How Hotels Could Improve Consumers' Satisfaction Using Online Reviews

Introduction & Literature Review

If you want to book a hotel online, you may read reviews first. According to TripAdvisor Traveler Survey (2018), online reviews affect people's booking behaviors. Among TripAdvisor users, 96% agree that reading reviews is important. 83% "usually" or "always" check reviews and 79% read six to twelve reviews or more before making a decision, while 76% report that user-uploaded pictures had influenced their choice. Indeed, this result intuitively shows eWOM is so ubiquitous that influences consumers' decision-making process in the hospitality industry nowadays. According to Litvin et al. (2018), electronic word-of-mouth (eWOM) refers to all informal communications, related to the usage or characteristics of specific goods, services and sellers, directed against consumers through Internet-based technology. It not only exists between producers and consumers, but also between consumers themselves.

To explain online review behaviors and eWOM, scholars borrowed the uncertainty reduction theory from Charles Berger and Richard Calabrese: "The acquisition, processing, retention, and retrieval of information is vital to the growth, maintenance, and decline of personal and social relationships" (as cited in Redmond, 2015, p.1). Thus, the rationale lies in the desire of reducing uncertainty of immaterial, indivisible service bundles by information exchange (Papathanassis & Knolle 2011). What customers usually take into account are the perceived trustworthiness, credibility, and usefulness of review (Litvin et al., 2018).

Different aspects of online review messages impact on consumers' evaluation and intention, including rating, text, pictures and review helpfulness. Rating is usually on a one-to-five scale, where one-score means "extremely displeased", "terrible experience" and five-score means "extremely satisfied", "excellent experience." Rating is also called valence**,** "the level of consumer satisfaction and product

evaluation" which correlates positively with expectations of a hotel and intention to purchase (Kwok et al., 2017; Ladhari & Michaud, 2015). Rating has variation, "the standard deviation to the average rating" which implies the degree of disagreement between customers (Kwok et al., 2017). Variation of rating is negatively related to hotel booking (Xie et al., 2014).

Text and picture are the backbones of an online review. Park and Nicolau (2015) found that lengthier reviews convey more information, so they seemed more useful and trustworthy to travelers. Meanwhile, Kwok and Xie (2016) showed that the readability of information also matters: Customers may refer to easy-to-read reviews that offer the needed information. In addition to textual messages, user-generated pictures also affect the review information's value. As a form of peripheral cue, photos serve as a means of persuasive communication to improve credibility (Zhang et al., 2016). Ma et al. (2018) found that pictures alone do not exert the same level of impact as review texts, but the combination of photo and text will have the highest impact. The more textual and photo reviews there are, the more popular a hotel is (Xie et al., 2014). A relevant term is volume, "a measure of the number of consumer reviews" which indicates exposures and popularity online (Kwok et al., 2017).

Apart from texts and pictures, review helpfulness is also a research focus. Thanks to socializing functions on websites and mobile apps, users can vote for helpful reviews. Reviews marked as helpful usually are those with extreme ratings (Park & Nicolau, 2015), being consistent with each other (Zehrer et al., 2011), expressing genuine emotions (Lee et al., 2012) and being posted by reviewers who possess higher expertise (Liu & Park, 2015). Researchers also proved that if consumers perceive online reviews to be useful, their online booking intentions would increase (Zhao et al., 2015). However, Ma et al. (2018) wrote that "the number of helpfulness votes tend to diminish considerably for those which already have a substantial number of reviews with a handful of helpfulness votes." In general, eWOM affects buying behaviors.

eWOM may be segmented and differentiated by product attributes like hotel star levels. After analyzing TripAdvisor reviews of hotels in Beijing and Shanghai, Hu (2019) discovered that for higher star-level hotels, reviewers frequently

mentioned hotel service and amenities; for lower star-level hotels, convenience and value were more conspicuous. In addition, some attributes fit into only one particular hotel star level, such as proximity to a station (three-star), bed and lobby (four-star), and experience and drinks (five-star) (Hu, 2019). Other helpful attributes fit in with different hotel levels, including walking distance (three and four stars), English fluency of staff (four and five stars), and frontdesk, street-road, taxi, and value (three to five stars) (Hu, 2019).

Although abundant studies are about eWOM in the hospitality industry, some topics have not yet been covered. Most papers in the English-speaking countries do not focus on hotels in China's emerging cities listed on Chinese online booking platforms. To bridge this gap, we will analyze three-to-five stars hotels in Shenzhen listed on Ctrip.com (*xiecheng*). This paper explores how hotels can leverage the power of online reviews to improve customers' satisfaction. Specifically, it consists of three research questions:

1. **What kind of reviews are deemed helpful?**
2. **How consumer segments differ in concerns and sentiments?**
3. **How review scores can be forecasted?**

Data

The total number of three to five stars' hotels in Shenzhen is 1,000 and they follow the ratio of five-star: four-star: three-star = 1:3:6. We stratified on hotel star level and sampled 211 hotels based on 95% confidence level, 6% confidence interval. On average, we have 1,200 reviews per hotel. We crawled the data from Ctrip.com using a software called Houyi. For this report, reviews are collected until Mar. 28th, 2020. For data using NLP in the report, data is cleaned by stopwords with Baidu Language Database as well as manually cleaning meaningless words which are not included in the database. Stopwords can be found in the folder named "support_info".

Methods & Analyses & Results

Research Question 1

Firstly, we investigate the relationship between hotel official descriptions on Ctrip.com and hotel total scores. We extract keywords from hotel descriptions and classify them into 12 categories listed in the table 1. For each category, if the hotel description contains such information, we input 1. Otherwise, we input 0. We define variable Complexity to measure the information abundance of the hotel description in the booking website. Complexity is computed according to the equation:

$$\begin{aligned} Complexity = {} & locality_i + transportation_i + local\ business_i + restaurtant_i \\ & + business\ meeting_i + fitness\ entertainment_i + landscape_i \\ & + sepcial\ culture\ and\ idea_i + laundry_i + decoration_i \\ & + experience\ expectation_i + offical\ photo\ number_i \\ & + \frac{[photo\_number_i - \mu(photo\_number_i)]}{standard\ deviation(photo\_number_i)} \end{aligned}$$

We use Stata command *pw_corr* to compute the correlation between hotel's total score and variables we listed above. The result suggests that there is a significant positive correlation between Complexity and hotel's total score, which means the more abundant the hotel description is, the higher total score the hotel gets.

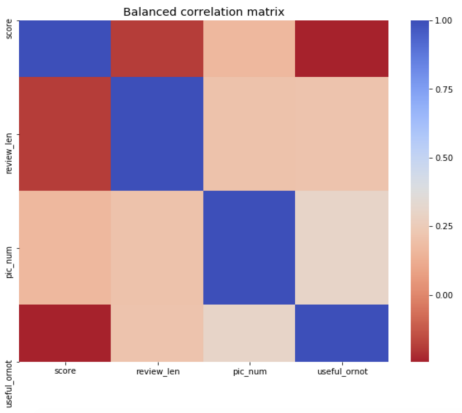
Table 1. Correlation between Hotel Score and Information Abundance

Variable	Hotel Score	Significance Level
Locality	.	
Transportation	0.170	
Local Business	0.266	*
Restaurant	0.176	
Business Meeting	0.051	
Fitness Entertainment	0.162	

Landscape	0.012	
Special Culture & Idea	0.072	
Laundry	0.262	*
Decoration	0.322	**
Experience Expectation	0.308	**
Official Photo Number	-0.283	**
Complexity	0.261	**

After the general exploration, we focus on reviews. While browsing through the webpage, viewers may rate certain reviews as "useful" and then the property named "useful review" will increase by one. This action indicates viewers' recognition, and whether these useful reviews point out specific aspects of the hotels becomes our research direction. The reason is that hotels can improve hotel official descriptions based on what consumers most concern about, while useful reviews may also strengthen the authenticity and reliability of hotel performance. We first conduct descriptive analytics to see if there are correlations among variables. As is shown in figure 1, there is little evidence, and only "the number of picture" and "whether the review is useful or not" show an approximately 0.3 correlation coefficient. Low correlations lead to more precise prediction. Since the variable correlations are low, predictive analytics will not be affected.

Figure 1. Correlation among variables related to useful reviews



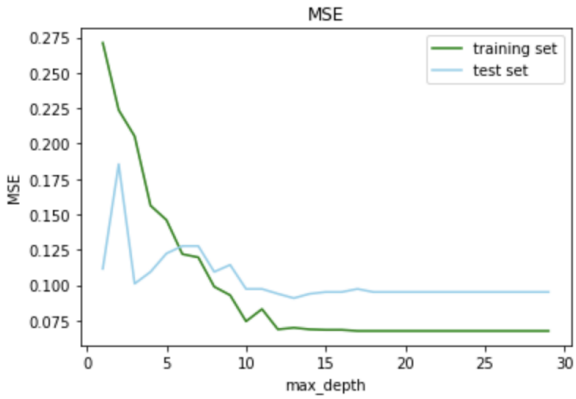
But the problem is that the dataset is unbalanced with 96% of observations in "not useful reviews" class. To reduce bias, we apply SMOTE, a synthetic minority over-sampling technique, to balance the dataset which gives the ratio between "useful reviews" (marked as "1") and "not useful reviews" (marked as "0") to be 0.5:0.5 (table 2).

Table 2. Ratio of useful & not-useful review before and after adjustment

```
0    0.958153
1    0.041847
Name: useful_ornot, dtype: float64
1    0.5
0    0.5
Name: useful_ornot, dtype: float64
```

Moreover, to prevent overfitting, mean squared errors of the training set and test set are plotted as figure 2. As usual, the error of the training set keeps dropping but when the depth reaches 13, the test set reaches a global minimum. Hence, a tree with depth equals to 13 is selected as the final model.

Figure 2. Mean squared error of the training set and test set



The tree is pruned with Grid Search scoring on recall rates. The new best model presents an accuracy of 0.86 and recall rates of two classes, 0.87 and 0.71 respectively, are shown in table 3. The recall rate of "useful review" has been largely improved from 0.31 to 0.71. Although the accuracy drops 0.4 and the recall rate of "not useful review" drops 0.7, the new best model can still be regarded as more balanced and reliable. Under this circumstance, the "review score" is recognized as the most important feature, while "the number of picture" is almost the same important as the review score, given that the importance of "review

score", "review length" and "the number of picture" are 0.37, 0.27, and 0.36 respectively.

Table 3. Evaluation of decision tree modelling before and after Grid Search

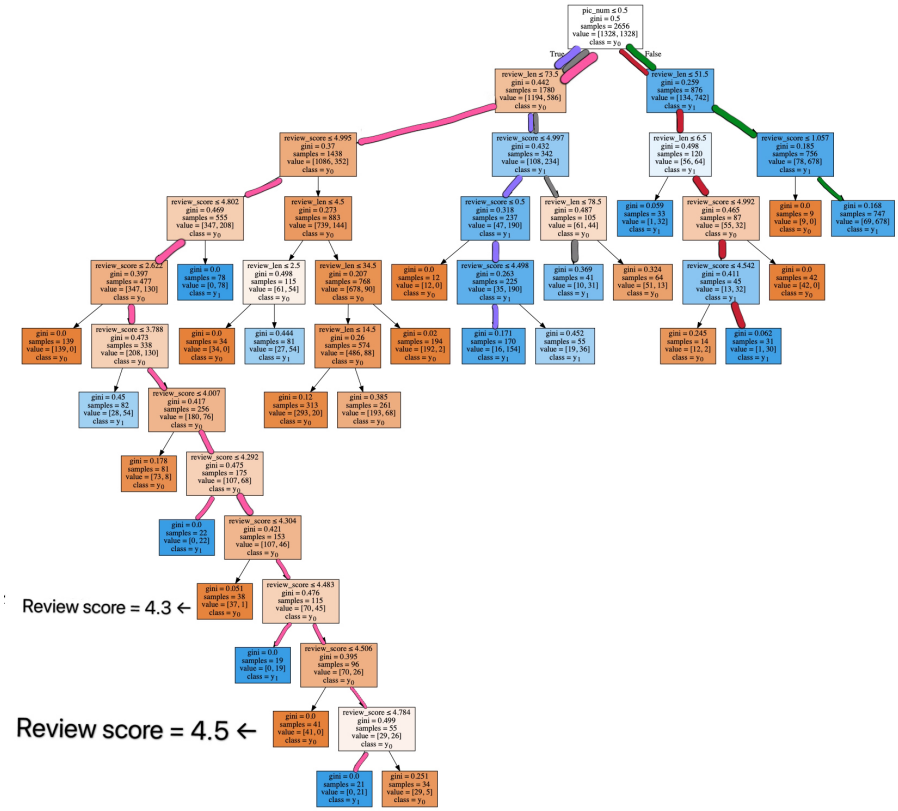
```
                precision    recall  f1-score   support

             0       0.96      0.94      0.95      3056
             1       0.23      0.31      0.26       178

      accuracy                           0.90      3234
     macro avg       0.59      0.63      0.61      3234
  weighted avg       0.92      0.90      0.91      3234

Test set score: 0.71
Best parameters:{'max_depth': 13, 'max_leaf_nodes': 26, 'min_samples_split': 2}
Best score on train set: 0.90
                precision    recall  f1-score   support

             0       0.98      0.87      0.92      3056
             1       0.24      0.71      0.36       178

      accuracy                           0.86      3234
     macro avg       0.61      0.79      0.64      3234
  weighted avg       0.94      0.86      0.89      3234

Importance of all features:
 [0.37115182 0.26529785 0.36355033]

The most importance features is: score
```

Then the decision tree is plotted as figure 3. The tree shows that reviews have higher chances to become useful with photos. Among the reviews, those between 7 and 51 words with scores between 4.5 and 5 are seen as useful (Red Route in figure 3). On the other hand, for reviews more than 51 words, if scores are higher than 1, they are also more likely to be useful (Green Route in figure 3). Nevertheless, reviews with extremely low scores (less than 1) are not considered as useful even when they are longer than 51 words, because it is highly possible that they become so negative that are suspected to be unfair. For the reviews without photos, unfortunately, the majority of them are not useful because there are 1,194 not useful reviews out of 1,780 reviews. In a nutshell, photos are important, and either long reviews or fair ratings can win the recognition of viewers.

In this scenario, the longer the reviews, the more of them are considered as useful. When reviews are more than 74 words, those with relatively middle scores (0.5-4.5) are more likely to be useful (Purple Route in figure 3) than those with high scores (over 4.5). It's probably because detailed reviews reflect some fair opinions and true feelings, and very high scores may not be considered as real by the prospective customers. Moreover, if the length of full-score reviews is within the range of 74 to 78, they are likely to be useful (Grey Route in figure 3). Reviews of this length may be easy to read and therefore increase the chance of being agreed with. When reviews are shorter than 74 words from which relatively less

information can be extracted, those with a high score between 4 and 5 are all considered as useful except for the score of 4.3 or 4.5 when the reviews are not useful (Pink Route in figure 3). In reality, it is hard to guarantee such precision to be consistent over time, so this conclusion lacks universality. There are 54 reviews with scores less than 4 that are useful but this group is small, so we still focus on the majority.

Figure 3. Decision tree details



To delve deeper into the content of the useful reviews, we utilize those reviews on Ctrip.com and generate word cloud (figure 4) for all the useful reviews in our sample. It's shown that the most frequently mentioned aspects are: "room is capacious", "satisfying service", "clean and tidy", "excellent location", "convenient transportation", "very satisfied", "recommended", "nice breakfast", and "cost-effective". To further investigate whether different groups of customers might have a similar focus of attention, we utilize cluster analysis with K-means method. However, figure 5 shows that there is no clear significant drop on the elbow graph which means that the results of clustering will not be ideal, so we do not proceed with this method., so we do not proceed with this method.

Figure 4. Word Cloud for useful reviews

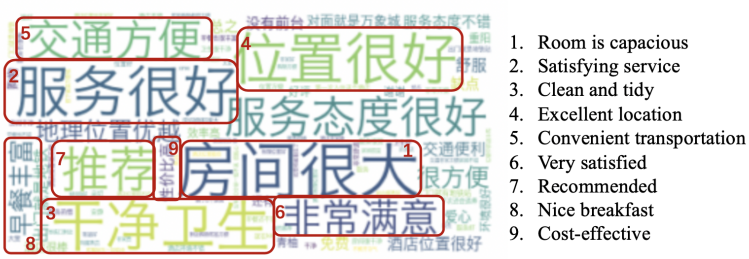
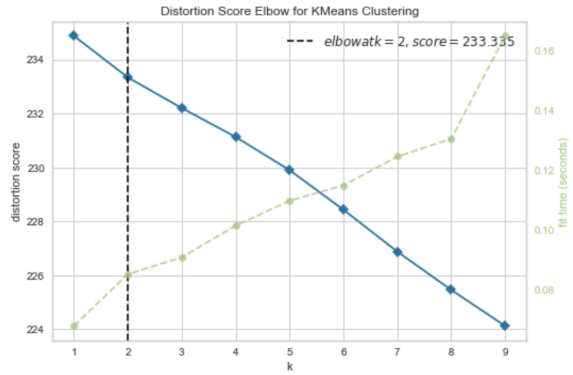


Figure 5. Cluster of useful reviews



In conclusion, among the reviews that are regarded as "useful", customers and viewers lay much emphasis on the room facility and cleanness, service quality, transportation and location, and even the food offered at the hotels, high standards of which lead to cost-effectiveness and satisfaction. Therefore, hotel managers shall strive to improve these aspects to ensure a more satisfying stay and to make sure that the price is reasonable, which may contribute to higher ratings. To obtain more useful reviews, hotels can suggest customers to post more photos, together with detailed reviews that express their true feelings. Hotels could also advise customers about giving fair scores objectively to avoid extreme scores. Kind reminders can be posted on the hotel webpages, on the platforms, at the reception desk in the hotel, and even on the reminder cards in the hotel rooms. It is anticipated that hotels will witness substantial business growth.

Research question 2

Different types of consumers may have different concerns, which can help hotels to customize service. Consumer types here are defined as business travelers, couple travelers, family travelers, friend travelers and alone travelers according to Ctrip Platform 's consumer segment principle. Hotels can target on consumer

types that have the lowest satisfaction to maximize efficiency and effectiveness. Using chi-square tests on consumer types and review scores, and using NLP to make text analysis, we answer research question 2 by resolving four sub-questions:

1. Which consumer type should hotels approach first to improve customers' satisfaction?
2. What are the major concerns of different types of consumers?
   This is because different consumer types may have different focus. For example, some consumers care about room design and service attitude while others concern more about price and location. We tend to find out the main focus of each consumer type.
3. What are the detailed concerns related to each major concern?
   Consumers may have some detailed concerns related to each major focus. For example, consumers who consider room as a major concern may pay high attention to design, space or inner equipment. We tend to find out detailed concerns related to each major concern of each consumer type.
4. What is the sentiment of each major concern for different consumer types?
   Different types of consumers may have different sentiments in the same major concern. Some consumers may tend to give positive comments on service attitude while others' may be more negative. We tend to find out the sentiment of each major concern of different consumer types.

1. Which consumer type should hotels approach first to improve customers' satisfaction?

We want to know how the distributions of review scores for these five consumer types are different from each other. We conduct chi-square tests on two variables, rounded review scores and travelling types, for the three-star, four-star, five-star hotels respectively. In table 4-6, chi_p is the chi-square test statistic. For the last column named 'similar', if it's False, then the chi-square test has a p-value<=0.01, which means on the 99% confidence level, the corresponding two pairs of consumer types have different distributions. From the results we can conclude that single trip, couple trip and friend trip always have similar distributions of rounded review scores in our sample across different hotel star levels. We further group these three types together and compare their box plots with the other two

consumer types for three-star, four-star, and five-star hotels respectively. Figure 6-8 show that family travelers account for the largest proportion of giving scores less than 5 in both three-star and five-star hotels. For four-star hotels, business travelers account for the largest proportion, although it's a slight difference.

Table 4. Chi-square tests on rounded review scores and travelling types for three-star hotels

type1	type2	chi_p	similar
family	couple	2.576632e-25	FALSE
friend	family	1.312548e-20	FALSE
family	alone	7.806899e-15	FALSE
couple	business	4.721656e-14	FALSE
family	business	4.839039e-12	FALSE
friend	business	8.315653e-09	FALSE
business	alone	3.943656e-05	FALSE
friend	alone	1.125010e-01	TRUE
couple	alone	1.175513e-01	TRUE
friend	couple	5.961630e-01	TRUE

Table 5. Chi-square tests on rounded review scores and travelling types for four-star hotels

type1	type2	chi_p	similar
couple	business	5.151540e-15	FALSE
friend	business	2.153661e-12	FALSE
family	couple	8.826099e-12	FALSE
friend	family	3.093578e-09	FALSE
family	alone	1.305851e-04	FALSE
business	alone	3.275851e-04	FALSE
friend	alone	1.013121e-02	TRUE
family	business	7.002297e-02	TRUE
couple	alone	9.414618e-02	TRUE
friend	couple	1.036975e-01	TRUE

Table 6. Chi-square tests on rounded review scores and travelling types for five-star hotels

type1	type2	chi_p	similar
family	alone	0.0003445175	FALSE
friend	family	0.0016712600	FALSE
business	alone	0.0037992916	FALSE
friend	business	0.0270327980	TRUE
family	business	0.0921086608	TRUE
couple	alone	0.1071841482	TRUE
family	couple	0.2534632732	TRUE
friend	alone	0.3689171525	TRUE
friend	couple	0.4846748422	TRUE
couple	business	0.7762442959	TRUE

Figure 6. Box-plot on rounded review scores and travelling types for three-star hotels

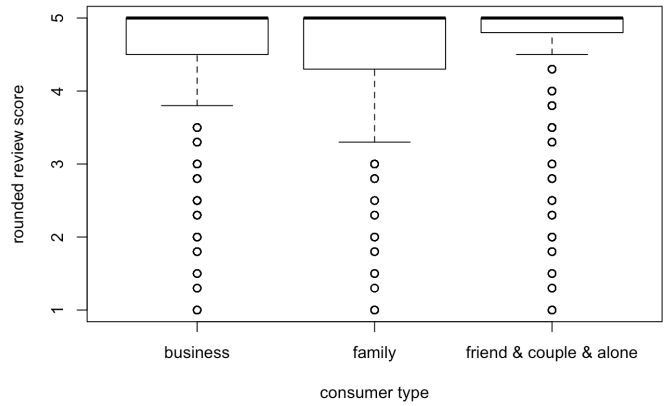


Figure 7. Box-plot on rounded review scores and travelling types for four-star hotels

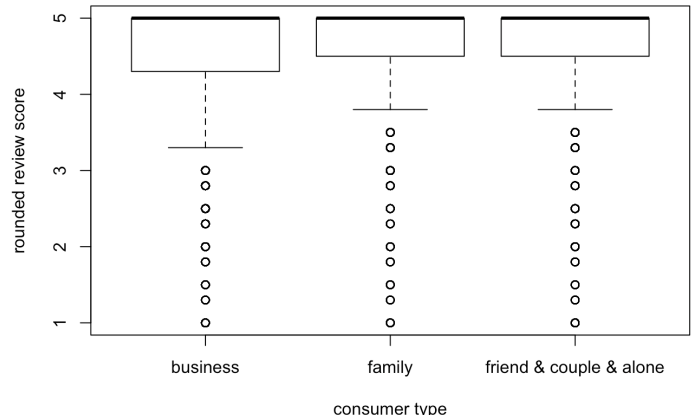
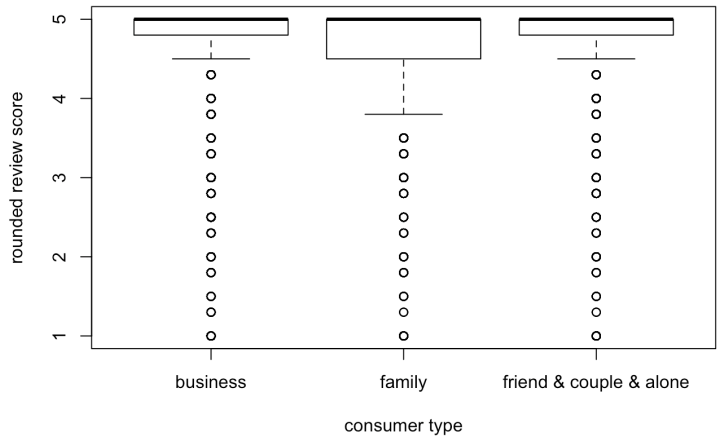


Figure 8. Box-plot on rounded review scores and travelling types for five-star hotels



To maximize the overall performance, only knowing the priority groups is not enough. In this case, strategies targeted on different consumer types are further explored. Due to the space limit, here we take the three-star hotel as an example to conduct three different analyses among customers' major concerns, detailed concerns and the sentiments of their major concerns, respectively. The review

texts are processed by NLP methods and finally visualized through Tableau for further analysis. Firstly, we use Jieba Library in python to segment words. Then we clean the stopwords with Baidu Language Database as well as manually cleaning meaningless words which are not included in the database.

We utilize different NLP methods listed below for sub-questions 2-4 in the following:

Aspect	Question	Method Used	Assumption	Output
Consumer concern	1. What are **major concern**	TF-IDF Value	More important the word, more concerned the word	**Major Concerned Word**
	2. What are **detail concerns** related to each major concern	Word2vec Similarity Index (word embedding)		**Detail Concerned Word**
Review sentiment	3. What is **sentiment** of each major concern	SnowNLP Sentiment Index	Sentiment of a major concern word = SUM(sentiment of each detail concerned word*similaity of this word)	**Sentiment of Major Concerned Word**

2. What are major concerns of different types of consumers?

We use jieba.analyse package in python to calculate TF-IDF (Term Frequency–Inverse Document Frequency) value for each word in the reviews of different types of consumers, and select top 20 words with the highest TF-IDF value (table 7). The major concerns are further narrowed with TF-IDF value higher than 0.1 in table 8. (Complete TF-IDF data named “major concern word.xlsx” can be found in the attached folder “support_info”).

To better analyze the result, we assume that more important words refer to higher concern from consumers. Based on this assumption, we can gain the following insights:

1. The major concerns of different consumers are similar among all types of consumers, which are:Room, Reception, Service attitude, Breakfast, Environment, Cost efficiency, Traffic and Equipment. In addition, words related to metro like "metro entrance" and "metro station" are also

mentioned frequently, which indicates that three-star consumers prefer metro more than self-driving.

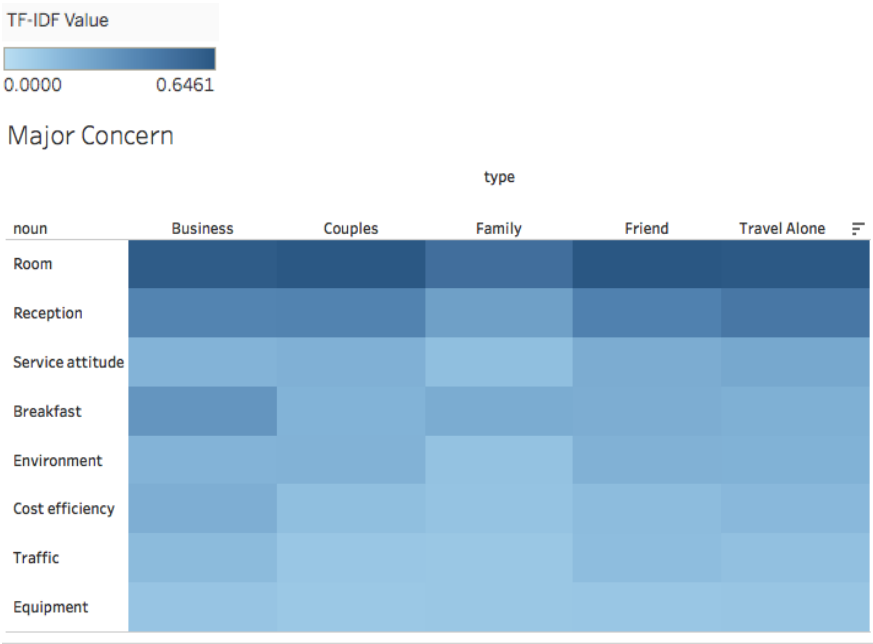
2. However, within these 20 major concerns, different types of customers present different focuses. Business travelers care most in terms of breakfast and cost-efficiency while couple consumers clearly care least for breakfast. Besides, family consumers obviously focus less on the major concern words like room, service attitude, reception and environment compared to other consumer types.

Table 7. Top 20 Concerns of Different Type Consumers of Three-Star Hotel (in index)

Major Concern

	type				
noun	Business	Couples	Family	Friend	Travel Alone
Room	0.6257	0.6450	0.5282	0.6461	0.6408
Reception	0.4195	0.4262	0.2885	0.4328	0.4822
Service attitude	0.1981	0.2093	0.1462	0.2301	0.2523
Breakfast	0.3369	0.1985	0.2326	0.2260	0.2125
Environment	0.1982	0.2016	0.1319	0.2067	0.2036
Cost efficiency	0.2195	0.1460	0.1285	0.1594	0.1740
Traffic	0.1625	0.1143	0.1089	0.1531	0.1388
Metro station	0.1107	0.1339	0.1205	0.1300	0.1332
Equipment	0.1217	0.1061	0.1084	0.1131	0.1161
Enthusiasm	0.0987	0.0855	0.0681	0.1085	0.1127
Metro	0.0920	0.0747	0.1086	0.0795	0.0927
Metro entrance	0.0758	0.0748	0.0986	0.0838	0.0887
Staff	0.0726		0.0605	0.0640	0.0746
Young lady		0.0520		0.0662	0.0707
Location	0.0698	0.0582	0.0597	0.0737	0.0704
Sound insulation	0.0611	0.0902		0.0546	0.0683
Silence	0.0772	0.0650	0.0529	0.0661	0.0616
Park	0.0744	0.0683	0.0813	0.0858	0.0607
Airport	0.0728	0.0568	0.0704	0.0617	0.0605
Variation					0.0523
Beach			0.0658		
Child			0.0693		
Experience	0.0513	0.0624			
Friend				0.0931	
Price	0.0552				

Table 8. Major Concerns of Different Type Consumers of Three-Star Hotel (in color)



3. What are the detailed concerns related to each major concern?

We use the word2vec.Word2Vec package in python to evaluate similarity of each word to the given major concern word. This package uses word embedding modelling method to vectorize each word in distributed word representation way. It calculates the cosine of angle between two word vectors and takes cosine value as similarity index. Higher similarity index means more similarity.

Because Word2vec training is an unsupervised task, there's no good way to objectively evaluate the result (Radim, 2014). Therefore, we apply trial and error approach by adjusting the parameters **min_count** (words appear less than min_count times will be ignored during the model training) and **size** (the size of NN layer corresponding to the degree of freedom the algorithm has) starting from the default value (min_count=5 and size=100). The performance is evaluated based on the correlation between the similar daily-used context of the words predicted by the model and the major concern word. The best parameters of model are as listed in table 9. The top 50 most similar words are selected as detailed words for each major concern (Detail data see "detail word.xlsx" in the attached folder "support_info") and first 5 detailed words are further summarized (appendix-A).

Table 9. Model parameter (min_count, size)

Business	Couples	Family	Friend	Travel Alone
(5,90)	(5,108)	(3,93)	(1,96)	(5,109)

4. What is the sentiment of each major concern of different consumer type?

Having collected 50 most similar words for each major concern word, we use the "SnowNLP" library in python to get the sentiment value of each detailed word. The range of the value is from 0 to 1 and higher value means more positive sentiment. To generate the sentiment value of certain major concerned word, we first multiplied each detailed word's sentiment value by its similarity index and then sum them together. We assume that the detail words with higher similarity to

major words can better represent the sentiment of major concern word. Based on this assumption, we generate the following equation:

$$sentiment\ of\ major\ word = \sum sentiment\ of\ detail\ word * its\ similarity$$

We can gain the following insights:

1. General Comparison

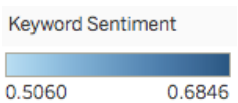
   Sentiments are different between consumer types. Overall, alone consumers show less positive sentiment in review while family travellers present the most positive sentiment (figure 10). Business consumers show the biggest difference within themselves, while couple consumers are more consistent (figure 10).

   Sentiments are also different between major concerns. Environment and service attitude show highest positive sentiment and price effectiveness shows higher variation on review sentiment (figure 9).

2. Detailed Insights

   Alone consumers write explicitly positive reviews on environment and room while they show least positive sentiment on equipment. Business travelling consumers show explicitly more positive sentiment on service attitude while they show less positive emotion on price effectiveness. Unlike other consumers, couple travelling consumers show little sentiment differences between different major concerns. Family travelling consumers show more positive sentiment on environment and price effectiveness while they are less positive when commenting on reception. Friend travelling consumers show positive emotion in reviews about the environment explicitly (Table 10 and Table 11).

Table 10. Sentiment of Major Concern Word (in color)



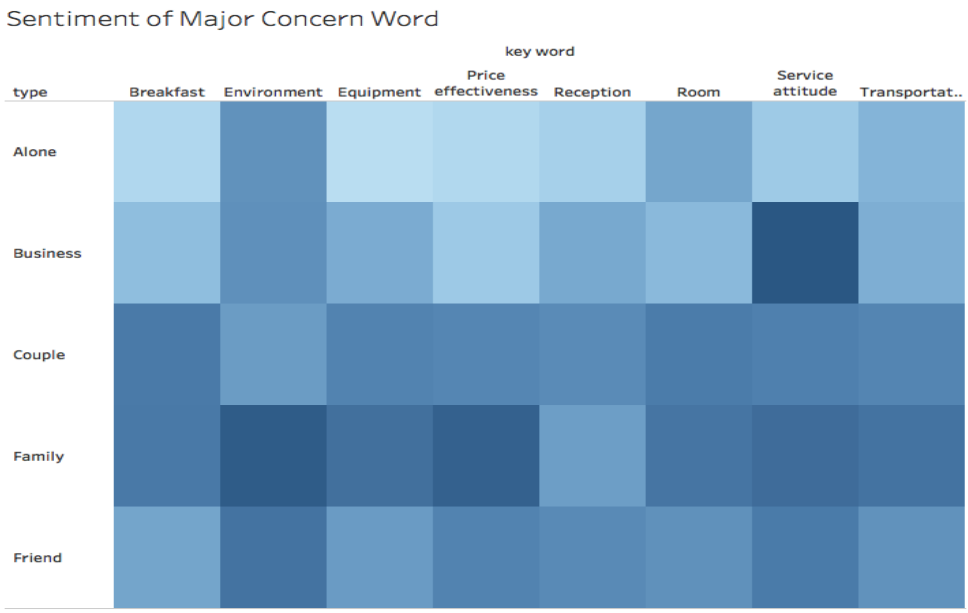
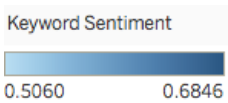


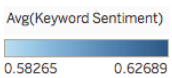
Table. 11 Sentiment of Major Concern Word (in index)



Sentiment of Major Concern Word

type	key word: Breakfast	Environment	Equipment	Price effectiveness	Reception	Room	Service attitude	Transportation
Alone	0.5146	0.6036	0.5060	0.5142	0.5248	0.5786	0.5327	0.5596
Business	0.5477	0.6058	0.5708	0.5335	0.5737	0.5529	0.6846	0.5672
Couple	0.6363	0.5908	0.6242	0.6195	0.6125	0.6330	0.6273	0.6202
Family	0.6374	0.6784	0.6504	0.6714	0.5886	0.6425	0.6552	0.6457
Friend	0.5792	0.6460	0.5920	0.6234	0.6136	0.6053	0.6347	0.6041

Figure 9. Average Sentiment Score (by major concern word)



(larger circle represent larger standard deviation of the score; the first number represent average sentiment score and the second number represent standard deviation)

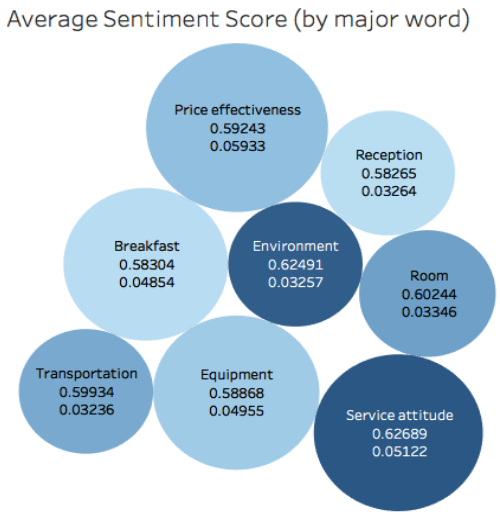
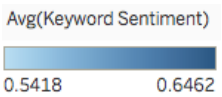
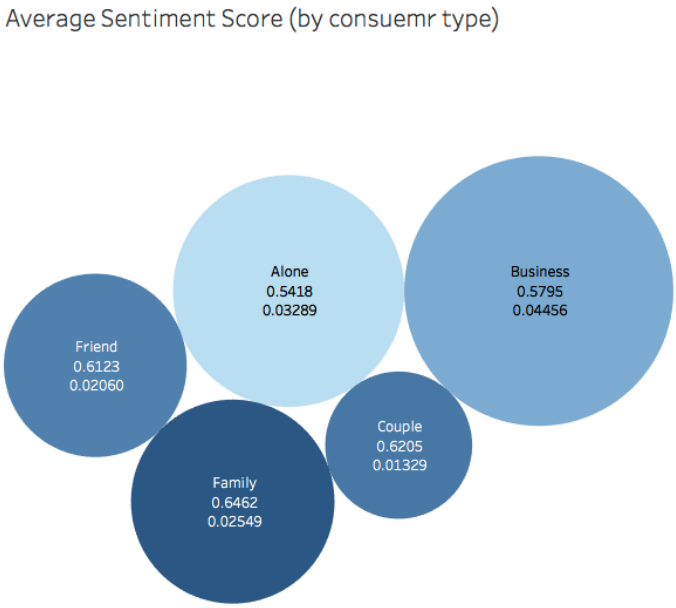


Figure 10. Average Sentiment Score (by consumer type)



(larger circle represents larger standard deviation of the score; the first number represents average sentiment score and the second number represents standard deviation)



Research Question 3

To consummate the review system from a hotel manager's perspective, a mechanism of review monitor is introduced. Review score given by every customer are useful quantitative indicators for hotel satisfaction. If potential connections between ratings are figured out, some models may be built which can give managers effective implications on how to improve satisfactions. We hypothesize that ratings are auto correlated and seasonality does not exist.

The analyses on the review scores are conducted both on single case and general situations. First, a hotel named Modern Classic Hotel Shenzhen is analyzed as one specific case to explore whether the detailed relationships in review score series fit some theoretical models. For general cases, the autocorrelation is detected in review scores of hotels which are grouped by hotel star levels. The programming language used is R (version 3.6.1). Because either seasonality or auto-correlation requires a time series analysis, the review scores are arranged by Date. If several ratings occur in the same day, an average will be taken.

1. A Specific Case

Before time-series analysis, seasonality as well as stationary series are checked for the data. If seasonality exists, a seasonally adjusted model should be applied after auto-correlation is detected. Since the average review scores of Modern Classic Hotel Shenzhen in each day show no seasonal trend in the series (figure 11), the autocorrelation analysis can be conducted without considering the seasonality. Then, a stationary series analysis is also required for time series analysis. After taking the first difference in the average ratings, the series presents a stationary property (figure 12). By applying the box-test to the stationary series for a general detection of autocorrelation, the p-value of the test is smaller than 5% which indicates an autocorrelation in the changes of average ratings (table 12).

Figure 11. Average ratings of Modern Classic Hotel Shenzhen

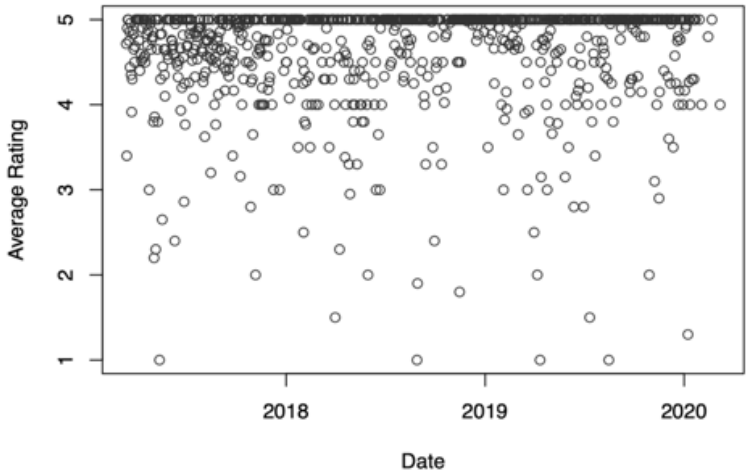


Figure 12. Changes in average ratings of Modern Classic Hotel Shenzhen

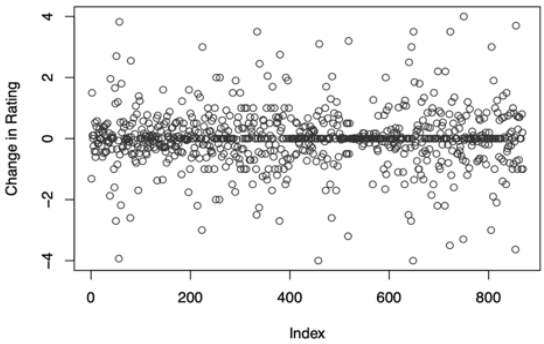


Table 12. Box test on the changes of average ratings

```
Box.test(deltaRate, lag = 10, type = c("Box-Pierce", "Ljung-Box"), fitdf = 0)
```

```
##
##  Box-Pierce test
##
## data:  deltaRate
## X-squared = 214.97, df = 10, p-value < 2.2e-16
```

To confirm the specific model to explain the relationship, ACF and PACF are plotted (figure 13). The two graphs suggest an MA(1) Model, which means the changes in average ratings are correlated with the white noise. The coefficients are regressed out (table 13). So, the changes in average ratings of Modern Classic Hotel Shenzhen are auto-correlated as the function indicates:

$\delta(Rt) = \varepsilon t - 0.9793\varepsilon t-1, \varepsilon t - i.i.d.N(0,\sigma 2)$

where,

$\delta(Rt)$ = daily change in average rating at period t, and

εt = the white noise at period t.

The model significance is checked by diagnosis (table 14). The p-value of the model residuals is 0.863, which is much larger than 5%, demonstrating a significance of the model.

Figure 13. ACF & PACF

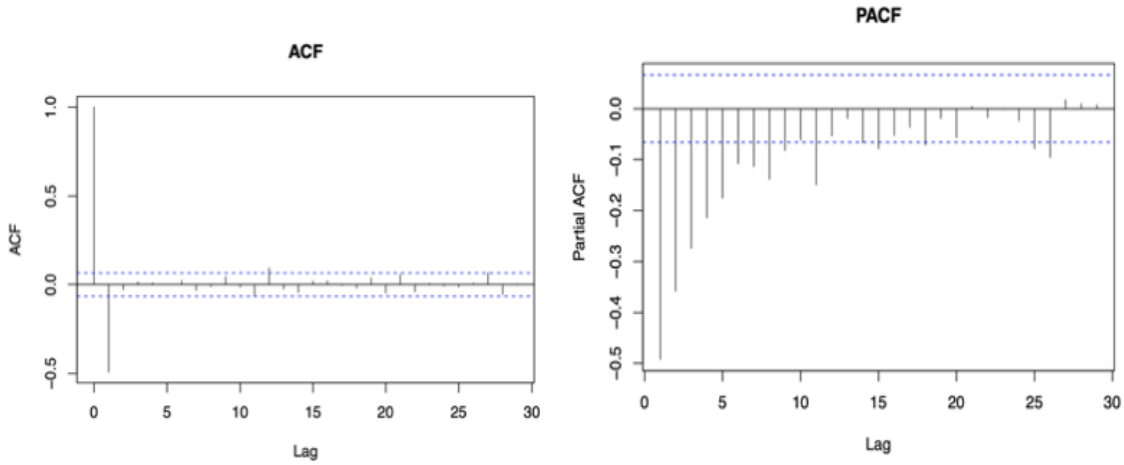


Table 13. Moving average model

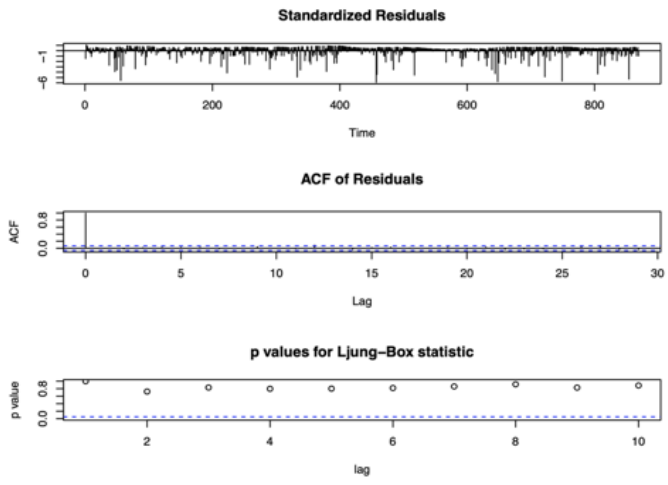
```
arima = arima(deltaRate, order = c(0, 0, 1))
arima
```

```
##
## Call:
## arima(x = deltaRate, order = c(0, 0, 1))
##
## Coefficients:
##           ma1  intercept
##       -0.9793      0e+00
## s.e.   0.0137      5e-04
##
## sigma^2 estimated as 0.4036:  log likelihood = -840.41,  aic = 1686.82
```

Table 14. Diagnosis

```
Box.test(arima$residuals, lag = 10, type = c("Box-Pierce", "Ljung-Box"), fitdf = 0)
```

```
##
##  Box-Pierce test
##
## data:  arima$residuals
## X-squared = 5.3988, df = 10, p-value = 0.863
```

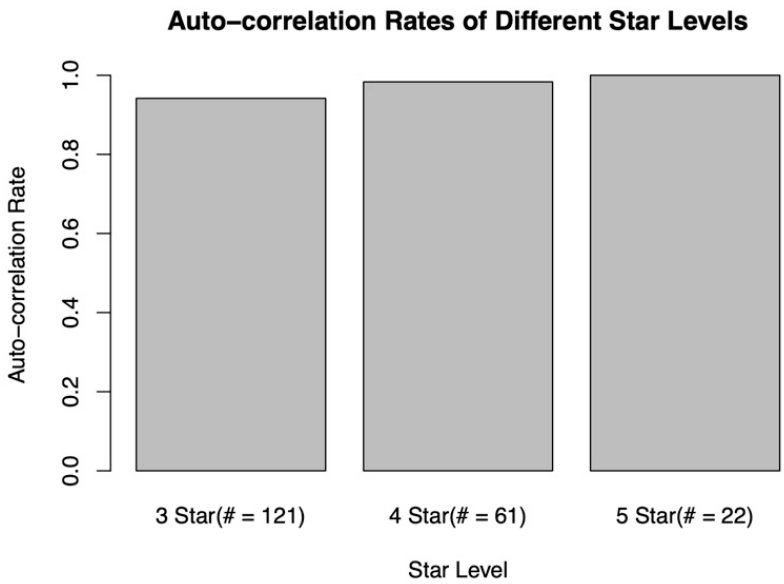


2. General Cases

Although the rating series of Modern Classic Hotel Shenzhen fits an MA(1) Model, the hotel ratings are auto-correlated and cannot be concluded from the single case. To find if the conclusion fits general cases, larger samples and further analysis are necessary. The hotels of Shenzhen are divided into three groups by their star levels. **Autocorrelation Rate** is defined as the probability for ratings of hotels belonging to a certain star level to be auto-correlated. **Autocorrelation Rate** of a star level can be calculated by the ratio of numbers of hotels being auto-correlated and the total number of hotels in that level.

The data are also cleaned as the process of the specific case. Two functions are defined to collect p-values of every hotel in each star level. Running the program for the three groups of data, **Autocorrelation Rate** of each star level is achieved (figure 14).

Figure 14. Auto-correlation rate



The graph of Modern Classic Hotel Shenzhen (figure 11) indicates there is no seasonality in the review score trend. The implication can be inferred to general cases because intuitively the ratings are based on the hotel itself, rather than the season. Since the seasonality plays as the indicator of modeling, it is not examined for general cases, but only for the specific case. For general cases, **Autocorrelation Rates** are calculated. Ratings of hotels present high

Autocorrelation Rates (over 90%) among all three levels. This demonstrated the fact that review scores of most hotels are autocorrelated. However, the number of samples decreases with the increase of star level, the statistical meaning of the result is weaker for high star level (i.e. five-star Hotel). Improvement can be made with increasing the sample size.

In the specific case of Modern Classic Hotel Shenzhen, the changes in average ratings follows an MA(1) Model: $\delta(Rt) = \varepsilon t - 0.9793\varepsilon t-1, \varepsilon t - i.i.d.N(0,\sigma 2)$. According to the World's Decomposition Theorem, the MA Model can be transformed into an AR(∞) Model, which means the current delta rating is correlated with the delta rating of every single day in the past.

Implication & Discussion

Our research is focused on how hotels can leverage the power of online reviews to improve customers' satisfaction. Starting with the correlation between hotel official description and hotel total scores, we then delve into the characteristics of useful reviews by first building a decision tree model to uncover how the number of pictures, review length , and review score distinguish a useful review. Next, we conduct text analysis on words frequently show up in those useful reviews. Then, we group reviews among different consumer types to analyze differences in consumer satisfaction. chi-square tests are conducted on consumer types and review scores. Then, frequent words and sentiments in reviews are derived for each consumer type. Finally, we check the auto-correlations in review scores for each hotel in our sample. To build a specific model as an example, we randomly pick one hotel and build up a time series model for it to predict for the future review scores.

Several useful implications on hotels' strategies for improving consumers' satisfaction are as follows. First, given a positive relationship between hotel official descriptions and hotel total scores, although the existence of causation requires further studies, improving hotels' official descriptions could be a beneficial strategy. One way is to include the most frequently mentioned information derived from

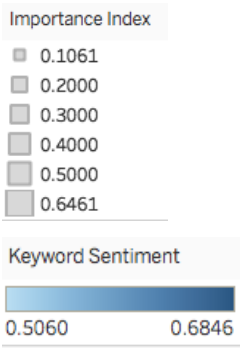
useful reviews, which are room facility and cleanness, service quality, transportation and location, and the food offered at the hotels based on our analysis. Additionally, reviews with photos and unextreme scores (i.e. not full score or zero score) tend to be regarded as useful by potential customers. This implicates hotels to guide customers to generate useful reviews by posting more photos and fair scores to enhance its authenticity. To do so, managers can utilize kind reminders posted on the hotel webpages, at the reception desk in the hotel, or on the reminder cards in the hotel rooms.

After grouping the reviews by customer types, the chi-square tests and box-plots indicate hotels to prioritize family travelers for three-star & five-star hotels and business travelers for four-star hotels when they want to improve customer satisfaction, so that they can maximize their efforts. Referring to the analysis on major concern, hotels can apply tailored strategies to their target customers. For example, hotels that focus on general consumers can work on room and reception improvement because they are the top 2 most concerned words for all types, and those that focus on business traveller should pay close attention to breakfast and price effectiveness. Referring to the analysis on detail concern word, hotels can spot what consumers concern most under certain major concern. For example, when hotels want to improve business consumer's reviews about rooms, they may try to start from size, space and design, etc.

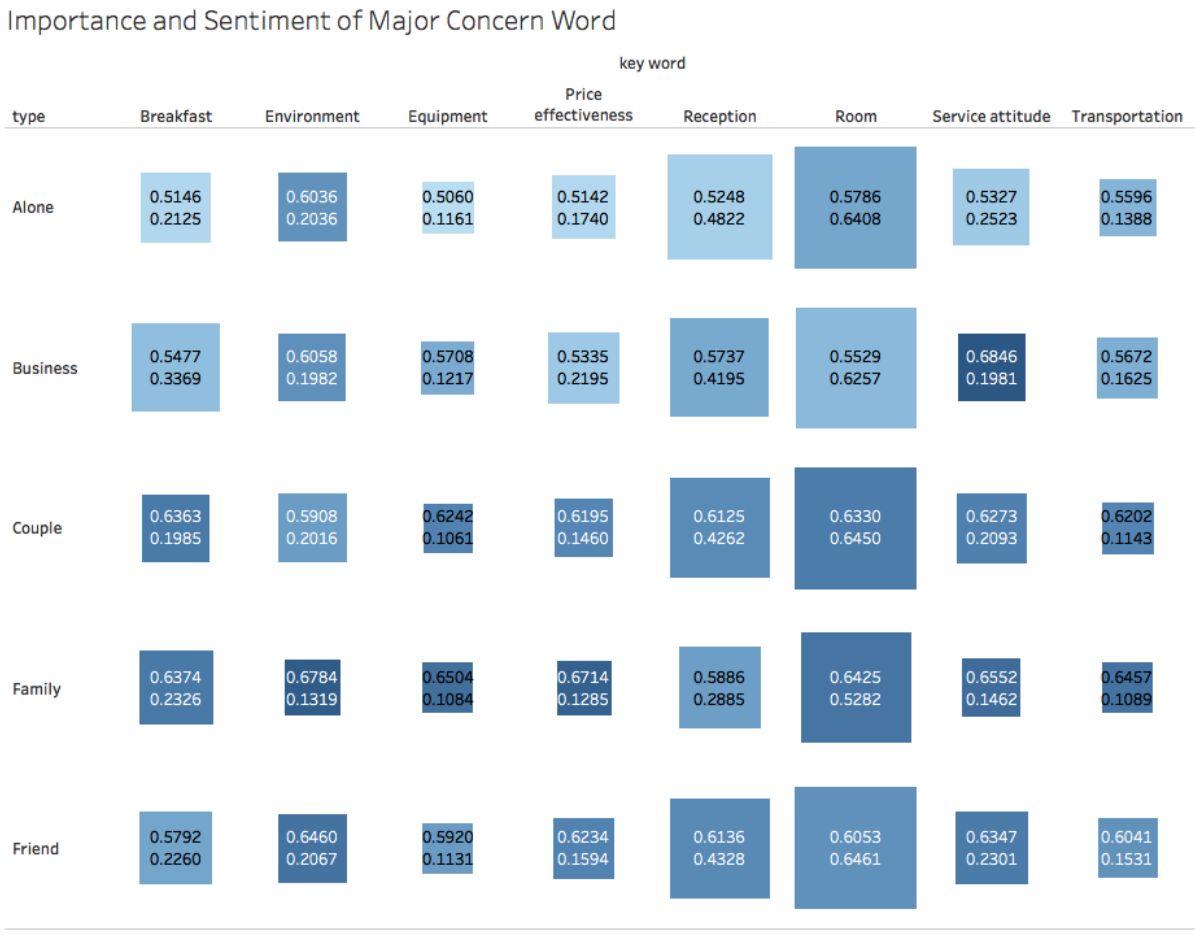
Referring to the analysis on sentiment, hotels can figure out the satisfaction level of their consumers on certain major concern and decide whether to make improvement. Hotels can also conduct marketing research on concerns with less positive sentiment. For example, hotels targeting family consumers can investigate why they present less positive sentiment on reviews related to reception and make improvement. Furthermore, hotels can combine our findings in concern words and sentiments when making strategies (table 15). For example, to make reviews more positive, hotel targeting at "Alone traveler" may want to improve their reception or room service which shows high importance and low positive sentiment in review. Then hotels can refer to the detail word table (appendix-A) to figure out what consumers concern most in terms of reception.

Finally, since review scores of most hotels are auto correlated, managers thus can monitor the ratings in every single day using times series modellings such as MA model in order to take actions in advance when customers' satisfaction is predicted to be low.

Table 15. Importance and sentiment of major concern word



(the first row represent sentiment score; the second row represent important index)



Limitation

Firstly, given the limited sample, deviation of our results may occur when applied to the population, which can be mitigated by increasing the sample size. Secondly, whether there is a causation between hotel official descriptions and hotel total scores remain unsolved in this paper and need to be further studied. It is possible that our conclusion needs adjustment after the causation analysis. Thirdly, due to the limited space, only reviews of three-star hotels are conducted as an example to investigate concerns from different consumer types, while reviews of four-star and five-star hotels may add more interesting and useful insights.

Conclusion

Hotels have many ways to take advantage of powerful online reviews to improve consumers' satisfaction. Our analysis shows that first hotels can derive what kind of information is regarded as useful by viewers so that hotels can provide these information as official hotel descriptions. Hotels can also derive concerns from different consumer types so that services can be tailored for each different type. Hotels can also build early warning model indicating consumers' dissatisfaction so that hotels can take actions in advance.

Online reviews of hotels contain lots of information that are helpful not only for users' decision-making, but also for improvement of hotel performance. However, due to the inappropriate monitoring, it is possible to manipulate misleading information. Thus, both hotels and users should use these reviews carefully, while online review platforms should also try to strengthen supervision and increase the credibility.

References

Hu, F. (2019). What makes a hotel review helpful? An information requirement perspective. Journal of Hospitality Marketing & Management, 1–21. https://doi.org/10.1080/19368623.2019.1661931

Kwok, L., Xie, K. L., & Richards, T. (2017). Thematic framework of online review research: A systematic analysis of contemporary literature on seven major hospitality and tourism journals. International Journal of Contemporary Hospitality Management, 29(1), 307–354.

Kwok, L. & Xie, K.L. (2016). Factors contributing to the helpfulness of online hotel reviews: does manager response play a role? International Journal of Contemporary Hospitality Management, 28(11), 2156-2177.

Ladhari, R. & Michaud, M. (2015), EWOM effects on hotel booking intentions, attitudes, trust, and website perceptions", International Journal of Hospitality Management, 46, 36-45.

Lee, W., Xiong, L. & Hu, C. (2012). The effect of Facebook users' arousal and valence on intention to go to the festival: applying an extension of the technology acceptance model", International Journal of Hospitality Management, 31(3), 819-827.

Liu, Z. & Park, S. (2015). What makes a useful online review? Implication for travel product websites", Tourism Management, 47, 140-151.

Litvin, S. W., Goldsmith, R. E., & Pan, B. (2018). A retrospective view of electronic word-of-mouth in hospitality and tourism management. International Journal of Contemporary Hospitality Management, 30(1). 313–325. https://doi.org/10.1108/IJCHM-08-2016-0461

Ma, Y., Xiang, Z., Du, Q., Fan, W. (2018). Effects of user-provided photos on hotel review helpfulness: an analytical approach with deep learning. International Journal of Hospitality Management, 120-131.

Papathanassis, A., & Knolle, F. (2011). Exploring the adoption and processing of online holiday reviews: a grounded theory approach. Tourism Management. 32, 215–224.

Park, S. & Nicolau, J.L. (2015). Asymmetric effects of online consumer reviews. Annals of Tourism Research, 50, 67-83.

Redmond, M. V. (2015). Uncertainty Reduction Theory. In *English Technical Reports and White Papers*. https://lib.dr.iastate.edu/engl_reports/3

TripAdvisor. (2018, January 25). 5 Tips Inspired by Our New Traveler Survey. TripAdvisor Insights. https://www.tripadvisor.com/TripAdvisorInsights/w661

Xie, K.L., Zhang, Z. & Zhang, Z. (2014). The business value of online consumer reviews and management response to hotel performance. International Journal of Hospitality Management, 43, 1-12.

Zehrer, A., Crotts, J.C. & Magnini, V.P. (2011). The perceived usefulness of blog postings: an extension of the expectancy-disconfirmation paradigm. Tourism Management, 32(1), 106-113.

Zhang, Z., Zhang, Z., & Yang, Y. (2016). The power of expert identity: How website-recognized expert reviews influence travelers' online rating behavior. Tourism Management, 55. 15-24.

Zhao, X., Wang, L., Guo, X. & Law, R. (2015). The influence of online reviews to online hotel booking intentions. International Journal of Contemporary Hospitality Management, 27(6), 1343-1364.

Rare-technologies.com. 2020. Word2vec Tutorial | RARE Technologies. [online] Available at: <https://rare-technologies.com/word2vec-tutorial/> [Accessed 10 May 2020].

Appendix

Appendix-A

3-star detailed word summary

	Business	Couple	Friend	Family	Alone
Room	Size New Equipment Space Design	Neat Overall New Like it Satisfied	Comfortable Equipment New Warm Reception	Comfortable New Satisfied Equipment Neat	Comportable Equipment Like Live Metro

Reception	Attitude Server Young lady Enthusiasm Staff	Enthusiasm Service Attitude Like Nice	Room Attitude Staff Enthusiasm Service attitude	Staff Attitude Server Thorough Recieve	Enthusiasm Service attitude Room Server Attitude
Service attitude	Very good Thorough service Nice Come back again	Enthusiasm Worth Recommend Reception Experience	Satisfied Comfortable Room Quite Location	Satisfied Comfortable Environment Neat Location	Room Overall Reception Outing Attitude
Breakfast	Diversity Delicious Environment Condition Price	Experience Nice Delicious Super Live	Live Price Eat Option Location	Option Overall Entire Location Nice	Surrounding Option Walk Price Special
Environment	Nice Overall Good Not bad As usual	Sanitary Clean Comfortable Nice Good	Service Clean Sanitary Comfortable Neat	Service attitude Comfortable Neat Sanitary Satisfied	Satisfied Live Special Comfortable Overall
Price Effectiveness	High Super High Price Benefit Overall	High Service attitude Comfortable Recommend Worth	Facility Room Neat Comfortabe Recommend	Recommend Environment Position High Compfortable	Live High Breakfast Location Check in

Trasnportation	Convenience Easy to find Position Meal Metro station	Location Conenience Surrounding Overall Attitude	Position Outing Room Easy to find Metro entrance	Position Convenience Outing Metro entrnce Metro	Position Room Location Surrounding Option
Equipment	New Thorough Facility Clean Room	live Facility Overall Design Entire	Room Large Design New Warm	New Hotel facility Special Room Satisfied	Parking Design Suitable Overall New

Code

Research Question 1

STATA code analyzing correlation between hotel official description and hotel total score

```
program define pwcorr_a, byable(recall)
    version 6
    syntax [varlist(min=2)] [if] [in] [aw fw] [, /*
       */ Bonferroni Obs Print(real -1) SIDak SIG star1(real 0.01) star5(real 0.05) star10(real 0.1) ]
    tempvar touse
    mark `touse' `if' `in'      /* but do not markout varlist */
    tokenize `varlist'

    local i 1
    while "``i''" != "" {
       capture confirm str var ``i''
       if _rc==0 {
          di in gr "(``i'' ignored because string variable)"
          local `i' " "
       }
       local i = `i' + 1
    }
    local varlist `*'
    tokenize `varlist'
    local nvar : word count `varlist'
    if `nvar' < 2 { error 102 }

    local weight "[`weight'`exp']"
    local nvar : word count `varlist'
    local adj 1
```

```
   if "`bonferr'"!="" | "`sidak'"!="" {
      if "`bonferr'"!="" & "`sidak'"!="" { error 198 }
      local nrho=(`nvar'*(`nvar'-1))/2
      if "`bonferr'"!="" { local adj `nrho' }
   }
/*
   if (`star'>=1) {
      local star = `star'/100
      if `star'>=1 {
         di in red "star() out of range"
         exit 198
      }
   }
*/
   if (`print'>=1) {
      local print = `print'/100
      if `print'>=1 {
         di in red "print() out of range"
         exit 198
      }

   }

   local j0 1
   while (`j0'<=`nvar') {
      di
      local j1=min(`j0'+6,`nvar')
      local j `j0'
      di in smcl in gr _skip(13) "{c |}" _c
      while (`j'<=`j1') {
         di in gr %9s abbrev("``j''",8) _c
         local j=`j'+1
      }
      local l=9*(`j1'-`j0'+1)
      di in smcl in gr _n "{hline 13}{c +}{hline `l'}"

      local i `j0'
      while `i'<=`nvar' {
         di in smcl in gr %12s abbrev("``i''",12) " {c |} " _c
         local j `j0'
         while (`j'<=min(`j1',`i')) {
            cap corr ``i'' ``j'' if `touse' `weight'
            if _rc == 2000 {
               local c`j' = .
            }
            else {
               local c`j'=r(rho)
            }
            local n`j'=r(N)
            local p`j'=min(`adj'*tprob(r(N)-2,/*
            */ r(rho)*sqrt(r(N)-2)/ /*
            */ sqrt(1-r(rho)^2)),1)
            if "`sidak'"!="" {
               local p`j'=min(1,1-(1-`p`j'')^`nrho')
            }
            local j=`j'+1
         }
         local j `j0'

**********************************************
         while (`j'<=min(`j1',`i')) {
```

```
            if (`p`j''<=`star1'&`i'!=`j'){
                  local ast "***"
            }
            else if (`p`j''>`star1'&`p`j''<=`star5'&`i'!=`j'){
                  local ast "** "
            }
            else if (`p`j''>`star5'&`p`j''<=`star10'&`i'!=`j'){
                  local ast "*  "
            }

      else local ast "   "
      if `p`j''<=`print' | `print'==-1 |`i'==`j' {
//       di " " %7.4f `c`j'' "`ast'" _c
                  di _col(1) %7.3f `c`j'' "`ast'" _col(2) _c
            }
            else    di _skip(9) _c
            local j=`j'+1
          }

    **************************************************
          di
          if "`sig'"!="" {
            di in smcl in gr _skip(13) "{c |}" _c
            local j `j0'
            while (`j'<=min(`j1',`i'-1)) {
              if `p`j''<=`print' | `print'==-1 {
                di "   " %7.4f `p`j'' _c
              }
              else    di _skip(9) _c
              local j=`j'+1
            }
            di
          }
          if "`obs'"!="" {
            di in smcl in gr _skip(13) "{c |}" _c
            local j `j0'
            while (`j'<=min(`j1',`i')) {
              if `p`j''<=`print' | `print'==-1 /*
              */ |`i'==`j' {
                di "  " %7.0g `n`j'' _c
              }
              else    di _skip(9) _c
              local j=`j'+1
            }
            di
          }
          if "`obs'"!="" | "`sig'"!="" {
            di in smcl in gr _skip(13) "{c |}"
          }
          local i=`i'+1
        }
        local j0=`j0'+7
      }
    end


    "/DMS 2051/Project/Description.dta"
    egen mean=mean(official_photo_count)
    sum official_photo_count
    gen standard_photo=(official_photo_count-mean)/48.13008
```

```
gen complexity=
locality+transportation+local_business+restaurant+business_meeting+fitness_entertainment+landscape+special_culture_idea+laundry+decoration+experience_expectation+standard_photo
pwcorr_a hotel_score locality transportation local_business restaurant business_meeting fitness_entertainment landscape special_culture_idea laundry decoration experience_expectation official_photo_count complexity standard_photo,star1(0.01) star5(0.05) star10(0.1)
```

Python code analyzing useful reviews' text

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_boston
from sklearn.linear_model import LinearRegression, SGDRegressor, Ridge
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.feature_selection import mutual_info_classif
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
import statsmodels.formula.api as smf

from sklearn.tree import DecisionTreeClassifier,export_graphviz
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
ctrip = pd.read_excel('携程酒店评论.xlsx')
ctrip = ctrip.drop(columns = ['reply','travel_type','address','user_detail'])

#review length
ctrip['review_len'] = ''
for i in range(0, ctrip.shape[0]):
    if pd.notnull(ctrip.at[i,'review']):
        review = str(ctrip.at[i,'review'])
        ctrip.at[i,'review_len'] = len(review)
    else:
        ctrip.at[i,'review_len'] = 0

#pic number
ctrip['pic_num'] = ''
photo = ["photo9","photo8","photo7","photo6","photo5","photo4","photo3","photo2","photo1"]
for i in range(0, ctrip.shape[0]):
    j = 0
    while j < len(photo):
        if pd.notnull(ctrip.at[i,photo[j]]):
            ctrip.at[i,'pic_num'] = photo[j][-1]
            break
        j += 1
    else:
        ctrip.at[i,'pic_num'] = 0

#save data
ctrip.to_csv("ctrip_data.csv",index=False,sep=',')
data1 = pd.DataFrame(ctrip[['score','review_len','pic_num','usefulness']])
data1['review_len'] = data1['review_len'].astype('str').astype('int')
data1['pic_num'] = data1['pic_num'].astype('int')
data1['useful_ornot']=''
data1 = data1.fillna(0)
for i in range(0, data1.shape[0]):
    if data1.at[i,'usefulness'] > 0:
        data1.at[i, 'useful_ornot'] = 1
    else:
        data1.at[i, 'useful_ornot'] = 0
```

```
data1['useful_ornot']= data1['useful_ornot'].astype('int')
data1.to_csv("data1.csv",index=False,sep=',')
from sklearn.utils import shuffle
print(data1['useful_ornot'].value_counts())
useful = data1.loc[data1['useful_ornot'] == 1]
notuseful = data1.loc[data1['useful_ornot'] == 0][:236]
normal_distributed = pd.concat([useful, notuseful])
normal_distributed = normal_distributed.drop(columns = ['usefulness'])
new_df = normal_distributed.sample(frac = 1, random_state = 42)
print(normal_distributed.describe())
print(normal_distributed.corr())
import seaborn as sns
f, (ax2) = plt.subplots(figsize = (10, 8),nrows=1)
new_corr = new_df.corr()
sns.heatmap(new_corr,cmap = 'coolwarm_r', annot_kws = {'size' : 20})
ax2.set_title('Balanced correlation matrix', fontsize = 14)
plt.show()

from imblearn.over_sampling import SMOTE
y = data1['useful_ornot'].astype('int')
x = data1[[x for x in data1.columns if x not in ['useful_ornot','usefulness']]]
X_train,X_test,Y_train,Y_test = train_test_split(x, y, train_size=0.3, random_state=100)

# balance the data
over_samples = SMOTE(random_state=1234)
over_samples_X,over_samples_Y = over_samples.fit_sample(X_train, Y_train)

# Proportion of categories before resampling
print(Y_train.value_counts()/len(Y_train))
# Proportion of categories after resampling
print(pd.Series(over_samples_Y).value_counts()/len(over_samples_Y))

from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
# List of values to try for max_depth:
max_depth_range = list(range(1, 30))
# List to store the average RMSE for each value of max_depth:
mse_train = []
mse_test = []
for depth in max_depth_range:
    grid_search_mse = GridSearchCV(DecisionTreeClassifier(class_weight='balanced',max_depth =
depth, random_state=100),param_grid,cv=5)
    grid_search_mse.fit(over_samples_X,over_samples_Y)
    best_model_mse = grid_search_mse.best_estimator_
    yhat1_mse = best_model_mse.predict(over_samples_X)
    yhat2_mse = best_model_mse.predict(X_test)
    mse_train.append(mean_squared_error(over_samples_Y, yhat1_mse, sample_weight=None,
multioutput='uniform_average'))
    mse_test.append(mean_squared_error(Y_test, yhat2_mse, sample_weight=None,
multioutput='uniform_average'))

        plt.title('MSE')
        plt.plot(max_depth_range, mse_train, color='green', label='training set')
        plt.plot(max_depth_range, mse_test, color='skyblue', label='test set')
        plt.legend()
        plt.xlabel('max_depth')
        plt.ylabel('MSE')
        plt.show()

        from sklearn import metrics
        # Reconstruction of decision tree model based on balanced data
```

```
        dt2 = DecisionTreeClassifier(random_state=100)
        dt2.fit(over_samples_X,over_samples_Y)
        # Prediction of model on test set
        pred2 =dt2.predict(X_test)
        # Model evaluation report
        print(classification_report(Y_test, pred2))

        #prune the tree
        param_grid={'max_leaf_nodes':list(range(2,100)),
                'min_samples_split':[2,3,4],
                'max_depth':[13]}
        grid_search_os = GridSearchCV(DecisionTreeClassifier(class_weight='balanced',random_state=100),param_grid,cv=5,scoring='recall')
        grid_search_os.fit(over_samples_X,over_samples_Y)
        print("Test set score: {:.2f}".format(grid_search_os.score(X_test,Y_test))) # 打印得分
        print("Best parameters:{}".format(grid_search_os.best_params_)) # 打印最好的参数组合
        print("Best score on train set: {:.2f}".format(grid_search_os.best_score_))  # 打印最好的得分

        #optimize the model
        best_model_os = grid_search_os.best_estimator_
        yhat2_os = best_model_os.predict(X_test)
        print(classification_report(Y_test,yhat2_os))
        best_model_os

        #most important feature
        feat_importance_os = best_model_os.tree_.compute_feature_importances()
        print('Importance of all features:\n',feat_importance_os)
        mostImportantFeature_os = x.columns[list(feat_importance_os).index(feat_importance_os.max())]
        print('\nThe most importance features is:',mostImportantFeature_os)

        #draw the tree
        import graphviz
        dot_data_os=export_graphviz(best_model_os, out_file=None,feature_names=['review_score','review_len','pic_num'],class_names = True, filled = True,special_characters=True)
        graph = graphviz.Source(dot_data_os)
        graph

        #export the tree
        import pydotplus
        graph = pydotplus.graph_from_dot_data(dot_data_os)
        graph.write_pdf("test.pdf")

        from sklearn.svm import LinearSVC
        from yellowbrick.classifier import ROCAUC

        model = LinearSVC()
        model.fit(X,y)
        visualizer = ROCAUC(model)
        visualizer.score(X,y)
        visualizer.poof()

        import numpy as np
        import pandas as pd
        ctrip = pd.read_excel('携程酒店评论.xlsx')
        useful_ctrip=ctrip.loc[ctrip['usefulness'] >= 1]
        useful_ctrip.head()
        useful_creview = useful_ctrip[['review']]
        useful_creview = useful_creview.dropna()
        useful_creview = useful_creview[~useful_creview['review'].str.contains("用户已好评")]
```

```
        useful_creview = useful_creview[~useful_creview['review'].str.contains("用户已中评")]
        useful_creview = useful_creview[~useful_creview['review'].str.contains("用户已差评")]
        useful_creview.reset_index(drop=True,inplace=True)
        print(useful_creview)

        stopwords = pd.read_csv("baidu_stopwords.txt", index_col=False, quoting=3, sep='\t',
    names=['stopword'], encoding='utf-8')
        stopwords = list(stopwords['stopword'].values)
        print(stop words)

        # Convert text documents to a matrix of token counts.
        from sklearn.feature_extraction.text import TfidfVectorizer
        tfidf2 = TfidfVectorizer(max_df = 0.9, stop_words = stopwords)
        C = tfidf2.fit_transform(useful_creview['review'])

        # tokenize the words in review
        import nltk
        from nltk.tokenize import word_tokenize
        #nltk.download('punkt')
        tokenized_df_c = useful_creview.apply(lambda row: nltk.word_tokenize(row['review']),
    axis=1)
        tokenized_words_c = [w for s in tokenized_df_c for w in s]
        #tokenized_words = [word_tokenize(i) for i in useful_review['review']]
        #delete the stopwords from the set
        filtered_words_c = [w for w in tokenized_words_c if not w in stopwords]

        from wordcloud import WordCloud
        # world cloud for the whole review
        text = " ".join(filtered_words_c)

        wc = WordCloud(font_path='/System/Library/Fonts/Hiragino Sans
    GB.ttc',max_font_size=50, max_words=100, background_color="white",
    collocations=False).generate(text)

        plt.figure()
        plt.imshow(wc, interpolation="bilinear")  # visulize worldcloud
        plt.axis('off')  # close axis
        wc.to_file("/Users/ashley/Desktop/DMS2051/project/temp.jpg")
        plt.show() # show image

        import yellowbrick
        from sklearn.cluster import KMeans
        from sklearn.datasets import make_blobs

        from yellowbrick.cluster import KElbowVisualizer
        # Instantiate the clustering model and visualizer
        model = KMeans()
        visualizer = KElbowVisualizer(model, k=(1,10))

        visualizer.fit(C)       # Fit the data to the visualizer
        visualizer.show()       # Finalize and render the figure
```

Research question 2

R code for chi-sqaure tests and box-pots

```
setwd('/Users/apple/Desktop/DMS_2051/Data/DA7A/data')
library(dplyr)
library(tidyr)
library(ggplot2)

```

```r
ctrip3=read.csv('ctrip_3_sample_review.csv',stringsAsFactors = F)
ctrip3=ctrip3%>%
  mutate(round_score=round(review_score))%>%
  filter(!is.na(travel_type))%>%
  filter(!is.na(round_score))%>%
  subset(travel_type!="其它" & travel_type!="代人预订")%>%
  mutate(travel_type=ifelse(travel_type=='情侣出游', 'couple',travel_type))%>%
  mutate(travel_type=ifelse(travel_type=='家庭亲子', 'family',travel_type))%>%
  mutate(travel_type=ifelse(travel_type=='商务出差', 'business',travel_type))%>%
  mutate(travel_type=ifelse(travel_type=='独自旅行', 'alone',travel_type))%>%
  mutate(travel_type=ifelse(travel_type=='朋友出游', 'friend',travel_type))

ctrip3_table=table(ctrip3$travel_type, ctrip3$round_score)
type_names=row.names(ctrip3_table)
type_names_comb=data.frame(type_names,stringsAsFactors = F)
type_names_comb=type_names_comb%>%
  merge(type_names_comb,by=NULL)
chi_p=data.frame()
ctrip3_table=t(ctrip3_table)
for(i in 1:ncol(ctrip3_table)){
  for(j in 1:ncol(ctrip3_table)){
    test=ctrip3_table[1:5,c(i,j)]
    rst=chisq.test(test)$p.value
    chi_p=rbind(chi_p,rst)
  }
}
type_names_comb=cbind(type_names_comb,chi_p)
colnames(type_names_comb)=c('type1','type2','chi_p')
ctrip3_type_chi=type_names_comb%>%
  filter(type1!=type2)%>%
  distinct(chi_p,.keep_all=T)%>%
  mutate(similar=ifelse(chi_p<=0.01,F,T))%>%
  arrange(chi_p)
ctrip3_type_chi
ctrip3_type_plot=ctrip3%>%
  mutate(travel_type=as.factor(travel_type))
#levels(ctrip3_type_plot$travel_type)=
#  c('alone','family','friend','couple','business')
ctrip3_type_plot=ctrip3_type_plot%>%
  mutate(travel_type=as.character(travel_type))%>%
  mutate(travel_type=ifelse((travel_type=='friend' | travel_type == 'couple' | travel_type ==
'alone'),'friend & couple & alone', travel_type))
boxplot(review_score~travel_type,data=ctrip3_type_plot,
        xlab='consumer type',ylab='rounded review score')


#mu=ctrip3_type_plot%>%
#  group_by(travel_type)%>%
#  summarize(mu_score=mean(round_score))
mu_count_p=ctrip3_type_plot%>%
  group_by(travel_type)%>%
  mutate(type_n=n(),mu_score=mean(round_score))%>%
  ungroup()%>%
  group_by(travel_type,round_score,mu_score,type_n)%>%
  mutate(p=n()/type_n)%>%
  summarize(p=first(p))%>%
  ungroup()

ctrip3_type_plot2=mu_count_p%>%
  ggplot(aes(x=round_score,y=p,color=travel_type))+
  geom_point()+
```

```
  geom_vline(aes(xintercept=mu_score, color=travel_type),
             linetype="dashed",size=0.5)+
#  geom_histogram(binwidth =1,
#                 alpha=0.2,position='identity',fill='white')+
#  geom_vline(data=mu, aes(xintercept=mu_score, color=travel_type),
#             linetype="dashed",size=0.5)+
  scale_color_manual(values = c("#CC79A7","#999999", "#D55E00",
                                "#56B4E9", "#009E73"
                                #,"#000000","#C3D7A4"
                                )) +
  labs(x='round_score',y='count_reviews')+
  theme_bw()

print(ctrip3_type_plot2)
print(mu_count_p%>%arrange(-mu_score))
```

###Ctrip 4 analysis

```{r}
ctrip4=read.csv('ctrip_4_sample_review.csv',stringsAsFactors = F)
ctrip4=ctrip4%>%
  mutate(round_score=round(review_score))%>%
  filter(!is.na(travel_type))%>%
  filter(!is.na(round_score))%>%
  subset(travel_type!="其它" & travel_type!="代人预订")%>%
  mutate(travel_type=ifelse(travel_type=='情侣出游', 'couple',travel_type))%>%
  mutate(travel_type=ifelse(travel_type=='家庭亲子', 'family',travel_type))%>%
  mutate(travel_type=ifelse(travel_type=='商务出差', 'business',travel_type))%>%
  mutate(travel_type=ifelse(travel_type=='独自旅行', 'alone',travel_type))%>%
  mutate(travel_type=ifelse(travel_type=='朋友出游', 'friend',travel_type))
ctrip4_table=table(ctrip4$travel_type, ctrip4$round_score)
type_names=row.names(ctrip4_table)
type_names_comb=data.frame(type_names,stringsAsFactors = F)
        type_names_comb=type_names_comb%>%
          merge(type_names_comb,by=NULL)
        chi_p=data.frame()
        ctrip4_table=t(ctrip4_table)
        ctrip4_table=ctrip4_table[2:nrow(ctrip4_table),1:ncol(ctrip4_table)]#only one review scored
  0
        for(i in 1:ncol(ctrip4_table)){
          for(j in 1:ncol(ctrip4_table)){
            test=ctrip4_table[1:nrow(ctrip4_table),c(i,j)]
            rst=chisq.test(test)$p.value
            chi_p=rbind(chi_p,rst)
          }
        }
        type_names_comb=cbind(type_names_comb,chi_p)
        colnames(type_names_comb)=c('type1','type2','chi_p')
        ctrip4_type_chi=type_names_comb%>%
          filter(type1!=type2)%>%
          distinct(chi_p,.keep_all=T)%>%
          mutate(similar=ifelse(chi_p<=0.01,F,T))%>%
          arrange(chi_p)
        ctrip4_type_chi
        ctrip4_type_plot=ctrip4%>%
          mutate(travel_type=as.factor(travel_type))
        #levels(ctrip4_type_plot$travel_type)=
        #  c('alone','family','friend','couple','business')
        ctrip4_type_plot=ctrip4_type_plot%>%
          mutate(travel_type=as.character(travel_type))%>%
```

```
        mutate(travel_type=ifelse((travel_type=='friend' | travel_type == 'couple' | travel_type == 'alone'),'friend & couple & alone', travel_type))
      boxplot(review_score~travel_type,data=ctrip4_type_plot,
           xlab='consumer type',ylab='rounded review score')


      mu_count_p=ctrip4_type_plot%>%
       group_by(travel_type)%>%
       mutate(type_n=n(),mu_score=mean(round_score))%>%
       ungroup()%>%
       group_by(travel_type,round_score,mu_score,type_n)%>%
       mutate(p=n()/type_n)%>%
       summarize(p=first(p))%>%
       ungroup()
      ctrip4_type_plot2=mu_count_p%>%
       ggplot(aes(x=round_score,y=p,color=travel_type))+
       geom_point()+
       geom_vline(aes(xintercept=mu_score, color=travel_type),
                  linetype="dashed",size=0.5)+
      #  geom_histogram(binwidth =1,
      #                 alpha=0.2,position='identity',fill='white')+
      #  geom_vline(data=mu, aes(xintercept=mu_score, color=travel_type),
      #             linetype="dashed",size=0.5)+
       scale_color_manual(values = c("#CC79A7","#999999", "#D55E00",
                                     "#56B4E9", "#009E73"
                                     #,"#000000","#C3D7A4"
                                     )) +
       labs(x='round_score',y='count_reviews')+
       theme_bw()
      print(ctrip4_type_plot2)
      print(mu_count_p%>%arrange(-mu_score))
      ```

      ###Ctrip 5 analysis

      ```{r}
      ctrip5=read.csv('ctrip_5_sample_review.csv',stringsAsFactors = F)
      ctrip5=ctrip5%>%
       mutate(round_score=round(review_score))%>%
       filter(!is.na(travel_type))%>%
       filter(!is.na(round_score))%>%
       subset(travel_type!="其它" & travel_type!="代人预订")%>%
       mutate(travel_type=ifelse(travel_type=='情侣出游', 'couple',travel_type))%>%
       mutate(travel_type=ifelse(travel_type=='家庭亲子', 'family',travel_type))%>%
       mutate(travel_type=ifelse(travel_type=='商务出差', 'business',travel_type))%>%
       mutate(travel_type=ifelse(travel_type=='独自旅行', 'alone',travel_type))%>%
       mutate(travel_type=ifelse(travel_type=='朋友出游', 'friend',travel_type))
      ctrip5_table=table(ctrip5$travel_type, ctrip5$round_score)
      type_names=row.names(ctrip5_table)
      type_names_comb=data.frame(type_names,stringsAsFactors = F)
      type_names_comb=type_names_comb%>%
       merge(type_names_comb,by=NULL)
      chi_p=data.frame()
      ctrip5_table=t(ctrip5_table)
      ctrip5_table=ctrip5_table[2:nrow(ctrip5_table),1:ncol(ctrip5_table)]#only one review scored 0
      for(i in 1:ncol(ctrip5_table)){
       for(j in 1:ncol(ctrip5_table)){
        test=ctrip5_table[1:nrow(ctrip5_table),c(i,j)]
        rst=chisq.test(test)$p.value
        chi_p=rbind(chi_p,rst)
```

```
        }
      }
      type_names_comb=cbind(type_names_comb,chi_p)
      colnames(type_names_comb)=c('type1','type2','chi_p')
      ctrip5_type_chi=type_names_comb%>%
        filter(type1!=type2)%>%
        distinct(chi_p,.keep_all=T)%>%
        mutate(similar=ifelse(chi_p<=0.01,F,T))%>%
        arrange(chi_p)
      ctrip5_type_chi
      ctrip5_type_plot=ctrip5%>%
        mutate(travel_type=as.factor(travel_type))
      #levels(ctrip5_type_plot$travel_type)=
      #  c('alone','family','friend','couple','business')
      ctrip5_type_plot=ctrip5_type_plot%>%
        mutate(travel_type=as.character(travel_type))%>%
        mutate(travel_type=ifelse((travel_type=='friend' | travel_type == 'couple' | travel_type
  == 'alone'),'friend & couple & alone', travel_type))
      boxplot(review_score~travel_type,data=ctrip5_type_plot,
           xlab='consumer type',ylab='rounded review score')


      mu_count_p=ctrip5_type_plot%>%
        group_by(travel_type)%>%
        mutate(type_n=n(),mu_score=mean(round_score))%>%
        ungroup()%>%
        group_by(travel_type,round_score,mu_score,type_n)%>%
        mutate(p=n()/type_n)%>%
        summarize(p=first(p))%>%
        ungroup()
      ctrip5_type_plot2=mu_count_p%>%
        ggplot(aes(x=round_score,y=p,color=travel_type))+
        geom_point()+
        geom_vline(aes(xintercept=mu_score, color=travel_type),
                   linetype="dashed",size=0.5)+
      #  geom_histogram(binwidth =1,
      #                 alpha=0.2,position='identity',fill='white')+
      #  geom_vline(data=mu, aes(xintercept=mu_score, color=travel_type),
      #             linetype="dashed",size=0.5)+
        scale_color_manual(values = c("#CC79A7","#999999", "#D55E00",
                                      "#56B4E9", "#009E73"
                                      #,"#000000","#C3D7A4"
                                      )) +
        labs(x='round_score',y='count_reviews')+
        theme_bw()
      print(ctrip5_type_plot2)
      print(mu_count_p%>%arrange(-mu_score))
      ```
```

## Python code for text analysis for different consumer types

```
#!/usr/bin/env python
coding: utf-8

In[ ]:

import jieba

In[ ]:
```
```

```
11.
12.
13. from google.colab import files # import reiew data file into colab
14. uploaded = files.upload()
15.
16.
17. # In[ ]:
18.
19.
20. import pandas as pd
21. import io
22.
23. df = pd.read_csv(io.StringIO(uploaded['ctrip_3_sample_review.csv'].decode('utf-8')))
24.
25.
26. # Select Major Concern Word
27.
28. # In[ ]:
29.
30.
31. swcc=df[df.travel_type=="独自旅行"] ##select consumer type
32. lswcc=list(swcc.review) ## select review
33.
34.
35. # In[ ]:
36.
37.
38. from google.colab import files   ### change review to txt type
39. with open('swcc.txt', 'w') as f:
40.   for i in lswcc:
41.     f.write(str(i)+"\n")
42. f.close()
43.
44.
45. # In[ ]:
46.
47.
48. import jieba              ##### cut the sentence with jieba and observe the word before stop-word
    cleaning
49.
50. txt = open("swcc.txt", "r", encoding='utf-8').read()
51. words = jieba.lcut(txt)     # use jieba to cut the word
52. counts = {}     # save word and its appearance time with dictionary
53.
54. for word in words:
55.     if len(word) == 1:    # do not count single word
56.         continue
57.     else:
58.         counts[word] = counts.get(word, 0) + 1    # add 1 when the word appear one more time
59.
60. def get_key (dict, value):   ####leave out the word below certain frequency
61.     return [k for k, v in dict.items() if v >= value]
62. def select_volumn(dict,value):
63.     l=get_key(dict,value)
64.     h={}
65.     for i in l:
66.         h[i]=dict[i]
67.     return list(h.items())
68. counts_new=select_volumn(counts,5)
69.
70. items = counts_new
```

```
items.sort(key=lambda x: x[1], reverse=True)    # rank the word by appearance frequency (from high to low)
#for i in range(len(counts_new)):
    #word, count = items[i]
    #print("{0:<5}{1:>5}".format(word, count))


# In[ ]:


from google.colab import files  # upload the baidu_stopword to colab
uploaded = files.upload()


# In[ ]:


import jieba  #### clean stopword and number
import re


# create stopword list
def get_stopwords_list():
    stopwords = [line.strip() for line in open('baidu_stopwords.txt',encoding='UTF-8').readlines()]
    return stopwords


def seg_depart(sentence):

    sentence_depart = jieba.lcut(sentence.strip())
            return sentence_depart

        def remove_digits(input_str):
            punc = u'0123456789.'
            output_str = re.sub(r'[{}]+'.format(punc), '', input_str)
            return output_str

        # clean stopword
        def move_stopwords(sentence_list, stopwords_list):
            # clean stopword
            out_list = []
            for word in sentence_list:
                if word not in stopwords_list:
                    if not remove_digits(word):
                        continue
                    if word != '\t':
                        out_list.append(word)
            return out_list


        # In[ ]:


        ### clean stopword officially
        stopword=get_stopwords_list() ### change txt form of stopword to list form
        swcc_stop1= move_stopwords(words, stopword) ## clean stopword and number


        # In[ ]:


        from google.colab import files   ### output the review after cleaning stopword as txt form
```

```
        with open('swcc_stop1.txt', 'w') as f:
          for i in swcc_stop1:
            f.write(str(i)+"\n")
        f.close()

        swcc_stop1_str=""              ###### output the review after cleaning stopword as str form
        for i in swcc_stop1:
          swcc_stop1_str+=i


        # In[ ]:


        import jieba.analyse   #### find out TF-IDF value

        sentence = swcc_stop1_str
        keywords = jieba.analyse.extract_tags(sentence, topK=20, withWeight=True,
    allowPOS=('n','nr','ns'))

        # print(type(keywords))
        # <class 'list'>

        for item in keywords:
            print(item[0],item[1])

        noun=[]
        importance=[]
        for item in keywords:
          noun.append(item[0])
          importance.append(item[1])

        from pandas.core.frame import DataFrame
        a=noun
        b=importance
        c={"noun" : a,
           "importance index" : b}
        data=DataFrame(c)
        print(data)

        data.to_excel('swcc_importance_index.xls')


        # In[ ]:


        counts = {}              ### count the frequency of word after cleaning the stopword
        for word in swcc_stop1:
            if len(word) == 1:    # do not count single word
                continue
            else:
                counts[word] = counts.get(word, 0) + 1    # add 1 when the word appear one more
    time

        def get_key (dict, value):   ####leave out the word below certain frequency
            return [k for k, v in dict.items() if v >= value]
        def select_volumn(dict,value):
            l=get_key(dict,value)
            h={}
            for i in l:
                h[i]=dict[i]
            return list(h.items())
        counts_new=select_volumn(counts,5)
```

```
192.
193.        items = counts_new
194.        items.sort(key=lambda x: x[1], reverse=True)    # rank the word by appearance frequency
    (from high to low)
195.
196.        #for i in range(len(counts_new)):
197.           #word, count = items[i]
198.           #print("{0:<5}{1:>5}".format(word, count))
199.
200.
201.        # Detail Concern Word Generation
202.
203.        # In[ ]:
204.
205.
206.        from gensim.models import word2vec ### import word2vec package to get detail concerned
    word
207.        sentences=word2vec.Text8Corpus(u'swcc_stop1.txt')
208.        model=word2vec.Word2Vec(sentences,min_count=5,size=109)
209.        model.save('word2vec_model_swcc')
210.
211.
212.        # In[ ]:
213.
214.
215.        important_1=model.most_similar(u"房间",topn=50) ### check the detail concern word
216.        important_1
217.
218.
219.        # In[ ]:
220.
221.
222.        related_word=[]
223.        similarity=[]
224.        for i in important_1:
225.          related_word.append(i[0])
226.          similarity.append(i[1])
227.
228.        from pandas.core.frame import DataFrame
229.        a=related_word
230.        b=similarity
231.        c={"related_word" : a,
232.           "similarity": b}
233.        data=DataFrame(c)
234.        print(data)
235.
236.        data.to_excel('swcc_similarity_房间.xls')
237.
238.
239.        # In[ ]:
240.
241.
242.        relate_list=[]                          ####output the detail concern word in list form
243.        for i in range(50):
244.          relate_list.append(important_1[i][0])
245.        #relate_list
246.
247.
248.        # Sentiment Analysis
249.
250.        # In[ ]:
251.
```

```
252.
253.         get_ipython().system('pip install snownlp          ##import snownlp ')
254.         from snownlp import SnowNLP
255.
256.
257.         # In[ ]:
258.
259.
260.         ###generate the sentiment value by generate detail concern word's sentiment
261.         important_1_key_weight={}
262.         for i in important_1:
263.           important_1_key_weight[i[0]]=i[1]
264.
265.         keystr=""
266.         for i in important_1:
267.           keystr+=i[0]+"。"
268.
269.         print(keystr)
270.
271.         from snownlp import SnowNLP
272.         s = SnowNLP(keystr)
273.         important_1_key_value={}
274.         n=-1
275.         for sentence in s.sentences:
276.           n+=1
277.           important_1_key_value[sentence]=SnowNLP(s.sentences[n]).sentiments
278.
279.         important_1_key_value
280.
281.         total_score=0
282.         for sentence in s.sentences:      ####calculate the sentiment value of detail word
283.           value=important_1_key_value[sentence]*important_1_key_weight[sentence]
284.           total_score+=value
285.
286.         average_score=total_score/50
287.
288.         average_score
```

Research Question 3

R code for time series analysis on review scores

```
1.  **1. Load the data**
2.  ```{r}
3.  library(readxl)
4.  library(dplyr)
5.  review = read_excel("/Users/yifanshen/Documents/SMEUG/Grade_3_Term_2/DMS
    2051/Project/review data/Ctrip_Rate.xlsx")
6.  ```
7.
8.
9.
10. **2. Data cleaning**
11. ```{r}
12. names(review) = c("Date", "HotelName", "Rate", "TravelType")
13. review = data.frame(review)
14. review[,1] = as.Date(review[,1])
15. review[,3] = as.numeric(review[,3])
16. review = na.omit(review)
17. Modern = review[which(review[2] == "Modern_Classic"),]
18. Modern = Modern[order(Modern$Date, decreasing = FALSE),]
19. Modern = Modern[,c(1,3)]
```

```
ModernAVG = Modern %>% group_by(Date) %>% mutate(avgRate = mean(Rate)) %>% ungroup()
ModernAVG = unique(ModernAVG[,-2])
```

Here, the average of ratings is taken in a single day. Thus, a series of rating has been formed corresponding to each date.

3. Transform the rating into a stationary series

```{r}
plot(ModernAVG, ylab='Average Rating')
```

Obviously, the average rating series is not stationary. The *seasonality* does **not** exist in the series.
To get a stationary series, let's take the first-difference of it.

```{r}
deltaRate = na.omit(ModernAVG$avgRate - lag(ModernAVG$avgRate))
plot(deltaRate, ylab='Change in Rating')
```

As the graph shows, the changes of rating between days is probably a stationary series.

4. Test for auto-correlation

```{r}
Box.test(deltaRate, lag = 10, type = c("Box-Pierce", "Ljung-Box"), fitdf = 0)
```

For p-value < 0.05, the series is supposed to be auto-correlated.

5. Modeling

```{r}
acf(deltaRate, main = "ACF")
pacf(deltaRate, main = "PACF")
```

Based on ACF plot, the data seem fit the MA(1) Model.

```{r}
arima = arima(deltaRate, order = c(0, 0, 1))
arima
```

MA(1) Model has been constructed.

6. Diagnose

```{r}
tsdiag(arima)
Box.test(arima$residuals, lag = 10, type = c("Box-Pierce", "Ljung-Box"), fitdf = 0)
```

```
80.
81. According to the diagnosis, the series is successfully explained by MA(1) Model.
82. The p-value of model residuals = 0.8886, which means the residuals are not auto-correlated and the
    data are fully explained by the model.
83.
84. The model is:
85. $$\delta(R_{t}) = {\varepsilon_{t}} - 0.9793 {\varepsilon_{t-1}} , {\varepsilon_{t}}-i.i.d. N(0,
    \sigma^2)$$
86.
87.
88.
89. **7. Generalization**
90.
91. The ratings of *Modern Classic Hotel Shenzhen* shows an auto-correlation fitting the MA(1) Model. To
    figure out if there are generalized auto-correlation in hotel ratings, the hotels are divided into three
    groups by their star levels. The *Auto-correlation Rate* is calculated to find the probability of
    presenting the auto-correlated ratings of a hotel, which is composed by formular:$AC = 1- P(p-
    value > 0.05)$.
92.
93.
94. First, some functions can be defined.
95.
96. ```{r}
97. #P-value for a single hotel
98. pvalue = function(ctrip, hotelname){
99.   Rating = ctrip[which(ctrip[,2] == hotelname),]
100.          if (nrow(Rating) > 1){
101.            Rating = Rating[order(Rating$Date, decreasing = FALSE),]
102.            Rating = Rating[,c(1,3)]
103.            RatingAVG = Rating %>% group_by(Date) %>% mutate(avgRate = mean(Rate)) %>%
    ungroup()
104.            RatingAVG = unique(RatingAVG[,-2])
105.            deltaRate = na.omit(RatingAVG$avgRate - lag(RatingAVG$avgRate))
106.            box = Box.test(deltaRate, lag = 1, type = c("Box-Pierce", "Ljung-Box"), fitdf = 0)
107.            return(box$p.value)
108.          }
109.        }
110.
111.        # P-value for all
112.        pvalueT = function(ctrip, hotelname){
113.          pv = rep(0,length(hotelname))
114.          for (x in seq(length(hotelname))){
115.            name = hotelname[x]
116.            pv[x] = pvalue(ctrip, name)
117.          }
118.          return(na.omit(pv))
119.        }
120.        ```
121.
122.
123.
124.        **(i.) 3 Star**
125.        ```{r}
126.        ctrip_3 = read.csv("/Users/yifanshen/Documents/SMEUG/Grade_3_Term_2/DMS
    2051/Project/review data/ctrip_3_sample_review.csv")
127.        # Data Cleaning
128.        ctrip_3 = ctrip_3[,-1]
129.        names(ctrip_3) = c("Date", "HotelName", "Rate", "TravelType")
130.        ctrip_3 = data.frame(ctrip_3)
131.        ctrip_3[,1] = as.Date(ctrip_3[,1])
132.        ctrip_3[,2] = as.character(ctrip_3[,2])
133.        ctrip_3[,3] = as.numeric(ctrip_3[,3])
```

```
        ctrip_3 = na.omit(ctrip_3)
        hotels = group_by(ctrip_3, HotelName)
        hotelname = summarise(hotels)
        hotelname = hotelname[[1]]
        hotelname = as.character(hotelname)
        for (x in seq(length(hotelname))){
          name = hotelname[x]
          Rating = ctrip_3[which(ctrip_3[,2] == name),]
          if (nrow(Rating) < 2){
            ctrip_31 = ctrip_3[-which(ctrip_3[,2] == name),]
            hotelname1 = hotelname[-x]
          }
        }
        ctrip_3 = ctrip_31
        hotelname = hotelname1
        (Sample_Number = length(hotelname))

        # Calculation
        pv = pvalueT(ctrip_3, hotelname)
        (AC_Rate3 =  1- sum(pv > 0.05) / length(pv))

        ```

        So, the *Auto-correlation Rate* of 3 Star Hotels is: **94.16667%**

        (ii.) 4 Star
        ```{r}
        ctrip_4 = read.csv("/Users/yifanshen/Documents/SMEUG/Grade_3_Term_2/DMS
    2051/Project/review data/ctrip_4_sample_review.csv")
        # Data Cleaning
        names(ctrip_4) = c("Date", "HotelName", "Rate", "TravelType")
        ctrip_4 = data.frame(ctrip_4)
        ctrip_4[,1] = as.Date(ctrip_4[,1])
        ctrip_4[,2] = as.character(ctrip_4[,2])
        ctrip_4[,3] = as.numeric(ctrip_4[,3])
        ctrip_4 = na.omit(ctrip_4)
        hotels = group_by(ctrip_4, HotelName)
        hotelname = summarise(hotels)
        hotelname = hotelname[[1]]
        hotelname = as.character(hotelname)
        for (x in seq(length(hotelname))){
          name = hotelname[x]
          Rating = ctrip_4[which(ctrip_4[,2] == name),]
          if (nrow(Rating) < 2){
            ctrip_41 = ctrip_4[-which(ctrip_4[,2] == name),]
            hotelname1 = hotelname[-x]
            ctrip_4 = ctrip_41
            hotelname = hotelname1
          }
        }
        (Sample_Number = length(hotelname))

        # Calculation
        pv = pvalueT(ctrip_4, hotelname)
        (AC_Rate4 =  1- sum(pv > 0.05) / length(pv))
        ```

        So, the *Auto-correlation Rate* of 4 Star Hotels is: **98.33333%**


```
```

```
        **(iii.) 5 Star**
        ```{r}
        ctrip_5 = read.csv("/Users/yifanshen/Documents/SMEUG/Grade_3_Term_2/DMS 2051/Project/review data/ctrip_5_sample_review.csv")
        # Data Cleaning
        names(ctrip_5) = c("Date", "HotelName", "Rate", "TravelType")
        ctrip_5 = data.frame(ctrip_5)
        ctrip_5[,1] = as.Date(ctrip_5[,1])
        ctrip_5[,2] = as.character(ctrip_5[,2])
        ctrip_5[,3] = as.numeric(ctrip_5[,3])
        ctrip_5 = na.omit(ctrip_5)
        hotels = group_by(ctrip_5, HotelName)
        hotelname = summarise(hotels)
        hotelname = hotelname[[1]]
        hotelname = as.character(hotelname)
        for (x in seq(length(hotelname))){
          name = hotelname[x]
          Rating = ctrip_5[which(ctrip_5[,2] == name),]
          if (nrow(Rating) < 2){
            ctrip_51 = ctrip_5[-which(ctrip_5[,2] == name),]
            hotelname1 = hotelname[-x]
            ctrip_5 = ctrip_51
            hotelname = hotelname1
          }
        }
        (Sample_Number = length(hotelname))

        # Calculation
        pv = pvalueT(ctrip_5, hotelname)
        (AC_Rate5 =  1- sum(pv > 0.05) / length(pv))
        ```

        So, the *Auto-correlation Rate* of 5 Star Hotels is: **100%**


        ```{r}
        AC_Rate = c(AC_Rate3, AC_Rate4, AC_Rate5)
        Starlvl = c("3 Star(# = 121)", "4 Star(# = 61)", "5 Star(# = 22)")
        barplot(AC_Rate,names.arg=Starlvl,xlab="Star Level",ylab="Auto-correlation Rate",
              main="Auto-correlation Rates of Different Star Levels")
        ```
```