Leeds University Business School

UNIVERSITY OF LEEDS

Assessed Coursework Coversheet

For use with *individual* assessed work

Student ID Number:	2	0	1	7	9	9	3	4	8
Module Code:	**LUBS5309M**								
Module Title:	**Forecasting and Advanced Business Analytics**								
Module Leader:	Panagiotis Stamolampros								
Declared Word Count:	2911								

Please Note:

Your declared word count must be accurate, and should not mislead. Making a fraudulent statement concerning the work submitted for assessment could be considered academic malpractice and investigated as such. If the amount of work submitted is higher than that specified by the word limit or that declared on your word count, this may be reflected in the mark awarded and noted through individual feedback given to you.

It is not acceptable to present matters of substance, which should be included in the main body of the text, in the appendices ("appendix abuse"). It is not acceptable to attempt to hide words in graphs and diagrams; only text which is strictly necessary should be included in graphs and diagrams.

By submitting an assignment you confirm you have read and understood the University of Leeds **Declaration of Academic Integrity** (http://www.leeds.ac.uk/secretariat/documents/academic_integrity.pdf).

Part 1: Forecasting on US Seasonally seasonally-adjusted personal consumption expenditure data

Introduction

This report focuses on forecasting US personal consumption expenditures (PCE) using historical data from the PCE.csv file. Three forecasting models are used: a simple method, an Exponential Smoothing model, and an ARIMA model. The aim is to compare their predictive abilities and identify the most accurate. The report details the data analysis process, model selection criteria, prediction results, and an estimate for October 2024's PCE. It also compares the models using one-step ahead rolling forecasting without re-estimation. The report's goal is to provide a comprehensive understanding of the selected models' forecasting abilities and identify the best model for future PCE predictions.

Data Preparation and Initial Steps

A crucial step in the process of every data analysis project is data preparation. Refinement and translation of raw data into a format that is structured and allows for more efficient analysis are its main goals. We used a multi-step technique to prepare the data for the forecasting algorithms. The dataset that is currently being examined which is "PCE.csv" consists of 779 rows and contains Personal consumption expenditure data from 1959 to 2023.

Handling Missing Values

Missing data can introduce bias or lead to loss of information in our analysis. We used interpolation to fill in missing values in our dataset. Interpolation is a method of estimating values between two known values. It's particularly useful when your data is ordered and you can reasonably assume trends or patterns between your data points. The interpolated new dataset is called "data_fnl"

In our case, we used linear interpolation, which assumes a straight line between two points and estimates the missing values along this line. This method is simple and computationally efficient, making it a good choice for our large dataset.

```
4  # Load data####
5  data <- read.csv("PCE.csv")
6  #Replacing missing values####
7  data_fnl <- na_interpolation(data)
8  view(data_fnl)
9
```

	DATE	PCE
1	01/01/1959	306.1
2	01/02/1959	309.6
3	01/03/1959	312.7
4	01/04/1959	312.2
5	01/05/1959	316.1
6	01/06/1959	318.2
7	01/07/1959	317.8
8	01/08/1959	320.2
9	01/09/1959	324.2
10	01/10/1959	322.8
11	01/11/1959	NA
12	01/12/1959	322.9
13	01/01/1960	323.6
14	01/02/1960	NA

	DATE	PCE
1	01/01/1959	306.10
2	01/02/1959	309.60
3	01/03/1959	312.70
4	01/04/1959	312.20
5	01/05/1959	316.10
6	01/06/1959	318.20
7	01/07/1959	317.80
8	01/08/1959	320.20
9	01/09/1959	324.20
10	01/10/1959	322.80
11	01/11/1959	322.85
12	01/12/1959	322.90
13	01/01/1960	323.60
14	01/02/1960	326.90

Figure 1: *Before (left) and after(right) interpolation*

Data Splitting

To evaluate the performance of our models, we split our data into a training set and a test set. The training set is used to train our models, while the test set evaluates their performance on unseen data.

We used an 80-20 split, meaning 80% of our data (623 rows out of 779) was used for training and the remaining 20% for testing. This split ensures that we have enough data to train our models, while still leaving a substantial amount for testing.

```
  ylab( Personal Consumption Expenditures(PCE)  )
# Split the data into training and testing sets
train_size <- floor(0.8 * nrow(data))
train <- data_fnl[1:train_size, ]
test <- data_fnl[(train_size + 1):nrow(data), ]
nrow(train)
nrow(test)
```

Creation of necessary time series

The preparatory phase preceding model creation involves the generation of three distinct time series datasets:

1. **dataTS**: This primary time series comprises data spanning from January 1959 to November 2023. It serves as the foundational dataset for subsequent analyses and model development.

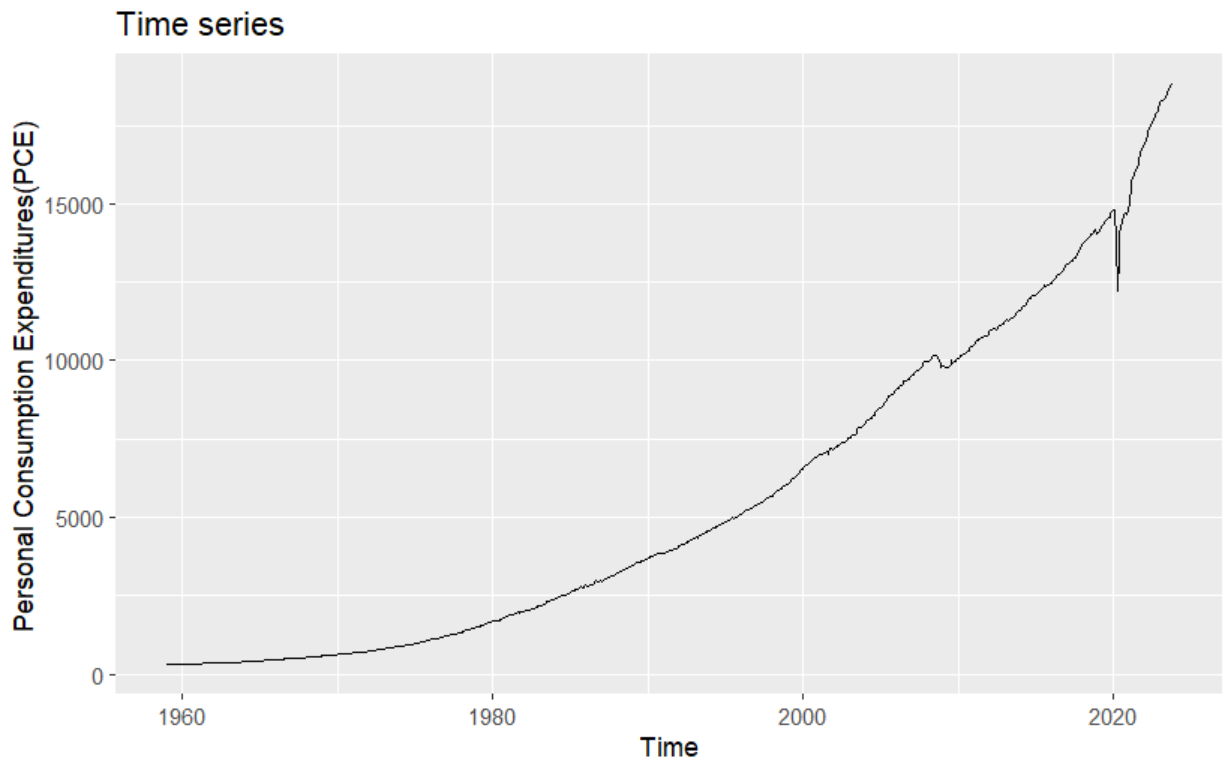
2. **trainTS**: Derived from the original dataset "data_fnl," the train time series is meticulously constructed by partitioning 80% of the dataset. This segment constitutes the training dataset, meticulously curated to facilitate model training and parameter estimation.

3. **testTS**: Comprising 20% of the "data_fnl" dataset, the test time series is meticulously isolated to form the test dataset. This dataset segment serves as a critical component in evaluating the performance and generalizability of the developed model, ensuring robustness and reliability in real-world applications.

Model Creation

One simple forecasting technique (Drift, Average, Naive, Snaive), Exponential Smoothing model and ARIMA models been used in this study to predict seasonally-adjusted personal consumption expenditures (PCE) in the US. These models were chosen based on their ability to handle a variety of data trends and patterns. The time series study started by initialising a monthly-frequency dataset that dates back to January 1959. The resulting time series plot on consumer expenditure from 1959 to 2023 has been created. All forecasting models are created on “trainTS” time series with forecast horizon h equal to length of test series “testTS".

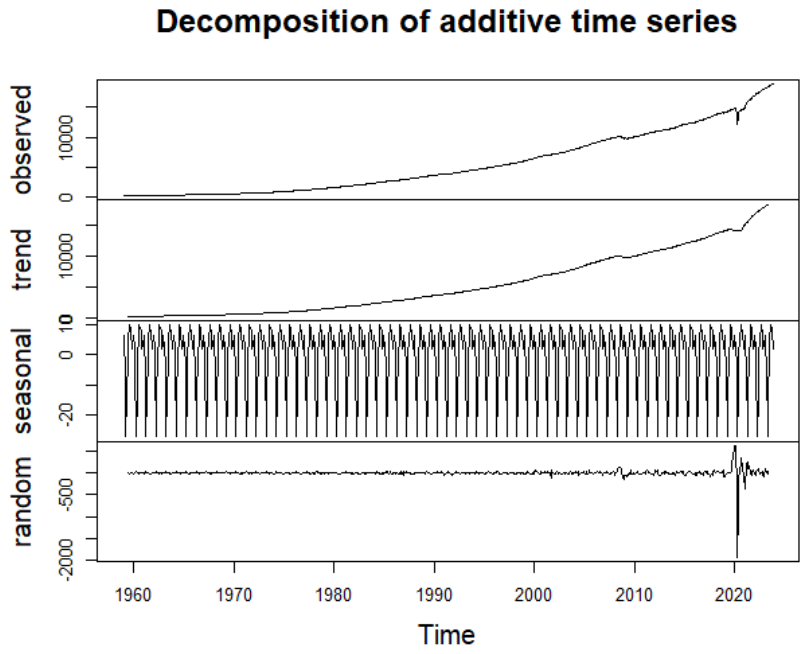
```
# Load data####
data <- read.csv("PCE.csv")
#changing the format of date column into date format####
#Replacing missing values####
data_fnl <- na_interpolation(data)
nrow(data_fnl)

#Creating and plotting the complete time series####
dataTS <- ts(data_fnl$PCE,start=c(1959, 1), frequency=12)
```



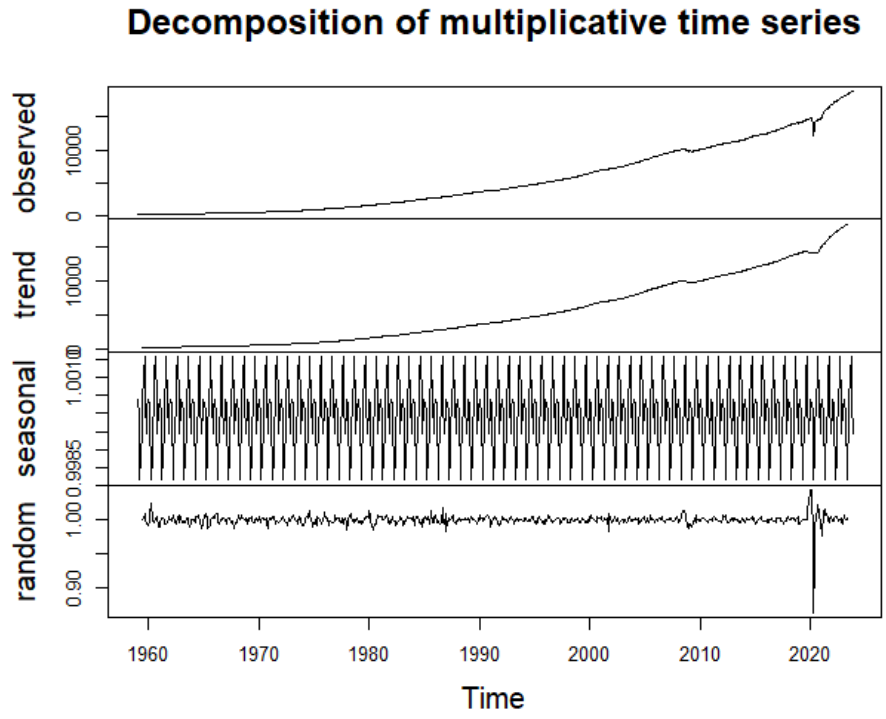
Primary time series plot of PCE

Following that, the 'decompose' function is used to perform time series decomposition, which separates the time series into its component trend, seasonal, and random components. The 'additive' and 'multiplicative' types are used to carry out this procedure twice, both additive and multiplicative decomposition verify the presence of a **trend** in the time series and the

results are shown graphically.



Plot of additive decomposition



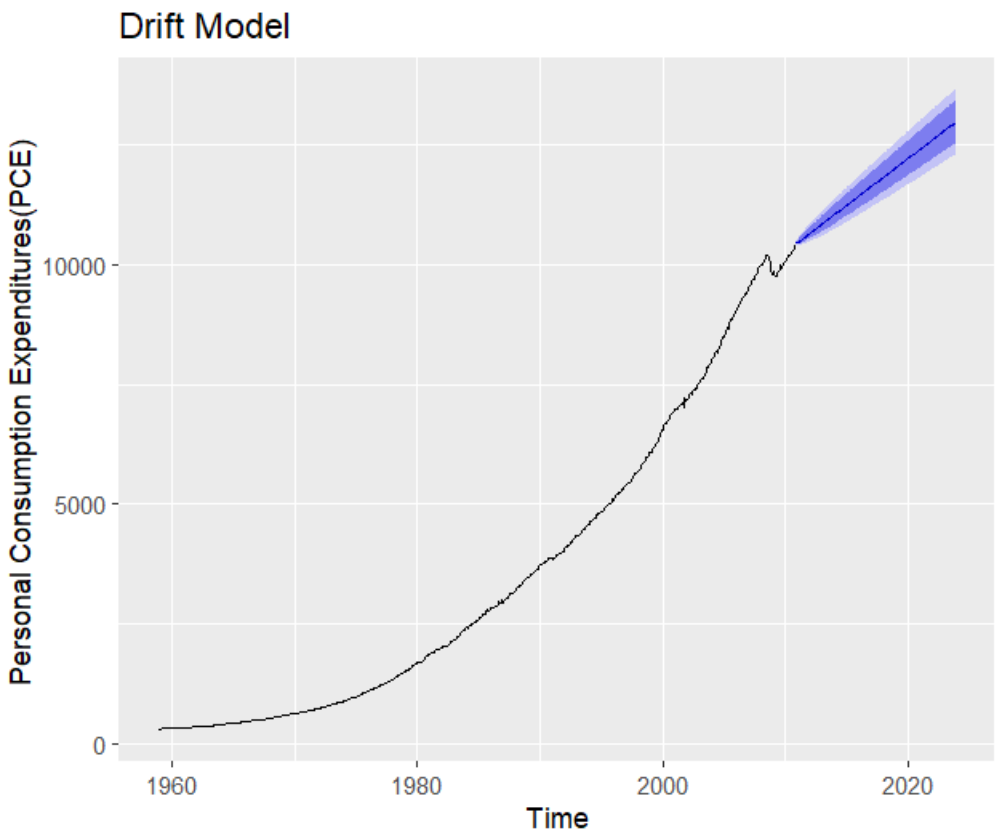
Plot of multiplicative decomposition

Simple forecasting method

The use of a drift model makes sense in light of the data's observed trend and seasonal adjustments. By projecting this line forward, the drift model makes it easier to build a linear trajectory that runs from the first to the last recorded data. It is especially useful for datasets that exhibit a clear trend but lack seasonal variations.

A drift model is fitted into the “trainTS” with h equal to length of “testTS”. The rwf function in R studio is used for this purpose.

```
# Fit the model using drift method
drift_model <- rwf(trainTS,h=length(testTS),drift = TRUE)
autoplot(drift_model) +
ggtitle("Drift Model") +  # Add title
  xlab("Time") +     # Add x-axis label
  ylab("Personal Consumption Expenditures(PCE) ")
```

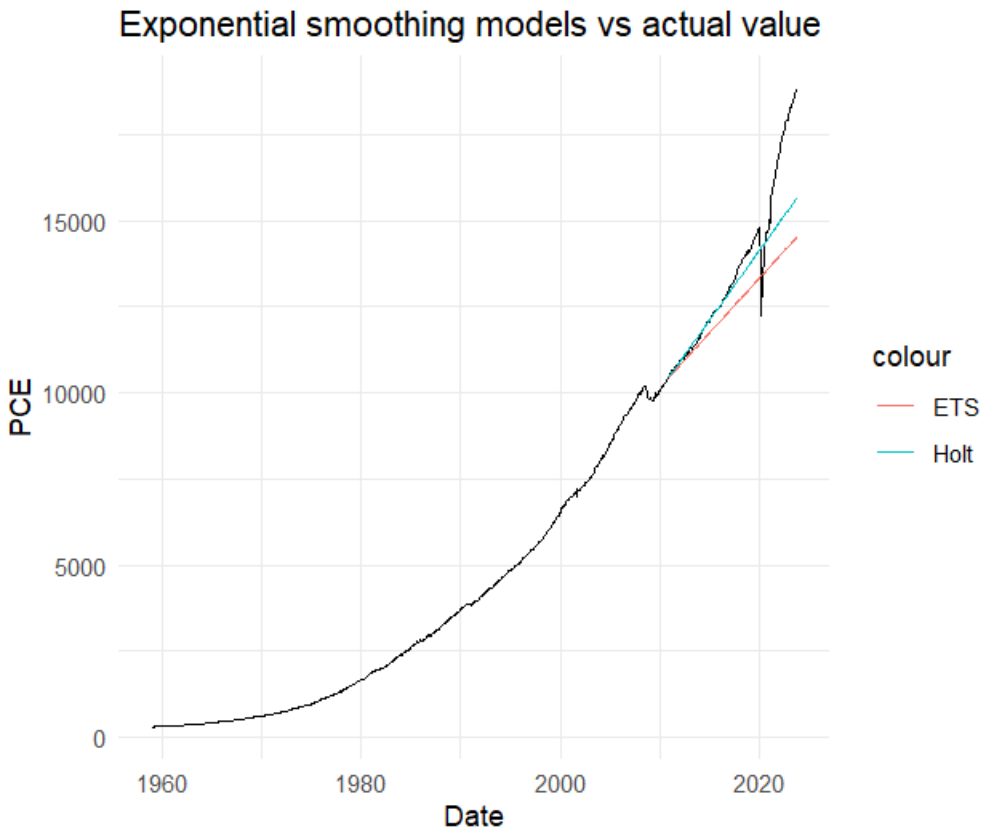


Plot of drift model

Exponential Smoothing Model

Exponential smoothing forecasts are based on the weighted average of past observations, with more recent observations weighted more heavily. The method assumes that more recent data points provide better information for predicting future values. For analysis, we consider Holt's linear smoothing method, optimal for trend-centric datasets, and the Exponential Smoothing State Space Model, which offers a versatile representation surpassing traditional models. It provides a robust theoretical foundation for model estimation and prediction. R programming features auto-exponential smoothing, autonomously selecting the best model and displaying parameters for streamlined analysis. Holt model is fitted on “trainTS” with h equals to “testTS” using “holt” function in R and state space model is made using “ets” function in R.

```
# Exponential Smoothing (auto.ses)####

PCE_ets <- ets(trainTS)
PCE_ets
forecast_ets <- forecast(PCE_ets, h = length(testTS))
autoplot(forecast_ets)

#Fcholt####

fcholt <- holt(trainTS, h= length(testTS))
summary(fcholt)
autoplot(fcholt)
```



When ETS and Holt's method are plotted together with actual values

Auto ETS function in R suggests ETS (M,A,N) model, but graphical analysis reveals Holt's method as superior due to better alignment with actual values. ETS (M,A,N) implies multiplicative error, additive trend, no seasonality. Holt's, with additive trend, adapts better to changing trends, evident in visual comparison.

ARIMA Model

ARIMA, an acronym for AutoRegressive Integrated Moving Average, is a robust modeling approach for time series analysis. It leverages historical data to forecast future values by considering its own lags and previous forecast errors. The model's three pivotal parameters, (p, d, q), delineate its architecture:

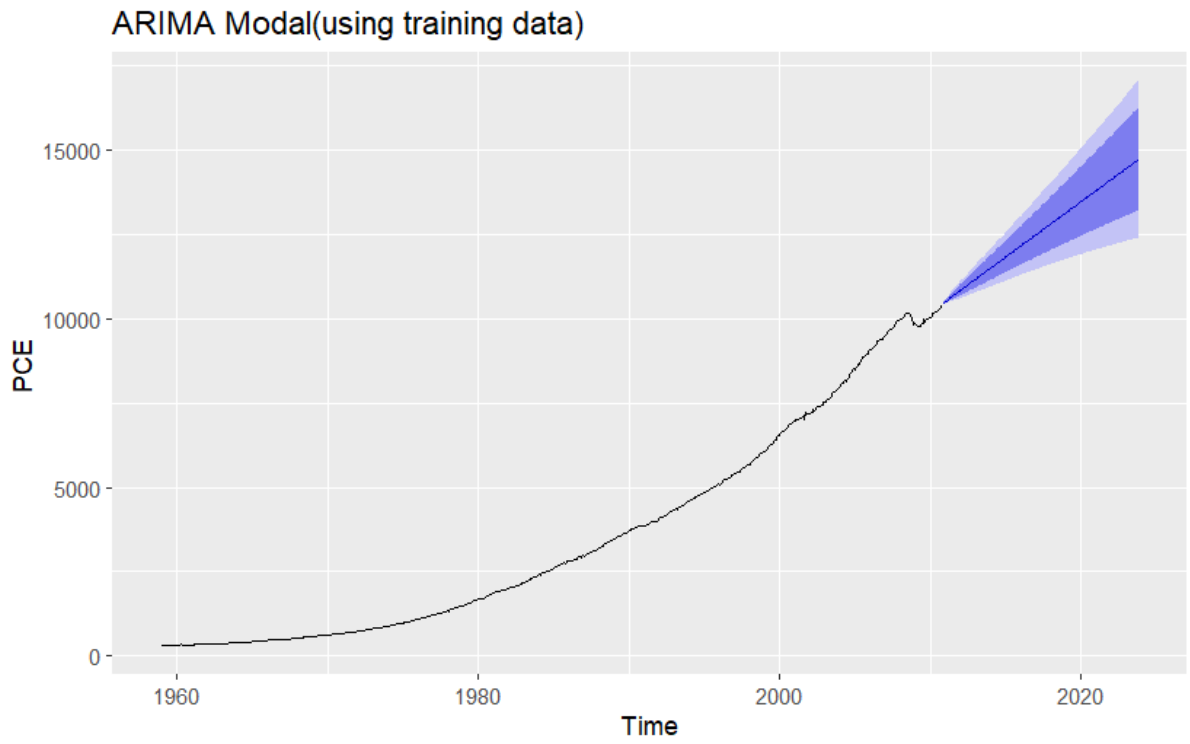
- p: Order of the Autoregressive (AR) component.
- d: Order of Integration (I) for achieving stationarity.
- q: Order of the Moving Average (MA) component.

The model is created using the “auto.arima” function in R. This function automates the process of ARIMA model selection by searching over various combinations on all three parameters p, q and d.

```
126
127 # ARIMA (auto.arima)####
128 arima_model <- auto.arima(trainTS)
129 arima_model
130 arima_forecast <- forecast(arima_model, h = length(testTS))
131 arima_forecast_values <- arima_forecast$mean
132 autoplot(arima_forecast)
133 autoplot(arima_forecast)+
134   ggtitle("ARIMA Modal(using training data)") +
135   xlab("Time") +
136   ylab("PCE ")
```

Figure 5: *ARIMA in R*

ARIMA model is designed using auto.arima and the model is fitted in “trainTS” and forecasted in “testTS”. The ARIMA model given by “auto. arima” is ARIMA (3,2,2) which translates to three past observations (AR=3), two differences (I=2) for stationarity, and two lagged forecast errors (MA=2) for predictions.



Plot of Arima forecast using auto ARIMA

Selection of the best model

An evaluation function is created to evaluate the metrics required for deciding the best model.

```
# Function to calculate evaluation metrics
evaluate <- function(trainTS, testTS) {
  mae <- mean(abs(trainTS - testTS))
  mse <- mean((trainTS - testTS)^2)
  rmse <- sqrt(mse)
  return(list(MAE = mae, MSE = mse, RMSE = rmse))
}
```

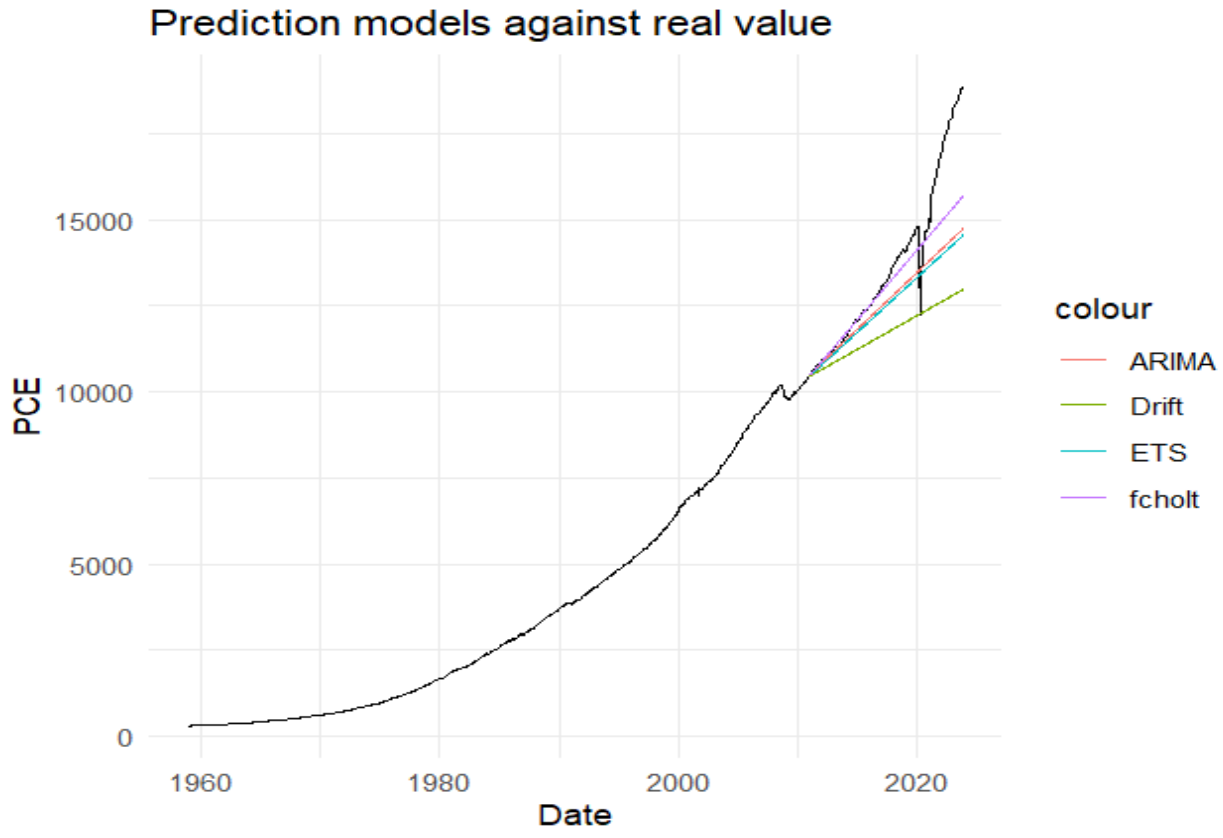
```
# Evaluate ARIMA
arima_eval <- evaluate(testTS, arima_forecast_values)
# Compare models
comparison <- data.frame(
  Model = c( "ETS", "ARIMA","Drift", "fcholt"),
  MAE = c( ETS_eval$MAE, arima_eval$MAE, drift_eval$MAE, fcholt_eval$MAE),
  MSE = c( ETS_eval$MSE, arima_eval$MSE, drift_eval$MSE, fcholt_eval$MSE),
  RMSE = c( ETS_eval$RMSE, arima_eval$RMSE, drift_eval$RMSE, fcholt_eval$RMSE)
)
print(comparison)
view(comparison)
write.csv(comparison,file = "cmpp1")
```

The Mean Absolute Error (MAE), Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) of ETS, ARIMA, Drift and Holt's models are measured and the model with the minimum value of all three parameters has been selected as the best model.

Model	MAE	MSE	RMSE
ETS	1188.248332	2960997.253	1720.75485
ARIMA	1076.198138	2611226.888	1615.92911
Drift	1954.94101	6583617.138	2565.856024
fcholt	665.8912391	1360380.18	1166.353368

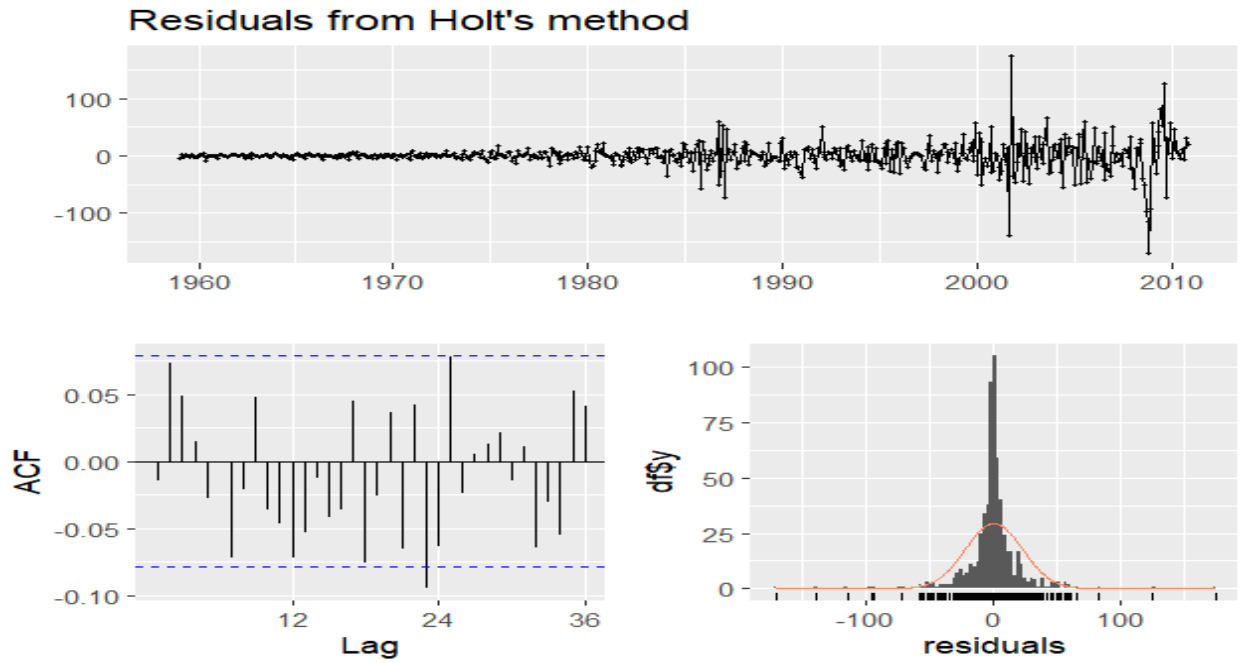
Table with the comparison of three models

It is evident from the table that the model with the least value of all three parameters is the **Holts' model**.



plotted predicted models against the actual value

Limitations of the model



Plot 7: *Residuals from Holts' method*

The assessment of residuals of Holt's method reveals notable limitations within the model. Inspection of residual plots reveals conspicuous autocorrelations at specific lags, indicating a departure from independently distributed residuals. Interestingly, despite its comparative advantage, the Holt model exhibits residual autocorrelation, to a lesser extent than other analysed models.

The statistical evaluation employing the Ljung-Box test underscores this observation, yielding a p-value of 0.0295. This result suggests a significant departure from the assumption of independent residuals, providing empirical evidence for the presence of autocorrelation within the residuals.

Predictions for October 2024

The prediction for October 2024 has been obtained through the utilization of Holt's method, a forecasting technique chosen for its proven effectiveness in prior analyses and esteemed status within the realm of time series analysis. This method's application is contingent upon the foundational input furnished by the initial time series dataset, denoted as "dataTS". The parameter 'h' has been set at a value of 11 to project into October 2024, aligning with the

forecast horizon and corresponding with the conclusion of the actual time series data in November 2023. The code and results are given below.

```
> oct_2024_fcholt
         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
Dec 2023       18923.27 18805.23 19041.31 18742.74 19103.80
Jan 2024       18987.63 18819.53 19155.74 18730.54 19244.73
Feb 2024       19052.00 18844.67 19259.33 18734.92 19369.08
Mar 2024       19116.36 18875.29 19357.44 18747.67 19485.06
Apr 2024       19180.73 18909.32 19452.13 18765.65 19595.81
May 2024       19245.09 18945.73 19544.46 18787.25 19702.94
Jun 2024       19309.46 18983.88 19635.04 18811.53 19807.39
Jul 2024       19373.83 19023.38 19724.27 18837.86 19909.79
Aug 2024       19438.19 19063.95 19812.43 18865.84 20010.54
Sep 2024       19502.56 19105.39 19899.72 18895.15 20109.96
Oct 2024       19566.92 19147.56 19986.28 18925.56 20208.28
> 
```

Month	Point.Forecast
December	18923.27
January	18987.63
February	19052.00
March	19116.36
April	19180.73
May	19245.09
June	19309.46
July	19373.83
August	19438.19
September	19502.56
October	19566.92

Forecasts until October 2024

The forecast for October 2024 is 19566.92 which is a 3.85% increase from October 2023

One-step ahead rolling forecasting

One-step ahead rolling forecasting is a method used in time series analysis where a model is used to predict the value of the next time step based on the information available up to the current time step. This process is then repeated by moving the time window one step forward and making a new prediction, hence the term “rolling”.

For the creation of the model the following steps are used:

1. Define Rolling Forecast Function: A function `rolling_forecast` is defined to perform one-step ahead forecasting.

```
# Define the function for one-step ahead rolling forecasting
rolling_forecast <- function(model, trainTS, testTS) {
  forecasts <- numeric(length(testTS))
  for (i in seq_along(testTS)) {
    model <- forecast(model, h = 1)
    forecasts[i] <- model$mean[1]
    model$x <- c(model$x, testTS[i])
  }
  return(forecasts)
}
```

2. Fit Models: ARIMA, ETS, Drift, and Holt's models are fitted on the training data.

3. Perform Rolling Forecasting: The `rolling_forecast` function is applied to each model.

4. Evaluate Forecasts: Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) are calculated for each model's forecasts.

5. Compare Models: The evaluation metrics for each model are stored in a data frame for comparison.

And finally, the comparison is saved in CSV format under the name “comparison_rolling_onestep.csv”. following are the R windows while conducting the analysis

```r
# Fit the models
arima_model_onestep <- auto.arima(trainTS)
ets_model_onestep <- ets(trainTS)
drift_model_onestep <- rwf(trainTS, drift = TRUE)
fcholt_model_onestep <- holt(trainTS)

# Perform rolling forecasting for each model
arima_forecasts_onestep <- rolling_forecast(arima_model_onestep, trainTS, testTS)
ets_forecasts_onestep <- rolling_forecast(ets_model_onestep, trainTS, testTS)
drift_forecasts_onestep <- rolling_forecast(drift_model_onestep, trainTS, testTS)
fcholt_forecasts_onestep <- rolling_forecast(fcholt_model_onestep, trainTS, testTS)

# Create a data frame to store the results
comparison_rolling_onestep <- data.frame(
  Model = c("ARIMA", "ETS", "DRIFT", "fcholt"),
  MAE_onestep = c(mean(abs(testTS - arima_forecasts_onestep)), mean(abs(testTS - ets_forecasts_onestep)),
                  mean(abs(testTS - drift_forecasts_onestep)), mean(abs(testTS - fcholt_forecasts_onestep))),
  MSE_onestep = c(mean((testTS - arima_forecasts_onestep)^2), mean((testTS - ets_forecasts_onestep)^2),
                  mean((testTS - drift_forecasts_onestep)^2), mean((testTS - fcholt_forecasts_onestep)^2)),
  RMSE_onestep = c(sqrt(mean((testTS - arima_forecasts_onestep)^2)), sqrt(mean((testTS - ets_forecasts_onestep)^2)
```

```r
# Print and save the comparison
print(comparison_rolling_onestep)
write.csv(comparison_rolling_onestep, file = "comparison_rolling_onestep.csv")

# Load necessary library
```

Model	MAE_onestep	MSE_onestep	RMSE_onestep
ARIMA	3173.688827	15615375.8	3951.629512
ETS	3183.465138	15677525.32	3959.485486
DRIFT	3187.863358	15705547.82	3963.022562
fcholt	3173.542566	15614447.45	3951.512046

Values of evaluation metrics corresponding to their model

Across the four models under examination, the empirical study clearly shows that **Holt's** model performs better than the others when one-step-ahead forecasting is applied.

Part 2: Topic Modelling on online hotel review

Introduction

This study aims to analyze customer reviews from the "HotelData.csv" file to identify factors influencing satisfaction and dissatisfaction in the hospitality industry. Using advanced natural language processing techniques, the project will categorize reviews into positive and negative sentiments based on Likert-scale ratings. A random sample of 2,000 reviews will be analyzed through a process that includes data preprocessing, model selection, and topic extraction and labeling. The goal is to provide a detailed understanding of the main factors contributing to customer satisfaction and dissatisfaction, thereby assisting in the development of targeted strategies for service improvement.

Methodology

Data review

The given csv file "HotelData.csv" contains online review for hotels and their corresponding ratings. The data has 10,000 rows of ratings from 1 to 5 and reviews. The dataset has data from languages other than English, so it would not be good for analysis at the beginning to solve that all the reviews from languages other than English are removed and the random sample was taken from the remaining 7689 rows of English review.

```r
set.seed(348)

# Load data
reviews <- read.csv("HotelsData.csv")

# Detect the language of each text
reviews <- reviews %>%
  mutate(language = textcat(Text.1))

# Filter out texts with non-English languages
reviews_english <- reviews %>%
  filter(language == "english")

selected_data <- subset(reviews_english, select = c(1,2))

# Selecting 2000
dataTM <- sample_n(selected_data, 2000)
```

Classification of reviews

The selected 2000 reviews are classified based on their scores, all reviews with scores of 4 and above are considered positive, have 1529 rows and 2 and below are negative has 214 rows and remaining are considered neutral.

```r
# Converting to positive and negative review
positive_reviews <- dataTM %>% filter(Review.score >= 3)
negative_reviews <- dataTM %>% filter(Review.score <= 2)
nrow(dataTM)
```

Data preprocessing

After separating the data into positive and negative based on their score, obtained two datasets with the name “positive_reviews” and “negative_reviews” . The intial preprocessing method before the creation of the model is the creation of the corpus.

Later the separated corpus under goes text preprocessing The tm_map() function is used to preprocess the text in the corpus. This includes converting all text to lower case, removing numbers, removing stop words in English, removing punctuation, and removing white spaces.

```
# Creation of corpus
positive_corpus <- Corpus(VectorSource(positive_reviews$Text.1))
negative_corpus <- Corpus(VectorSource(negative_reviews$Text.1))

# Preprocessing
positive_corpus <- tm_map(positive_corpus, content_transformer(tolower)) # Convert all text to lower case
positive_corpus <- tm_map(positive_corpus, removeNumbers) # Remove numbers from document
positive_corpus <- tm_map(positive_corpus, removeWords, stopwords("english")) # Remove stopwords in English
positive_corpus <- tm_map(positive_corpus, removePunctuation, preserve_intra_word_dashes = TRUE)
positive_corpus <- tm_map(positive_corpus, stripWhitespace) # Remove white spaces

negative_corpus <- tm_map(negative_corpus, content_transformer(tolower)) # Convert all text to lower case
negative_corpus <- tm_map(negative_corpus, removeNumbers) # Remove numbers from document
negative_corpus <- tm_map(negative_corpus, removeWords, stopwords("english")) # Remove stopwords in English
negative_corpus <- tm_map(negative_corpus, removePunctuation, preserve_intra_word_dashes = TRUE)
negative_corpus <- tm_map(negative_corpus, stripWhitespace) # Remove white spaces
```

Finaly both the reviews undergo Tokenization which is used to tokenize the preprocessed text into individual words and Lemmatization which is used to lemmatize the tokens.

```
# Tokenize the corpus
positive_Token <- tokenize_words(positive_corpus$content)
Negative_Token <- tokenize_words(negative_corpus$content)

# Lemmatize tokens
positive_lemmas <- lemmatize_words(positive_Token)
negative_lemmas <- lemmatize_words(Negative_Token)
```

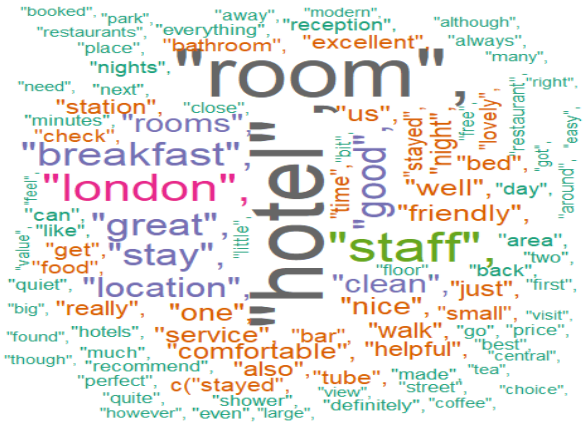
Creation of Document Term Matrix (DTM)

A Document-Term Matrix (DTM) is formed using the DocumentTermMatrix() function. In a DTM, rows represent documents and columns represent terms. The value in each cell indicates the frequency of a term in a specific document. This structured representation of text data is crucial for efficient text mining. DTM is made for both positive and negative lemmas under the variable “positive_DTM” and “negative_DTM”. The frequency of each word in the DTM are calculated.

```
# DTM for positive and negative lemmas####
positive_DTM <- DocumentTermMatrix(positive_lemmas)
dtm_positive <- as.matrix(positive_DTM)
frequency_positive <- colSums(dtm_positive)
frequency_positive <- sort(frequency_positive, decreasing=TRUE)
doc_length_positive <- rowSums(dtm_positive)

negative_DTM <- DocumentTermMatrix(negative_lemmas)
dtm_negative <- as.matrix(negative_DTM)
frequency_negative <- colSums(dtm_negative)
frequency_negative <- sort(frequency_negative, decreasing = TRUE)
doc_length_negative <- rowSums(dtm_negative)
```

World cloud generation

A word cloud is generated to identify the distribution of texts in both negative and positive reviews, the text which appears in a higher number would be big in the world cloud.



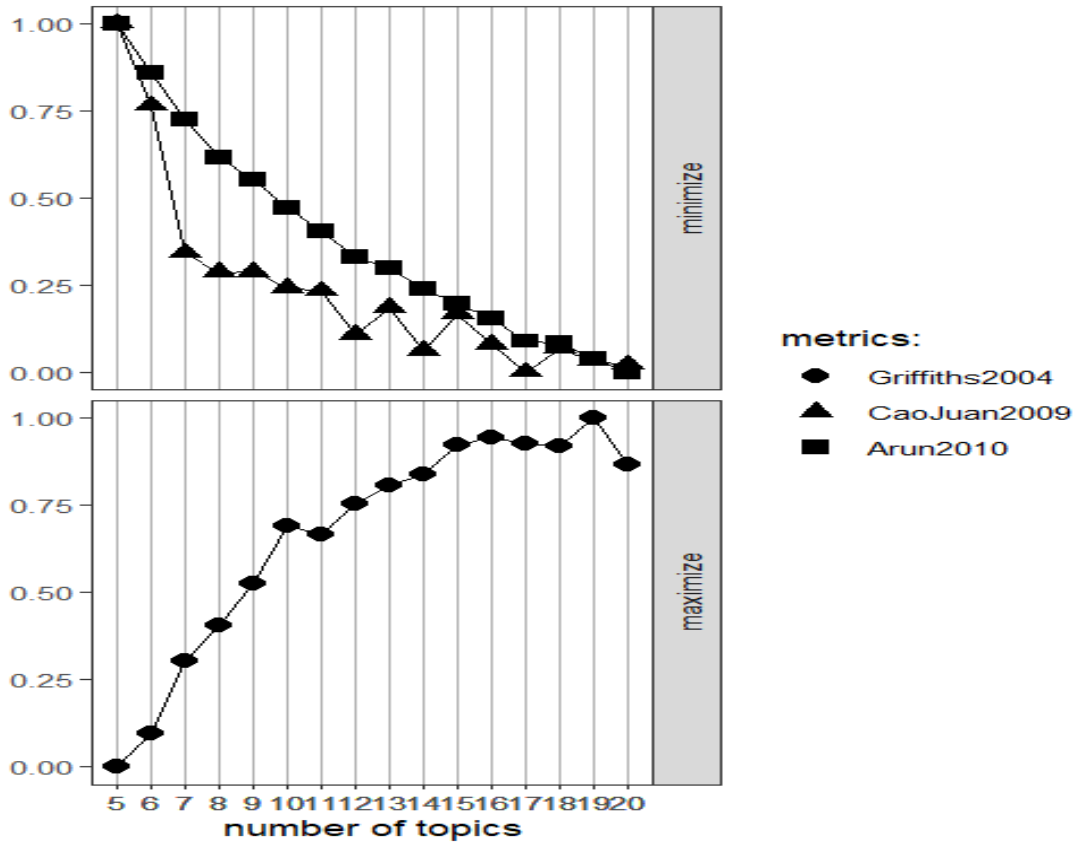
Positive word cloud which has words like room, hotel and breakfast have higher density

Negative word cloud with words hotel and room has a higher density

Topic number identification

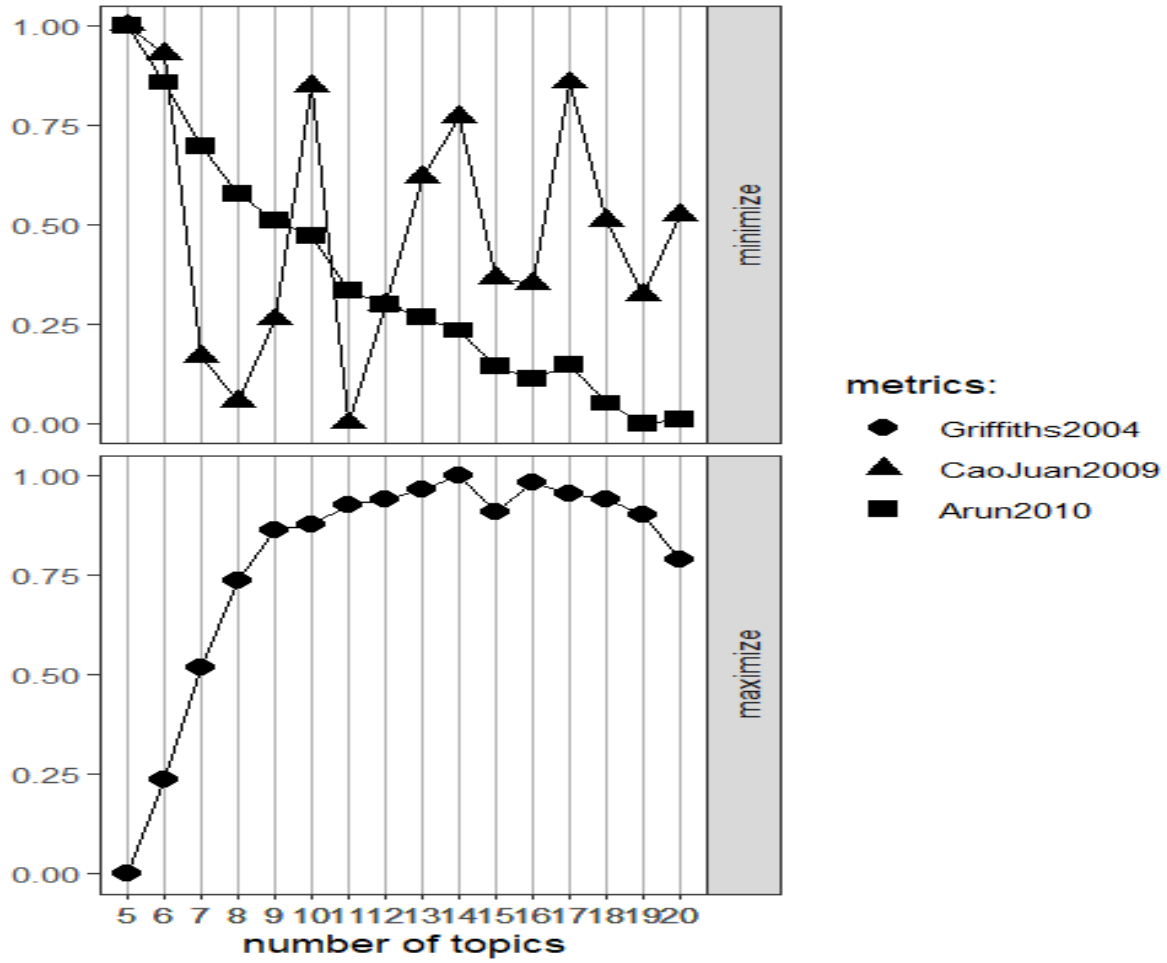
Determining the optimal number of topics for the Latent Dirichlet Allocation (LDA) model involves using the FindTopicsNumber() function. This function evaluates various metrics, such as "Griffiths2004", "CaoJuan2009", and "Arun2010", across different numbers of topics. To select the optimal number, the goal is to minimize the criteria for "Arun2010" and "CaoJuan2009" while maximizing "Griffiths2004". The results of this evaluation are visualized using the FindTopicsNumber_plot() function, which is crucial for ensuring the quality and interpretability of the topics generated by the LDA model.

```
#Topic number identification####
result_positive <- FindTopicsNumber(
  positive_DTM,
  topics = seq(from = 5, to = 20, by = 1),
  metrics = c("Griffiths2004", "CaoJuan2009", "Arun2010"),
  method = "Gibbs",
  control = list(seed = 348),
  mc.cores = 2L,
  verbose = TRUE
)
FindTopicsNumber_plot(result_positive)

result_negative <- FindTopicsNumber(
  negative_DTM,
  topics = seq(from = 5, to = 20, by = 1),
  metrics = c("Griffiths2004", "CaoJuan2009", "Arun2010"),
  method = "Gibbs",
  control = list(seed = 348),
  mc.cores = 2L,
  verbose = TRUE
```



The plot of positive topics

While analysing the plot it is at number of topics = 19 where Griffiths2004 is maximum and the other two metrics ", "CaoJuan2009", and "Arun2010" are minimum



The plot of negative topics

While analysing the plot it is at number of topics = 19 where Griffiths2004 is maximum and the other two metrics ", "CaoJuan2009", and "Arun2010" are minimum.

Topic modelling using LDA

A quantitative, unsupervised method for identifying abstract subjects in a set of documents is called topic modelling. Within this field, a statistical model called Latent Dirichlet Allocation (LDA) is very helpful for identifying latent patterns in big text datasets using text mining and natural language processing. According to this generative model, words appear in documents based on these underlying subjects, and their occurrences are somewhat deterministic and partially random.

```
positive_lda <- LDA(positive_DTM, k = 19, method = "Gibbs", control = list(seed = 348))
negative_lda <- LDA(negative_DTM, k = 19, method = "Gibbs", control = list(seed = 348))
```

The method used for estimation is Gibbs sampling, controlled by a specific seed for reproducibility. The result is a fitted model for each set of reviews (positive and negative), which can be used for further analysis and interpretation.

Topic Terms Extraction

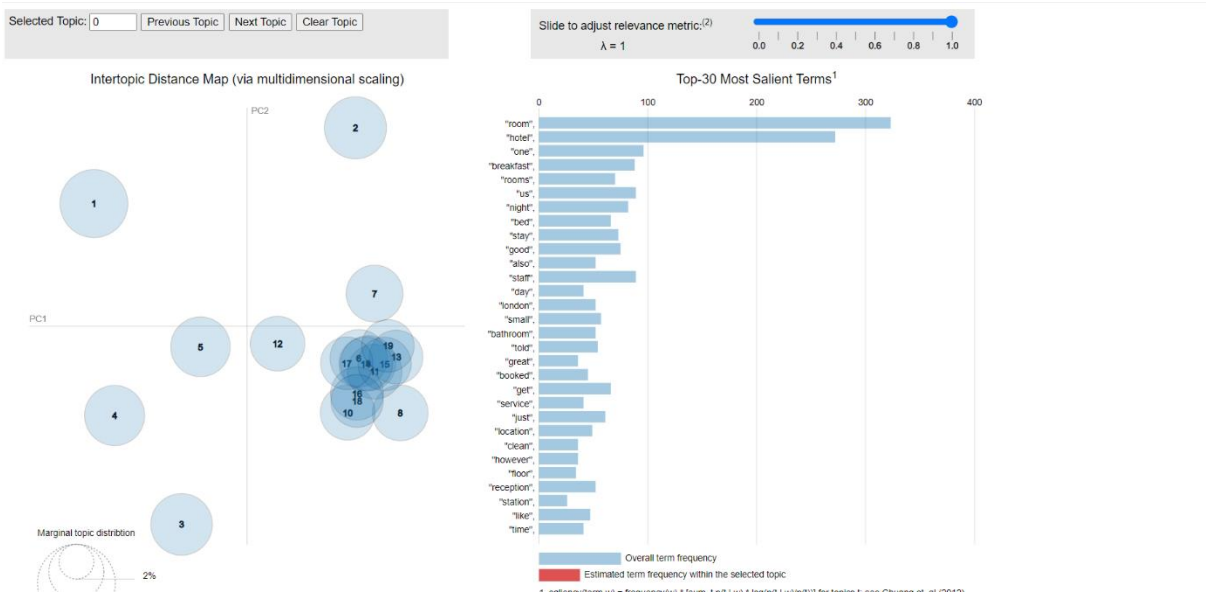
The terms represent the most defining words for each topic and can be used to interpret and understand the underlying themes of the topics. The extracted terms are stored for both positive and negative reviews. This step is crucial for understanding the content and context of the topics generated by the LDA model.

```
positive_topics <- terms(positive_lda, 10)
negative_topics <- terms(negative_lda, 10)
print(positive_topics)
print(negative_topics)
```

Topic Identification and Labelling

The LDAvis package in R provides a robust, interactive platform for the interpretation and visualization of topics in a topic model. This package is particularly adept at identifying and labeling topics derived from the Latent Dirichlet Allocation (LDA) method. One of the key features of LDAvis is its ability to highlight the top 30 most salient terms for each topic, providing a clear and concise summary of the topic's main themes. Relevance metric λ is adjusted to 0.6 for balance between the term frequency within a topic and its lift.

```
# For positive reviews
phi_positive <- posterior(positive_lda)$terms %>% as.matrix
theta_positive <- posterior(positive_lda)$topics %>% as.matrix

# For negative reviews
phi_negative <- posterior(negative_lda)$terms %>% as.matrix
theta_negative <- posterior(negative_lda)$topics %>% as.matrix

# Create the JSON object to feed the visualization in LDAvis
positive_vocab <- colnames(phi_positive) # Vocab list in DTM
json_lda_positive <- createJSON(phi = phi_positive, theta = theta_positive,
                                vocab = positive_vocab, doc.length = doc_length_positive,
                                term.frequency = frequency_positive)

serVis(json_lda_positive, out.dir = 'vis', open.browser = TRUE)

# For negative reviews
negative_vocab <- colnames(phi_negative)  # Vocab list in DTM
json_lda_negative <- createJSON(phi = phi_negative, theta = theta_negative,
                                vocab = negative_vocab, doc.length = doc_length_negative,
```



Visualisation using LDAvis

Top three factors that affects the satisfaction of the customers

Factors are identified based on the percentage of tokens from a topic in the total

The top three factors influencing customer satisfaction are:

1. **Hotel Room Design and Amenities:** This category includes various aspects of hotel room design and amenities. Key elements such as bed, bathroom, shower, TV, window, etc., are crucial components of hotel room design. Utilities like water, air, and noise management are essential for ensuring comfort and functionality.

2. **Customer Experience in Hotel Check-in Process:** This category revolves around the experience of checking into a hotel and interacting with hotel staff. It highlights the importance of customer satisfaction during the initial stage of their stay, which involves booking, arrival, room readiness, upgrades, interactions with receptionists, and timing considerations such as early or late check-in/out.

3. **Hospitality and Accommodation:** This label encompasses various aspects related to staying in a hotel or accommodation, including time (night, last, nights), payment (pay, paid, prices, price, per), services (reception, key), issues (problem), amenities (door), and work-related aspects (work, working).

The top three negative factors are:

1. **Communication with the Hotel:** Words like email, speak, booking, busy, and check depict factors related to communication with hotel management. The word 'busy' conveys a negative impression.

2. **Food and Restaurant Experience:** The words like waitress, sandwiches, minutes, returned, and drinks seem associated with the restaurant service. Words like problem, returned, and fix might be associated with a bad experience.

3. **Dissatisfaction with Facilities and Services:** Words like bed, room, shower, and small depict customer dissatisfaction with the services provided.

Limitations

While `LDAvis` provides a wealth of information on the topics, it does have certain limitations, primarily due to the following reasons:

- Negative terms appear in positive tokens and vice versa, which can lead to confusion in the interpretation of the topics.
- The number of both positive and negative terms is limited, which may restrict the comprehensiveness of the topic analysis.