city, project, start_date):
    logging.info(" ---- Exporting summary spreadsheet" +
                 " for " + city +
                 " using project " + project)
    city_bar = city.replace(" ", "_").lower()
    today = dt.date.today().isoformat()
    xlsxfile = ("./{project}/slee_{project}_{city_bar}_summary_{today}.xlsx"
                ).format(project=project, city_bar=city_bar, today=today)
    writer = pd.ExcelWriter(xlsxfile, engine="xlsxwriter")
    df = survey_df(ab_config, city, start_date)
    city_view = city_view_name(ab_config, city)
    logging.info("Total listings...")
    df = total_listings(ab_config, city_view)
    df.to_excel(writer, sheet_name="Total Listings", index=False)
    logging.info("Listings by room type...")
    df = by_room_type(ab_config, city_view)
    df["Listings"].to_excel(writer,
                            sheet_name="Listings by room type", index=True)
    df["Reviews"].to_excel(writer,
                           sheet_name="Reviews by room type", index=True)
    logging.info("Listings by host type...")
    df = by_host_type(ab_config, city_view)
    df["Hosts"].to_excel(writer,
                         sheet_name="Hosts by host type", index=True)
    df["Listings"].to_excel(writer,
                            sheet_name="Listings by host type", index=True)
    df["Reviews"].to_excel(writer,
                           sheet_name="Reviews by host type", index=True)
    logging.info("Listings by neighborhood...")
    df = by_neighborhood(ab_config, city_view)
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
    df["Listings"].to_excel(writer,
                            sheet_name="Listings by Neighborhood", index=True)
    df["Reviews"].to_excel(writer,
                           sheet_name="Reviews by Neighborhood", index=True)
    logging.info("Saving " + xlsxfile)
    writer.save()


def export_city_data(ab_config, city, project, format, start_date):
    logging.info(" ---- Exporting " + format +
                 " for " + city +
                 " using project " + project)

    df = survey_df(ab_config, city, start_date)
    survey_ids = df["survey_id"].tolist()
    survey_dates = df["survey_date"].tolist()
    logging.info(" ---- Surveys: " + ', '.join(str(id) for id in survey_ids))
    conn = ab_config.connect()

    # survey_ids = [11, ]
    if project == "gis":
        city_view = city_view_name(ab_config, city)
        sql = """
        select room_id, host_id, room_type,
            borough, neighborhood,
            reviews, overall_satisfaction,
            accommodates, bedrooms, bathrooms,
            price, minstay,
            latitude, longitude,
            last_modified as collected
        from {city_view}
        where survey_id = %(survey_id)s
        order by room_id
        """.format(city_view=city_view)
    elif project == "hvs":
        sql = """
        select room_id, host_id, room_type,
            borough, neighborhood,
            reviews, overall_satisfaction,
            accommodates, bedrooms, bathrooms,
            price, minstay,
            latitude, longitude,
            last_modified as collected
        from hvs.listing
        where survey_id = %(survey_id)s
        order by room_id
        """
    else:
        sql = """
        select room_id, host_id, room_type,
            city, neighborhood,
            reviews, overall_satisfaction,
            accommodates, bedrooms, bathrooms,
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
            price, minstay,
            latitude, longitude,
            last_modified as collected
        from survey_room(%(survey_id)s)
        order by room_id
        """

    city_bar = city.replace(" ", "_").lower()
    if format == "csv":
        for survey_id, survey_date in \
                zip(survey_ids, survey_dates):
            csvfile = ("./{project}/ts_{city_bar}_{survey_date}.csv").format(
                project=project, city_bar=city_bar,
                survey_date=str(survey_date))
            csvfile = csvfile.lower()
            df = pd.read_sql(sql, conn,
                             # index_col="room_id",
                             params={"survey_id": survey_id.item()}
                             )
            logging.info("CSV export: survey " +
                         str(survey_id) + " to " + csvfile)
            df.to_csv(csvfile)
            # default encoding is 'utf-8' on Python 3
    else:
        today = dt.date.today().isoformat()
        xlsxfile = ("./{project}/slee_{project}_{city_bar}_{today}.xlsx"
                    ).format(project=project, city_bar=city_bar, today=today)
        writer = pd.ExcelWriter(xlsxfile, engine="xlsxwriter")
        logging.info("Spreadsheet name: " + xlsxfile)
        # read surveys
        for survey_id, survey_date in \
                zip(survey_ids, survey_dates):
            logging.info("Survey " + str(survey_id) + " for " + city)
            df = pd.read_sql(sql, conn,
                             # index_col="room_id",
                             params={"survey_id": survey_id.item()}
                             )
            if len(df) > 0:
                logging.info("Survey " + str(survey_id) +
                             ": to Excel worksheet")
                df.to_excel(writer, sheet_name=str(survey_date))
            else:
                logging.info("Survey " + str(survey_id) +
                             " not in production project: ignoring")

        # neighborhood summaries
        if project == "gis":
            sql = "select to_char(survey_date, 'YYYY-MM-DD') as survey_date,"
            sql += " neighborhood, count(*) as listings from"
            sql += " " + city_view + " li,"
            sql += " survey s"
            sql += " where li.survey_id = s.survey_id"
            sql += " and s.survey_date > %(start_date)s"
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
            sql += " group by survey_date, neighborhood order by 3 desc"
            try:
                df = pd.read_sql(sql, conn, params={"start_date": start_date})
                if len(df.index) > 0:
                    logging.info("Exporting listings for " + city)
                    dfnb = df.pivot(index='neighborhood', columns='survey_date',
                                    values='listings')
                    dfnb.fillna(0)
                    dfnb.to_excel(writer, sheet_name="Listings by neighborhood")
            except pg.InternalError:
                # Miami has no neighborhoods
                pass
            except pd.io.sql.DatabaseError:
                # Miami has no neighborhoods
                pass

        sql = "select to_char(survey_date, 'YYYY-MM-DD') as survey_date,"
        sql += " neighborhood, sum(reviews) as visits from"
        sql += " " + city_view + " li,"
        sql += " survey s"
        sql += " where li.survey_id = s.survey_id"
        sql += " and s.survey_date > %(start_date)s"
        sql += " group by survey_date, neighborhood order by 3  desc"
        try:
            df = pd.read_sql(sql, conn, params={"start_date": start_date})
            if len(df.index) > 0:
                logging.info("Exporting visits for " + city)
                dfnb = df.pivot(index='neighborhood', columns='survey_date',
                                values='visits')
                dfnb.fillna(0)
                dfnb.to_excel(writer, sheet_name="Visits by neighborhood")
        except pg.InternalError:
            # Miami has no neighborhoods
            pass
        except pd.io.sql.DatabaseError:
            pass

        logging.info("Saving " + xlsxfile)
        writer.save()


def main():
    parser = \
        argparse.ArgumentParser(
            description="Create a spreadsheet of surveys from a city")
    parser.add_argument("-cfg", "--config_file",
                        metavar="config_file", action="store", default=None,
                        help="""explicitly set configuration file, instead of
                        using the default <username>.config""")
    parser.add_argument('-c', '--city',
                        metavar='city', action='store',
                        help="""set the city""")
    parser.add_argument('-p', '--project',
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
                         metavar='project', action='store', default="public",
                         help="""the project determines the table or view: public
                         for room, gis for listing_city, default public""")
    parser.add_argument('-f', '--format',
                         metavar='format', action='store', default="xlsx",
                         help="""output format (xlsx or csv), default xlsx""")
    parser.add_argument('-s', '--summary',
                         action='store_true', default=False,
                         help="create a summary spreadsheet instead of raw data")
    parser.add_argument('-sd', '--start_date',
                         metavar="start_date", action='store',
                         default=DEFAULT_START_DATE,
                         help="create a summary spreadsheet instead of raw data")
    args = parser.parse_args()
    ab_config = ABConfig(args)

    if args.city:
        if args.summary:
            export_city_summary(ab_config, args.city, args.project.lower(),
                    args.start_date)
        else:
            export_city_data(ab_config, args.city, args.project.lower(),
                    args.format, args.start_date)
    else:
        parser.print_help()


if __name__ == "__main__":
    main()
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

APPENDIX G

Platform Home Sharing Survey Schema Management Tool “schema_update.py”

This tool customizes the relationships among the collection of databases compiled through the scraping and implemented in the survey.

```
import logging
import re
from lxml import html
import psycopg2
import json
import airbnb_ws

logger = logging.getLogger()


class ABListing():
    """
    # ABListing represents an Airbnb room_id, as captured at a moment in time.
    # room_id, survey_id is the primary key.
    # Occasionally, a survey_id = None will happen, but for retrieving data
    # straight from the web site, and not stored in the database.
    """
    def __init__(self, config, room_id, survey_id, room_type=None):
        self.config = config
        self.room_id = room_id
        self.host_id = None
        self.room_type = room_type
        self.country = None
        self.city = None
        self.neighborhood = None
        self.address = None
        self.reviews = None
        self.overall_satisfaction = None
        self.accommodates = None
        self.bedrooms = None
        self.bathrooms = None
        self.price = None
        self.deleted = None
        self.minstay = None
        self.latitude = None
        self.longitude = None
        self.survey_id = survey_id
        #  extra fields added from search json:
        # coworker_hosted (bool)
        self.coworker_hosted = None
        # extra_host_languages (list)
        self.extra_host_languages = None
        # name (str)
        self.name = None
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
        # property_type (str)
        self.property_type = None
        # currency (str)
        self.currency = None
        # rate_type (str) - "nightly" or other?
        self.rate_type = None
        """ """

    def status_check(self):
        status = True  # OK
        # if sufficient of the values are None or don't exist, the room
        # entry was not properly parsed and we may as well throw the whole
        # thing away.
        unassigned_values = {key: value
                             for key, value in vars(self).items()
                             if not key.startswith('__') and
                             not callable(key) and
                             value is None
                             }
        if len(unassigned_values) > 9:  # just a value indicating deleted
            logger.info("Room " + str(self.room_id) + ": marked deleted")
            status = False  # probably deleted
            self.deleted = 1
        else:
            for key, val in unassigned_values.items():
                if (key == "overall_satisfaction" and "reviews" not in
                        unassigned_values):
                    if val is None and self.reviews > 2:
                        logger.debug("Room " + str(self.room_id) + ": No value for " + key)
                elif val is None:
                    logger.debug("Room " + str(self.room_id) + ": No value for " + key)
        return status

    def get_columns(self):
        """
        Hack: callable(attr) includes methods with (self) as argument.
        Need to find a way to avoid these.
        This hack does also provide the proper order, which matters
        """
        # columns = [attr for attr in dir(self) if not
        # callable(attr) and not attr.startswith("__")]
        columns = ("room_id", "host_id", "room_type", "country",
                   "city", "neighborhood", "address", "reviews",
                   "overall_satisfaction", "accommodates", "bedrooms",
                   "bathrooms", "price", "deleted", "minstay",
                   "latitude", "longitude", "survey_id", "last_modified",)
        return columns

    def save_as_deleted(self):
        try:
            logger.debug("Marking room deleted: " + str(self.room_id))
            if self.survey_id is None:
                return
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
            conn = self.config.connect()
            sql = """
                update room
                set deleted = 1, last_modified = now()::timestamp
                where room_id = %s
                and survey_id = %s
            """
            cur = conn.cursor()
            cur.execute(sql, (self.room_id, self.survey_id))
            cur.close()
            conn.commit()
        except Exception:
            logger.error("Failed to save room as deleted")
            raise

    def save(self, insert_replace_flag):
        """
        Save a listing in the database. Delegates to lower-level methods
        to do the actual database operations.
        Return values:
            True: listing is saved in the database
            False: listing already existed
        """
        try:
            rowcount = -1
            if self.deleted == 1:
                self.save_as_deleted()
            else:
                if insert_replace_flag == self.config.FLAGS_INSERT_REPLACE:
                    rowcount = self.__update()
                if (rowcount == 0 or
                        insert_replace_flag == self.config.FLAGS_INSERT_NO_REPLACE):
                    try:
                        self.__insert()
                        return True
                    except psycopg2.IntegrityError:
                        logger.debug("Room " + str(self.room_id) + ": already collected")
                        return False
        except psycopg2.OperationalError:
            # connection closed
            logger.error("Operational error (connection closed): resuming")
            del(self.config.connection)
        except psycopg2.DatabaseError as de:
            self.config.connection.conn.rollback()
            logger.erro(psycopg2.errorcodes.lookup(de.pgcode[:2]))
            logger.error("Database error: resuming")
            del(self.config.connection)
        except psycopg2.InterfaceError:
            # connection closed
            logger.error("Interface error: resuming")
            del(self.config.connection)
        except psycopg2.Error as pge:
            # database error: rollback operations and resume
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
            self.config.connection.conn.rollback()
            logger.error("Database error: " + str(self.room_id))
            logger.error("Diagnostics " + pge.diag.message_primary)
            del(self.config.connection)
        except (KeyboardInterrupt, SystemExit):
            raise
        except UnicodeEncodeError as uee:
            logger.error("UnicodeEncodeError Exception at " +
                         str(uee.object[uee.start:uee.end]))
            raise
        except ValueError:
            logger.error("ValueError for room_id = " + str(self.room_id))
        except AttributeError:
            logger.error("AttributeError")
            raise
        except Exception:
            self.config.connection.rollback()
            logger.error("Exception saving room")
            raise

    def print_from_web_site(self):
        """ What is says """
        try:
            print_string = "Room info:"
            print_string += "\n\troom_id:\t" + str(self.room_id)
            print_string += "\n\tsurvey_id:\t" + str(self.survey_id)
            print_string += "\n\thost_id:\t" + str(self.host_id)
            print_string += "\n\troom_type:\t" + str(self.room_type)
            print_string += "\n\tcountry:\t" + str(self.country)
            print_string += "\n\tcity:\t\t" + str(self.city)
            print_string += "\n\tneighborhood:\t" + str(self.neighborhood)
            print_string += "\n\taddress:\t" + str(self.address)
            print_string += "\n\treviews:\t" + str(self.reviews)
            print_string += "\n\toverall_satisfaction:\t"
            print_string += str(self.overall_satisfaction)
            print_string += "\n\taccommodates:\t" + str(self.accommodates)
            print_string += "\n\tbedrooms:\t" + str(self.bedrooms)
            print_string += "\n\tbathrooms:\t" + str(self.bathrooms)
            print_string += "\n\tprice:\t\t" + str(self.price)
            print_string += "\n\tdeleted:\t" + str(self.deleted)
            print_string += "\n\tlatitude:\t" + str(self.latitude)
            print_string += "\n\tlongitude:\t" + str(self.longitude)
            print_string += "\n\tminstay:\t" + str(self.minstay)
            print_string += "\n\tcoworker_hosted:\t" + str(self.coworker_hosted)
            print_string += "\n\tlanguages:\t" + str(self.extra_host_languages)
            print_string += "\n\tproperty_type:\t" + str(self.property_type)
            print(print_string)
        except Exception:
            raise

    def print_from_db(self):
        """ What it says """
        try:
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
            columns = self.get_columns()
            sql = "select room_id"
            for column in columns[1:]:
                sql += ", " + column
            sql += " from room where room_id = %s"
            conn = self.config.connect()
            cur = conn.cursor()
            cur.execute(sql, (self.room_id,))
            result_set = cur.fetchall()
            if len(result_set) > 0:
                for result in result_set:
                    i = 0
                    print("Room information: ")
                    for column in columns:
                        print("\t", column, "=", str(result[i]))
                        i += 1
                return True
            else:
                print("\nNo room", str(self.room_id), "in the database.\n")
                return False
            cur.close()
        except Exception:
            raise

    def ws_get_room_info(self, flag):
        """ Get the room properties from the web site """
        try:
            # initialization
            logger.info("-" * 70)
            logger.info("Room " + str(self.room_id) +
                        ": getting from Airbnb web site")
            room_url = self.config.URL_ROOM_ROOT + str(self.room_id)
            response = airbnb_ws.ws_request_with_repeats(self.config, room_url)
            if response is not None:
                page = response.text
                tree = html.fromstring(page)
                self.__get_room_info_from_tree(tree, flag)
                return True
            else:
                return False
        except (KeyboardInterrupt, SystemExit):
            raise
        except Exception as ex:
            logger.exception("Room " + str(self.room_id) +
                             ": failed to retrieve from web site.")
            logger.error("Exception: " + str(type(ex)))
            raise

    def __insert(self):
        """ Insert a room into the database. Raise an error if it fails """
        try:
            logger.debug("Values: ")
            logger.debug("\troom_id: {}".format(self.room_id))
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
            logger.debug("\thost_id: {}".format(self.host_id))
            conn = self.config.connect()
            cur = conn.cursor()
            sql = """
                insert into room (
                    room_id, host_id, room_type, country, city,
                    neighborhood, address, reviews, overall_satisfaction,
                    accommodates, bedrooms, bathrooms, price, deleted,
                    minstay, latitude, longitude, survey_id,
                    coworker_hosted, extra_host_languages, name,
                    property_type, currency, rate_type

                )
                """
            sql += """
                values (%s, %s, %s, %s, %s, %s, %s, %s, %s,
                %s, %s, %s, %s, %s, %s, %s, %s, %s,
                %s, %s, %s, %s, %s, %s
                )"""
            insert_args = (
                self.room_id, self.host_id, self.room_type, self.country,
                self.city, self.neighborhood, self.address, self.reviews,
                self.overall_satisfaction, self.accommodates, self.bedrooms,
                self.bathrooms, self.price, self.deleted, self.minstay,
                self.latitude, self.longitude, self.survey_id,
                self.coworker_hosted, self.extra_host_languages, self.name,
                self.property_type, self.currency, self.rate_type
                )
            cur.execute(sql, insert_args)
            cur.close()
            conn.commit()
            logger.debug("Room " + str(self.room_id) + ": inserted")
            logger.debug("(lat,      long)      =      ({lat:+.5f},      {lng:+.5f})".format(lat=self.latitude,
lng=self.longitude))
        except psycopg2.IntegrityError:
            # logger.info("Room " + str(self.room_id) + ": insert failed")
            conn.rollback()
            cur.close()
            raise
        except:
            conn.rollback()
            raise

    def __update(self):
        """ Update a room in the database. Raise an error if it fails.
        Return number of rows affected."""
        try:
            rowcount = 0
            conn = self.config.connect()
            cur = conn.cursor()
            logger.debug("Updating...")
            sql = """
                update room
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
                set host_id = %s, room_type = %s,
                    country = %s, city = %s, neighborhood = %s,
                    address = %s, reviews = %s, overall_satisfaction = %s,
                    accommodates = %s, bedrooms = %s, bathrooms = %s,
                    price = %s, deleted = %s, last_modified = now()::timestamp,
                    minstay = %s, latitude = %s, longitude = %s,
                    coworker_hosted = %s, extra_host_languages = %s, name = %s,
                    property_type = %s, currency = %s, rate_type = %s
                where room_id = %s
                and survey_id = %s"""
            update_args = (
                self.host_id, self.room_type,
                self.country, self.city, self.neighborhood,
                self.address, self.reviews, self.overall_satisfaction,
                self.accommodates, self.bedrooms, self.bathrooms,
                self.price, self.deleted,
                self.minstay, self.latitude,
                self.longitude,
                self.coworker_hosted, self.extra_host_languages, self.name,
                self.property_type, self.currency, self.rate_type,
                self.room_id,
                self.survey_id,
                )
            logger.debug("Executing...")
            cur.execute(sql, update_args)
            rowcount = cur.rowcount
            logger.debug("Closing...")
            cur.close()
            conn.commit()
            logger.info("Room " + str(self.room_id) +
                        ": updated (" + str(rowcount) + ")")
            return rowcount
        except:
            # may want to handle connection close errors
            logger.warning("Exception in __update: raising")
            raise

    def __get_country(self, tree):
        try:
            temp = tree.xpath(
                "//meta[contains(@property,'airbedandbreakfast:country')]"
                "/@content"
                )
            if len(temp) > 0:
                self.country = temp[0]
        except:
            raise

    def __get_city(self, tree):
        try:
            temp = tree.xpath(
                "//meta[contains(@property,'airbedandbreakfast:city')]"
                "/@content"
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
                )
            if len(temp) > 0:
                self.city = temp[0]
        except:
            raise

    def __get_rating(self, tree):
        try:
            # 2016-04-10
            s = tree.xpath("//meta[@id='_bootstrap-listing']/@content")
            temp = tree.xpath(
                "//meta[contains(@property,'airbedandbreakfast:rating')]"
                "/@content"
                )
            if s is not None:
                j = json.loads(s[0])
                self.overall_satisfaction = j["listing"]["star_rating"]
            elif len(temp) > 0:
                self.overall_satisfaction = temp[0]
        except IndexError:
            return
        except:
            raise

    def __get_latitude(self, tree):
        try:
            temp = tree.xpath("//meta"
                              "[contains(@property,"
                              "'airbedandbreakfast:location:latitude')]"
                              "/@content")
            if len(temp) > 0:
                self.latitude = temp[0]
        except:
            raise

    def __get_longitude(self, tree):
        try:
            temp = tree.xpath(
                "//meta"
                "[contains(@property,'airbedandbreakfast:location:longitude')]"
                "/@content")
            if len(temp) > 0:
                self.longitude = temp[0]
        except:
            raise

    def __get_host_id(self, tree):
        try:
            # 2016-04-10
            s = tree.xpath("//meta[@id='_bootstrap-listing']/@content")
            temp = tree.xpath(
                "//div[@id='host-profile']"
                "//a[contains(@href,'/users/show')]"
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
            "/@href"
        )
        if s is not None:
            j = json.loads(s[0])
            self.host_id = j["listing"]["user"]["id"]
            return
        elif len(temp) > 0:
            host_id_element = temp[0]
            host_id_offset = len('/users/show/')
            self.host_id = int(host_id_element[host_id_offset:])
        else:
            temp = tree.xpath(
                "//div[@id='user']"
                "//a[contains(@href,'/users/show')]"
                "/@href")
            if len(temp) > 0:
                host_id_element = temp[0]
                host_id_offset = len('/users/show/')
                self.host_id = int(host_id_element[host_id_offset:])
    except IndexError:
        return
    except:
        raise

def __get_room_type(self, tree):
    try:
        # -- room type --
        # new page format 2015-09-30?
        temp = tree.xpath(
            "//div[@class='col-md-6']"
            "/div/span[text()[contains(.,'Room type:')]]"
            "/../strong/text()"
            )
        if len(temp) > 0:
            self.room_type = temp[0].strip()
        else:
            # new page format 2014-12-26
            temp_entire = tree.xpath(
                "//div[@id='summary']"
                "//i[contains(concat(' ', @class, ' '),"
                " ' icon-entire-place ')]"
                )
            if len(temp_entire) > 0:
                self.room_type = "Entire home/apt"
            temp_private = tree.xpath(
                "//div[@id='summary']"
                "//i[contains(concat(' ', @class, ' '),"
                " ' icon-private-room ')]"
                )
            if len(temp_private) > 0:
                self.room_type = "Private room"
            temp_shared = tree.xpath(
                "//div[@id='summary']"
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
                "//i[contains(concat(' ', @class, ' '),"
                " ' icon-shared-room ')]"
                )
            if len(temp_shared) > 0:
                self.room_type = "Shared room"
    except:
        raise

def __get_neighborhood(self, tree):
    try:
        temp2 = tree.xpath(
            "//div[contains(@class,'rich-toggle')]/@data-address"
            )
        temp1 = tree.xpath("//table[@id='description_details']"
                           "//td[text()[contains(.,'Neighborhood:')]]"
                           "/following-sibling::td/descendant::text()")
        if len(temp2) > 0:
            temp = temp2[0].strip()
            self.neighborhood = temp[temp.find("(")+1:temp.find(")")]
        elif len(temp1) > 0:
            self.neighborhood = temp1[0].strip()
        if self.neighborhood is not None:
            self.neighborhood = self.neighborhood[:50]
    except:
        raise

def __get_address(self, tree):
    try:
        temp = tree.xpath(
            "//div[contains(@class,'rich-toggle')]/@data-address"
            )
        if len(temp) > 0:
            temp = temp[0].strip()
            self.address = temp[:temp.find(",")]
        else:
            # try old page match
            temp = tree.xpath(
                "//span[@id='display-address']"
                "/@data-location"
                )
            if len(temp) > 0:
                self.address = temp[0]
    except:
        raise

def __get_reviews(self, tree):
    try:
        # 2016-04-10
        s = tree.xpath("//meta[@id='_bootstrap-listing']/@content")
        # 2015-10-02
        temp2 = tree.xpath(
            "//div[@class='___iso-state___p3summarybundlejs']"
            "/@data-state"
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
            )
        if s is not None:
            j = json.loads(s[0])
            self.reviews = \
                j["listing"]["review_details_interface"]["review_count"]
        elif len(temp2) == 1:
            summary = json.loads(temp2[0])
            self.reviews = summary["visibleReviewCount"]
        elif len(temp2) == 0:
            temp = tree.xpath(
                "//div[@id='room']/div[@id='reviews']//h4/text()")
            if len(temp) > 0:
                self.reviews = temp[0].strip()
                self.reviews = str(self.reviews).split('+')[0]
                self.reviews = str(self.reviews).split(' ')[0].strip()
            if self.reviews == "No":
                self.reviews = 0
        else:
            # try old page match
            temp = tree.xpath(
                "//span[@itemprop='reviewCount']/text()"
                )
            if len(temp) > 0:
                self.reviews = temp[0]
        if self.reviews is not None:
            self.reviews = int(self.reviews)
    except IndexError:
        return
    except Exception as e:
        logger.exception(e)
        self.reviews = None

def __get_accommodates(self, tree):
    try:
        # 2016-04-10
        s = tree.xpath("//meta[@id='_bootstrap-listing']/@content")
        temp = tree.xpath(
            "//div[@class='col-md-6']"
            "/div/span[text()[contains(.,'Accommodates:')]]"
            "/../strong/text()"
            )
        if s is not None:
            j = json.loads(s[0])
            self.accommodates = j["listing"]["person_capacity"]
            return
        elif len(temp) > 0:
            self.accommodates = temp[0].strip()
        else:
            temp = tree.xpath(
                "//div[@class='col-md-6']"
                "/div[text()[contains(.,'Accommodates:')]]"
                "/strong/text()"
                )
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
                if len(temp) > 0:
                    self.accommodates = temp[0].strip()
                else:
                    temp = tree.xpath(
                        "//div[@class='col-md-6']"
                        "//div[text()[contains(.,'Accommodates:')]]"
                        "/strong/text()"
                    )
                    if len(temp) > 0:
                        self.accommodates = temp[0].strip()
            if type(self.accommodates) == str:
                self.accommodates = self.accommodates.split('+')[0]
                self.accommodates = self.accommodates.split(' ')[0]
            self.accommodates = int(self.accommodates)
        except:
            self.accommodates = None

    def __get_bedrooms(self, tree):
        try:
            temp = tree.xpath(
                "//div[@class='col-md-6']"
                "/div/span[text()[contains(.,'Bedrooms:')]]"
                "/../strong/text()"
                )
            if len(temp) > 0:
                self.bedrooms = temp[0].strip()
            else:
                temp = tree.xpath(
                    "//div[@class='col-md-6']"
                    "/div[text()[contains(.,'Bedrooms:')]]"
                    "/strong/text()"
                    )
                if len(temp) > 0:
                    self.bedrooms = temp[0].strip()
            if self.bedrooms:
                self.bedrooms = self.bedrooms.split('+')[0]
                self.bedrooms = self.bedrooms.split(' ')[0]
            self.bedrooms = float(self.bedrooms)
        except:
            self.bedrooms = None

    def __get_bathrooms(self, tree):
        try:
            temp = tree.xpath(
                "//div[@class='col-md-6']"
                "/div/span[text()[contains(.,'Bathrooms:')]]"
                "/../strong/text()"
                )
            if len(temp) > 0:
                self.bathrooms = temp[0].strip()
            else:
                temp = tree.xpath(
                    "//div[@class='col-md-6']"
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
                "/div/span[text()[contains(.,'Bathrooms:')]]"
                "/../strong/text()"
                )
            if len(temp) > 0:
                self.bathrooms = temp[0].strip()
        if self.bathrooms:
            self.bathrooms = self.bathrooms.split('+')[0]
            self.bathrooms = self.bathrooms.split(' ')[0]
        self.bathrooms = float(self.bathrooms)
    except:
        self.bathrooms = None

def __get_minstay(self, tree):
    try:
        # -- minimum stay --
        temp3 = tree.xpath(
            "//div[contains(@class,'col-md-6')"
            "and text()[contains(.,'minimum stay')]]"
            "/strong/text()"
            )
        temp2 = tree.xpath(
            "//div[@id='details-column']"
            "//div[contains(text(),'Minimum Stay:')]"
            "/strong/text()"
            )
        temp1 = tree.xpath(
            "//table[@id='description_details']"
            "//td[text()[contains(.,'Minimum Stay:')]]"
            "/following-sibling::td/descendant::text()"
            )
        if len(temp3) > 0:
            self.minstay = temp3[0].strip()
        elif len(temp2) > 0:
            self.minstay = temp2[0].strip()
        elif len(temp1) > 0:
            self.minstay = temp1[0].strip()
        if self.minstay is not None:
            self.minstay = self.minstay.split('+')[0]
            self.minstay = self.minstay.split(' ')[0]
        self.minstay = int(self.minstay)
    except:
        self.minstay = None

def __get_price(self, tree):
    try:
        temp2 = tree.xpath(
            "//meta[@itemprop='price']/@content"
            )
        temp1 = tree.xpath(
            "//div[@id='price_amount']/text()"
            )
        if len(temp2) > 0:
            self.price = temp2[0]
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
            elif len(temp1) > 0:
                self.price = temp1[0][1:]
                non_decimal = re.compile(r'[^\d.]+')
                self.price = non_decimal.sub('', self.price)
            # Now find out if it's per night or per month
            # (see if the per_night div is hidden)
            per_month = tree.xpath(
                "//div[@class='js-per-night book-it__payment-period  hide']")
            if per_month:
                self.price = int(int(self.price) / 30)
            self.price = int(self.price)
        except:
            self.price = None

    def __get_room_info_from_tree(self, tree, flag):
        try:
            # Some of these items do not appear on every page (eg,
            # ratings, bathrooms), and so their absence is marked with
            # logger.info. Others should be present for every room (eg,
            # latitude, room_type, host_id) and so are marked with a
            # warning.  Items coded in <meta
            # property="airbedandbreakfast:*> elements -- country --

            self.__get_country(tree)
            self.__get_city(tree)
            self.__get_rating(tree)
            self.__get_latitude(tree)
            self.__get_longitude(tree)
            self.__get_host_id(tree)
            self.__get_room_type(tree)
            self.__get_neighborhood(tree)
            self.__get_address(tree)
            self.__get_reviews(tree)
            self.__get_accommodates(tree)
            self.__get_bedrooms(tree)
            self.__get_bathrooms(tree)
            self.__get_minstay(tree)
            self.__get_price(tree)
            self.deleted = 0

            # NOT FILLING HERE, but maybe should? have to write helper methods:
            # coworker_hosted, extra_host_languages, name,
            #     property_type, currency, rate_type

            self.status_check()

            if flag == self.config.FLAGS_ADD:
                self.save(self.config.FLAGS_INSERT_REPLACE)
            elif flag == self.config.FLAGS_PRINT:
                self.print_from_web_site()
            return True
        except (KeyboardInterrupt, SystemExit):
            raise
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY

```
except IndexError:
    logger.exception("Web page has unexpected structure.")
    raise
except UnicodeEncodeError as uee:
    logger.exception("UnicodeEncodeError Exception at " +
                     str(uee.object[uee.start:uee.end]))
    raise
except AttributeError:
    logger.exception("AttributeError")
    raise
except TypeError:
    logger.exception("TypeError parsing web page.")
    raise
except Exception:
    logger.exception("Error parsing web page.")
    raise
```

제주대학교 중앙도서관
JEJU NATIONAL UNIVERSITY LIBRARY