Final-project-report

Raj Deelip - Raghul Sekar - Mohammad Rafi Shak

12/13/2021

Data Wranglling

1. Eliminate NA and other undefined values

2. Very few booking shows the adults number is larger than 4. So, we regards those booking as abnormal value. Remove unreasonable values that have more than 4 adults in a room.

3. Subset columns that are being used for further analysis

4. Format month from string format to number

5. Classify hotel__type and got__desired__roomtype as required for clustering

```
customerdata <- read.csv("hotel_bookings.csv",na.strings = "")
customerdata <- customerdata[!is.na(customerdata$children), ]
customerdata$children <- as.integer(customerdata$children)
```

```
## Warning: NAs introduced by coercion
```

```
customerdata <- customerdata[!is.na(customerdata$children), ]

#eliminate NA and other undefined value
customerdata$meal[customerdata$meal=='Undefined'] <- 'SC'
customerdata$children[is.na(customerdata$children)] <- 0
customerdata <- subset(customerdata, market_segment!='Undefined')
customerdata <- subset(customerdata, distribution_channel!='Undefined')

cat('the number of booking that adults number is lower than 5:',  nrow(subset(customerdata, adults <= 4
```

```
## the number of booking that adults number is lower than 5: 119369
```

```
cat('the number of booking that adults number is larger than 4:',  nrow(subset(customerdata, adults > 4
```

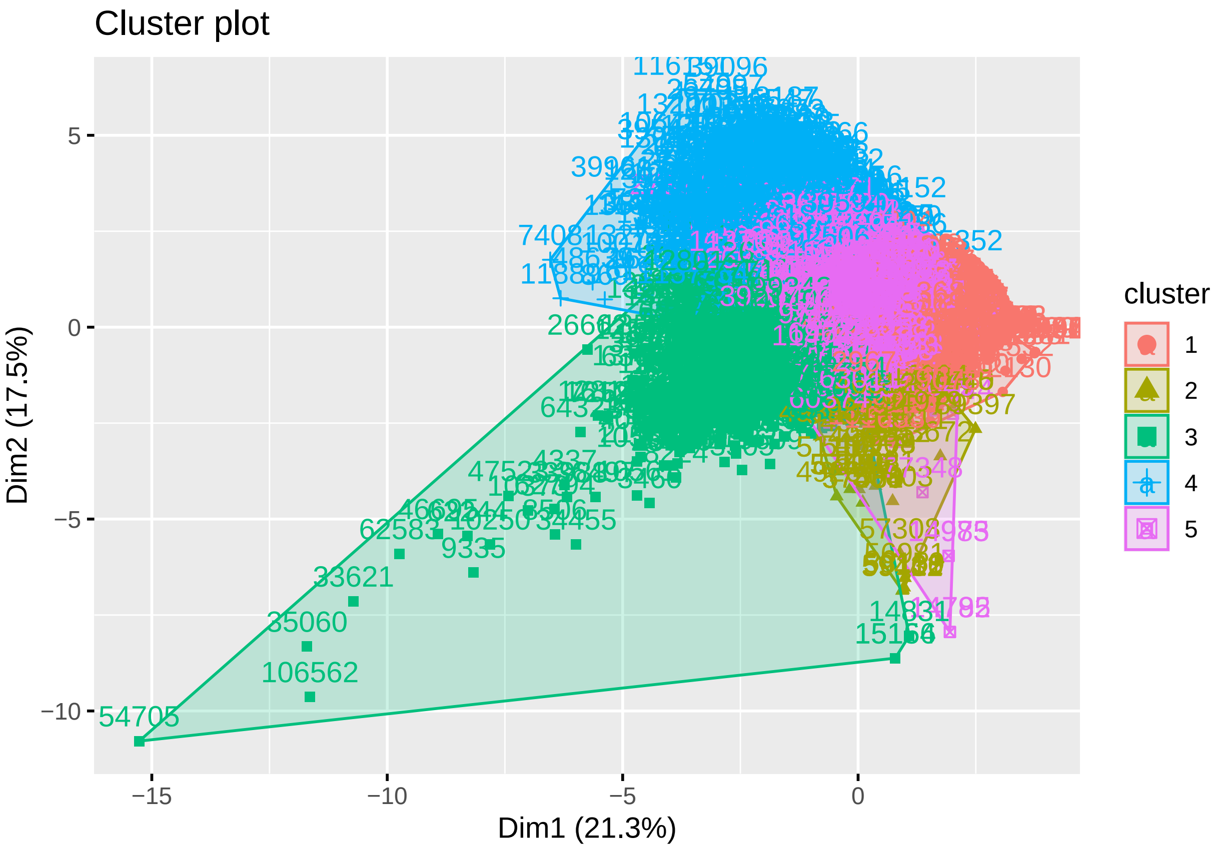
```
## the number of booking that adults number is larger than 4: 16
```

```
customerdata <- subset(customerdata, adults <= 4)

tempdataset<- customerdata[,c("adr","hotel","lead_time","is_canceled","arrival_date_month","is_repeated

tempdataset["Arrival_month"] <- match(tempdataset[,"arrival_date_month"],month.name)

tempdataset <- tempdataset %>%
  mutate(hotel_type= case_when(hotel == "Resort Hotel" ~ 1,TRUE  ~ 2)) %>%
  mutate(got_desired_roomtype= case_when(reserved_room_type == assigned_room_type ~ 1,TRUE  ~ 0))

tempdataset <- tempdataset[ , -which(names(tempdataset) %in% c("hotel","arrival_date_month","is_repeate
```

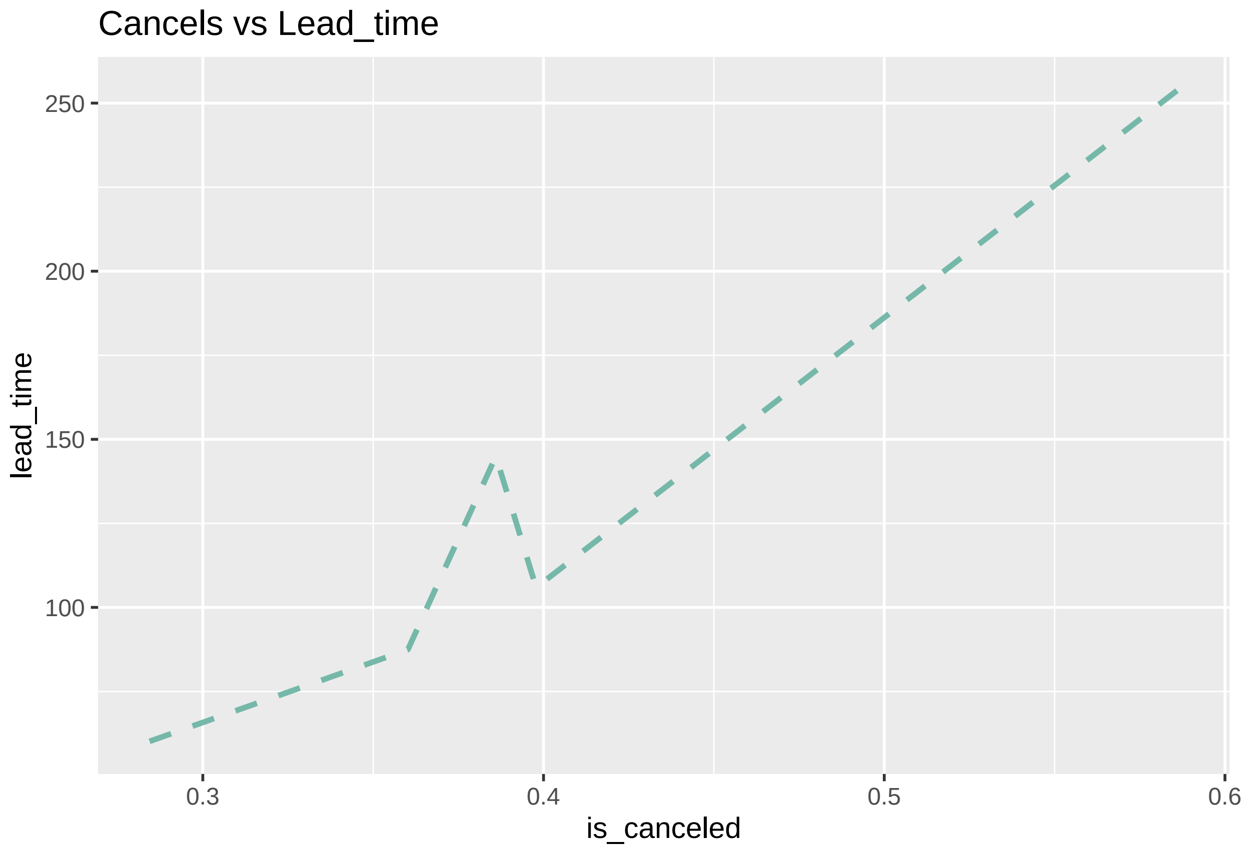
Clustering

Using fviz_nbclust, finding optimal number of clusters

Using K-Means, visualizing clusters



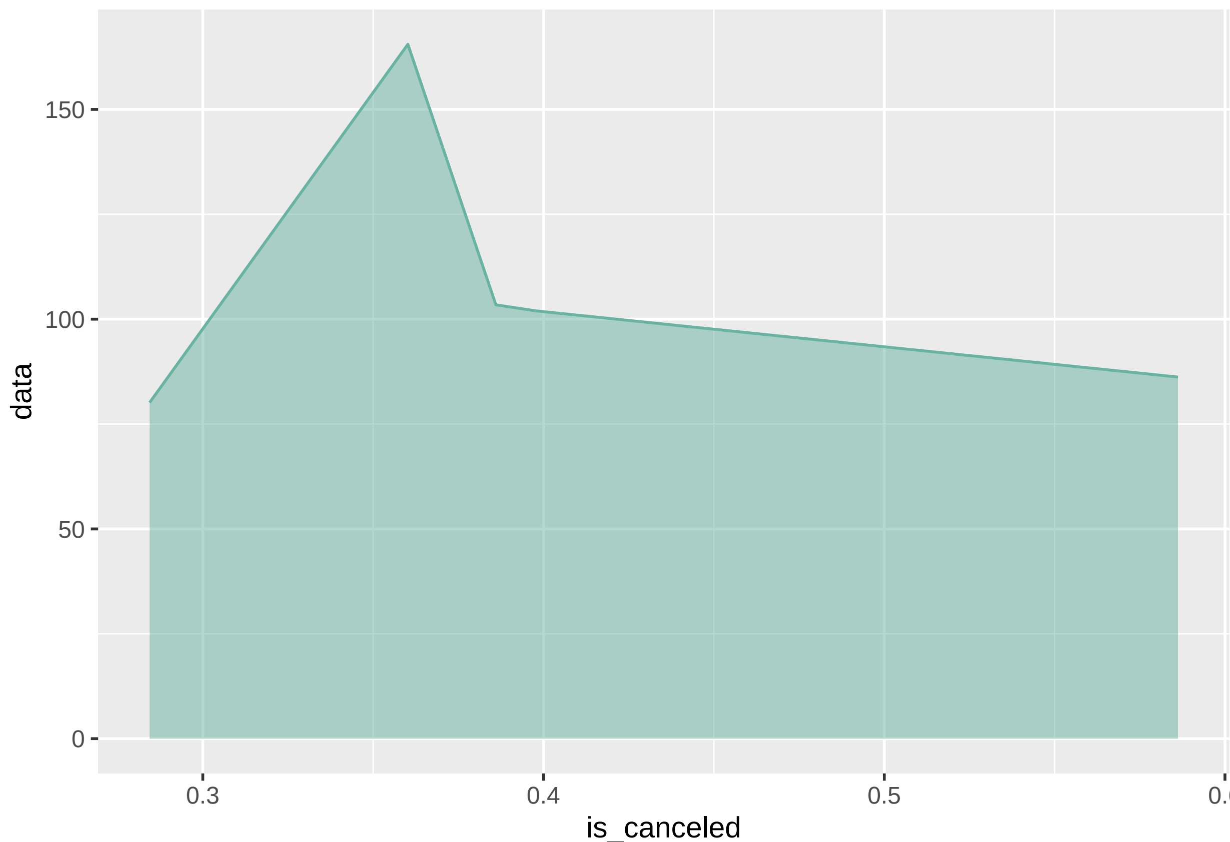
Evolution of Cancels vs Lead time from cluster data ## It is evident that, Cancels are mostly likely to happen if lead-time is more, meanwhile they might have changed plans or found a better deal



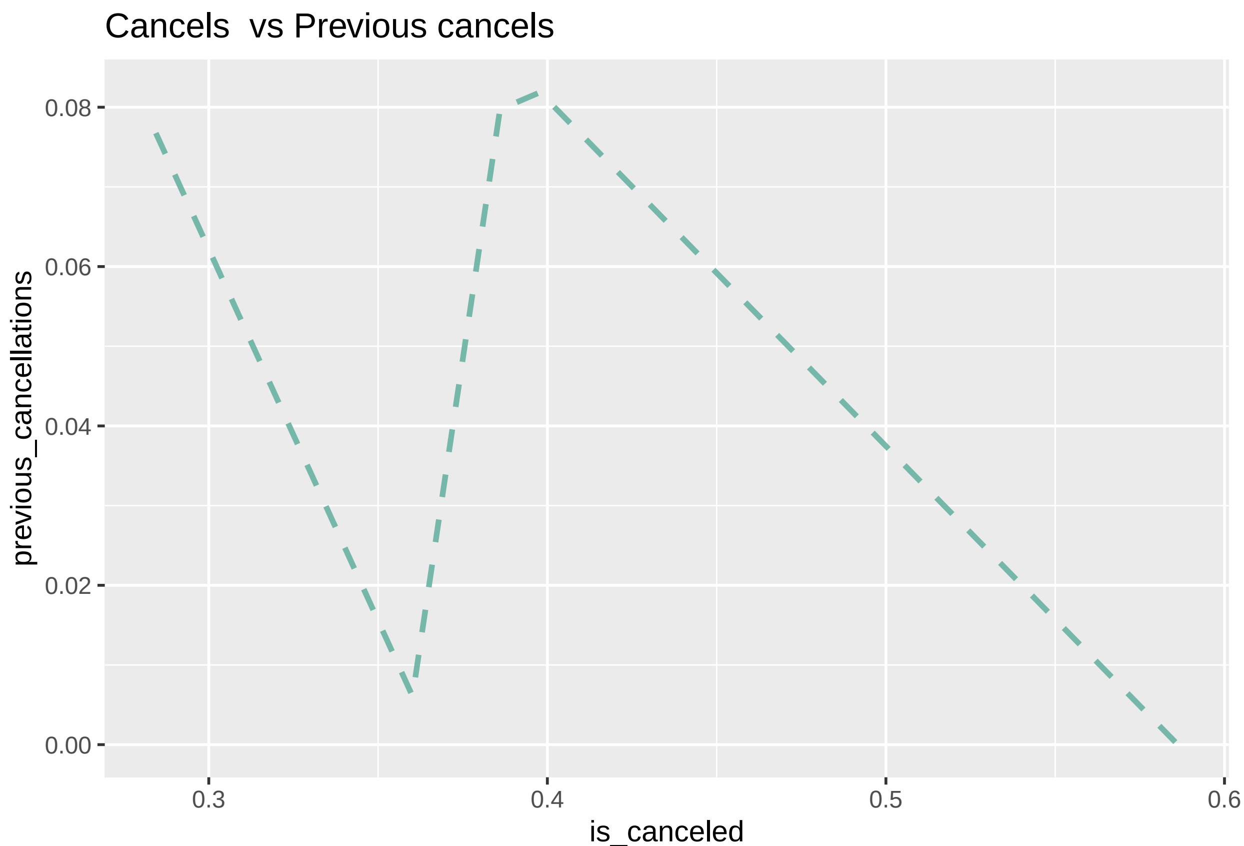
Cancels vs Lead_time
lead_time
250
200
150
100
0.3
0.4
0.5
0.6
is_canceled

Area plot for Cancels vs ADR

Mostly cancels are uniform with respect to ADR



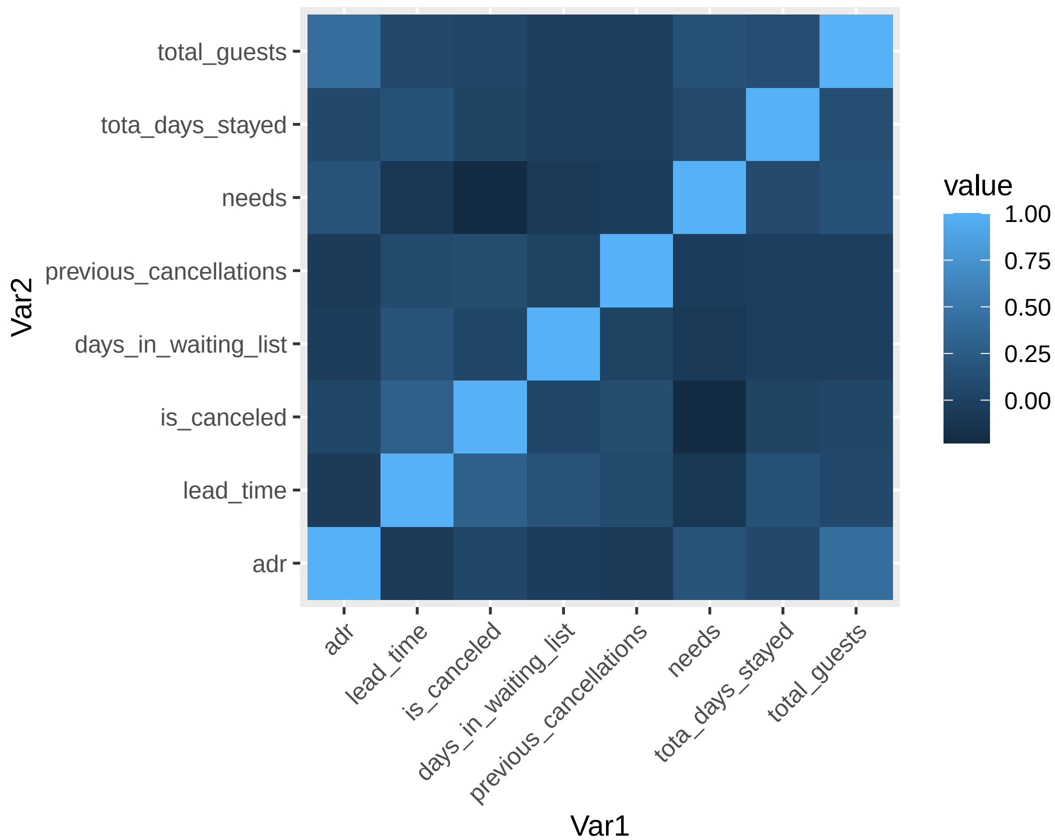
Likel-hood of Cancels with respect to previous cancels



Abbrovate ADR

Corelation heat map for all main attributes

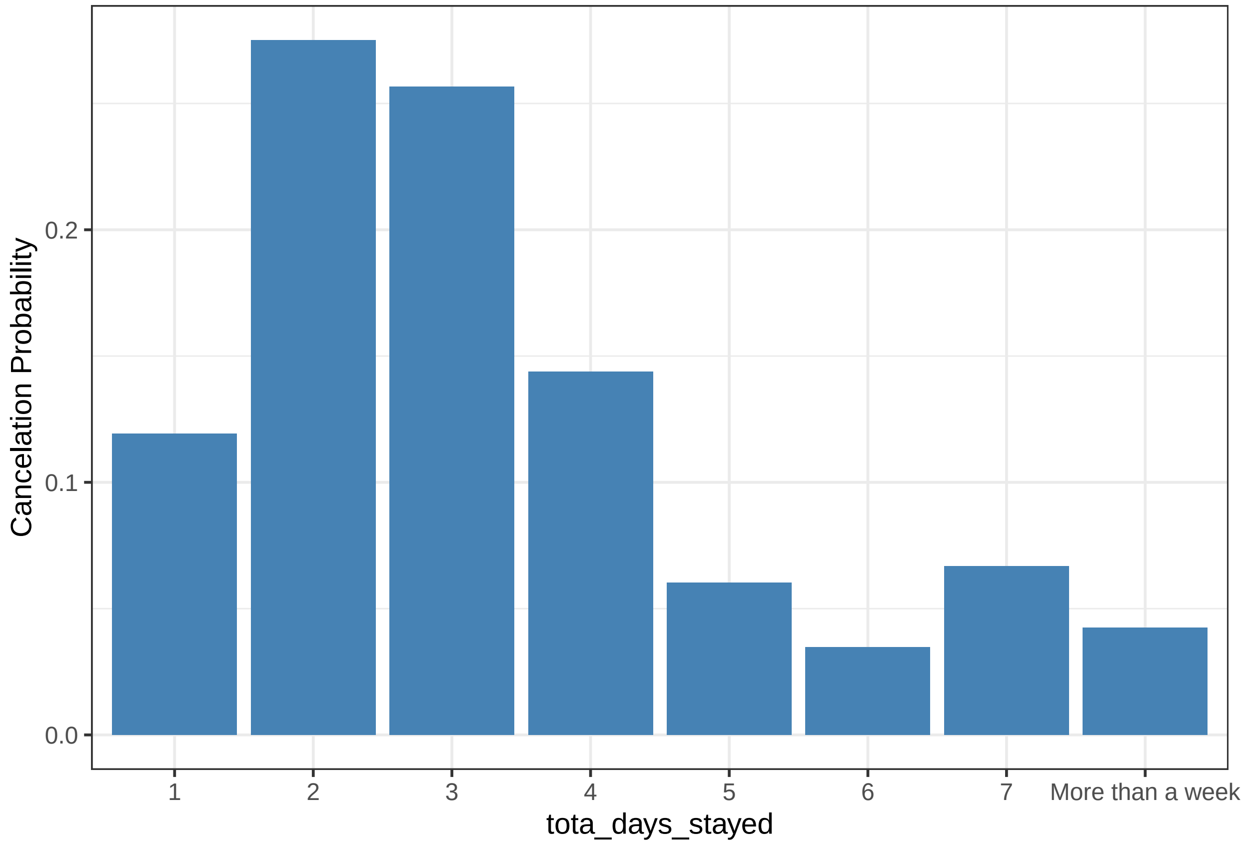
It is evident that, Cancels are mostly likely to happen if lead-time is more, meanwhile they might have changed plans or found a better deal



Probability

PMF and CDF for total_days_stayed in hotel a reservation

Bar plot showing the distribution of cancellation probability vs total_days

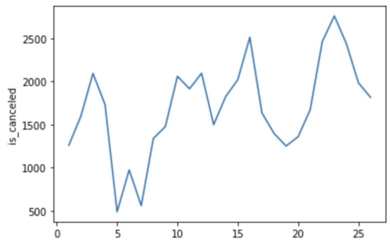


Time Series Analysis

SNS line plot for cancels per month over the time

```python
import pandas as pd
import seaborn as sns
from datetime import datetime

hotel_df = pd.read_csv('hotel_bookings.csv')

for i in range(len(hotel_df)):
  hotel_df.loc[i, 'arrival_date_month'] = datetime.strptime(hotel_df.loc[i, 'arrival_date_month'], "%B").month

hotel_df1 = hotel_df[["arrival_date_year", "arrival_date_month", "is_canceled"]]
hotel_df2 =  hotel_df1[hotel_df1["is_canceled"] == 1].groupby(["arrival_date_year", "arrival_date_month"]).agg({'is_canceled': 'sum'})

sns.lineplot(x=range(1, len(hotel_df2['is_canceled'])+1), y=hotel_df2['is_canceled'])
```

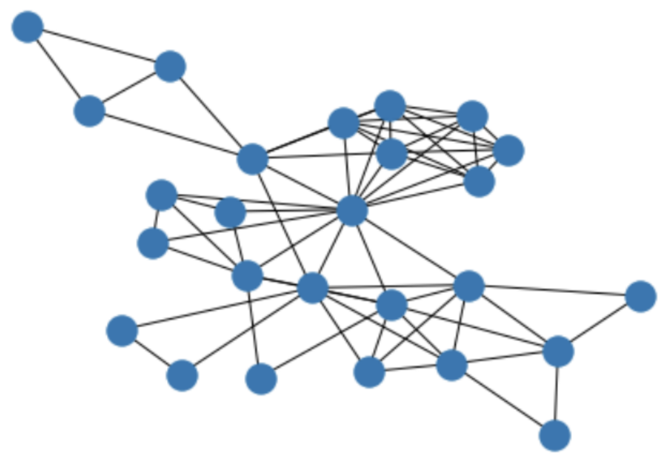
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6788fc1710>
```



Natural Visual Graph

```python
from ts2vg import NaturalVG
import numpy as np
g = NaturalVG()
df = hotel_df2['is_canceled']
g.build(df)
ig_g = g.as_igraph()
nx_g = g.as_networkx()
import networkx as nx
nx.draw_kamada_kawai(nx_g)

print('Number of Nodes:',ig_g.vcount())
print('Number of Links:',ig_g.ecount())
print('Average Degree:',np.mean(ig_g.degree()))
print('Network Diameter:',ig_g.diameter())
print('Average Path Length:',ig_g.average_path_length())
```

```
Number of Nodes: 26
Number of Links: 67
Average Degree: 5.153846153846154
Network Diameter: 5
Average Path Length: 2.3476923076923075
```



Horizontal Visual Graph

```python
from ts2vg import HorizontalVG
g = HorizontalVG()
df = hotel_df2['is_canceled']
g.build(df)
ig_g = g.as_igraph()
nx_g = g.as_networkx()
import networkx as nx
nx.draw_kamada_kawai(nx_g)

print('Number of Nodes:',ig_g.vcount())
print('Number of Links:',ig_g.ecount())
print('Average Degree:',np.mean(ig_g.degree()))
print('Network Diameter:',ig_g.diameter())
print('Average Path Length:',ig_g.average_path_length())
```

```
Number of Nodes: 26
Number of Links: 42
Average Degree: 3.230769230769231
Network Diameter: 9
Average Path Length: 3.6123076923076924
```

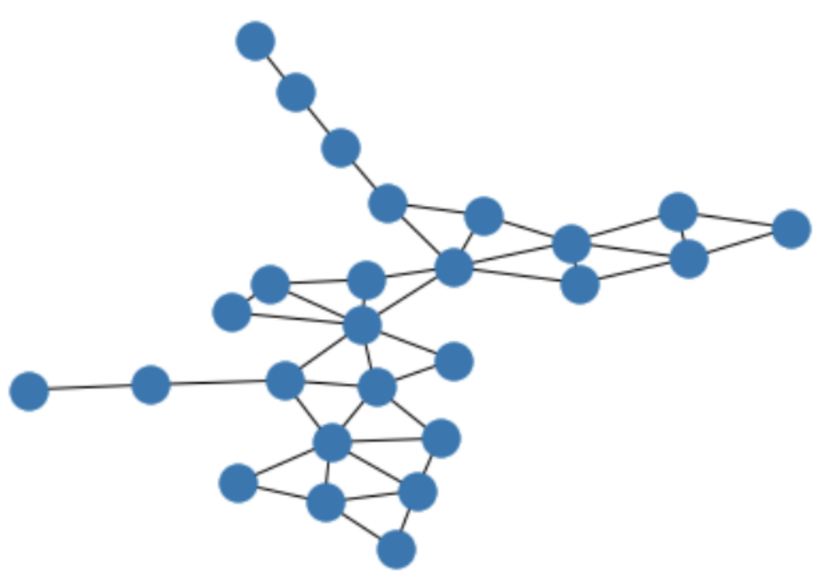
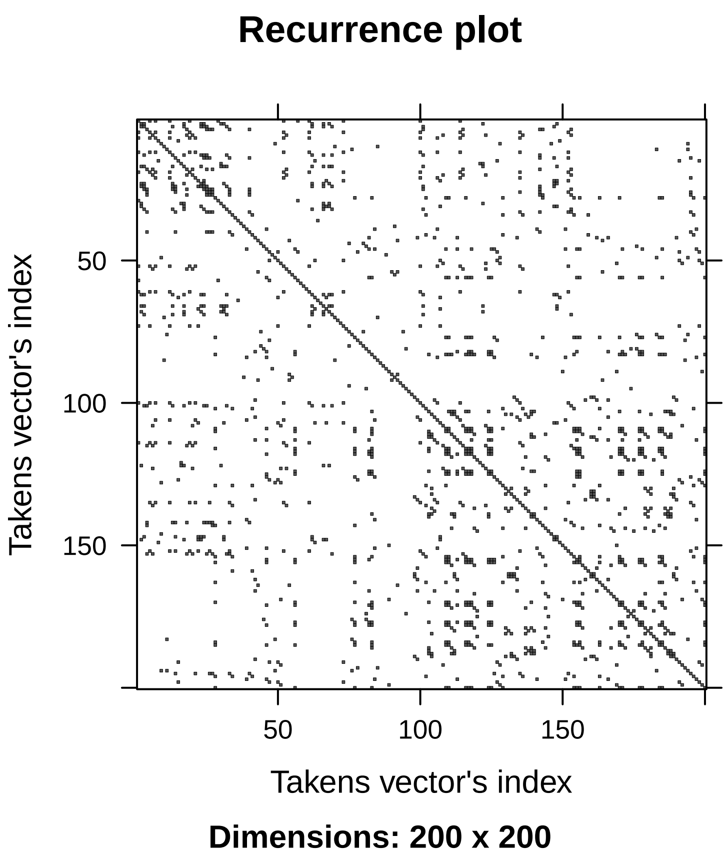


Figure 1: Horizontal Visual Graph for cancels per month from Oct-2014 - Aug 2017

RQA analysis

Reccurence plot of acc signals for Cancels

```
time_s <- df$Cancels[400:600]
rqa.analysis=rqa(time.series = time_s, embedding.dim=2, time.lag=1 ,
                 radius=10,lmin=2,do.plot=TRUE,distanceToBorder=2)
```



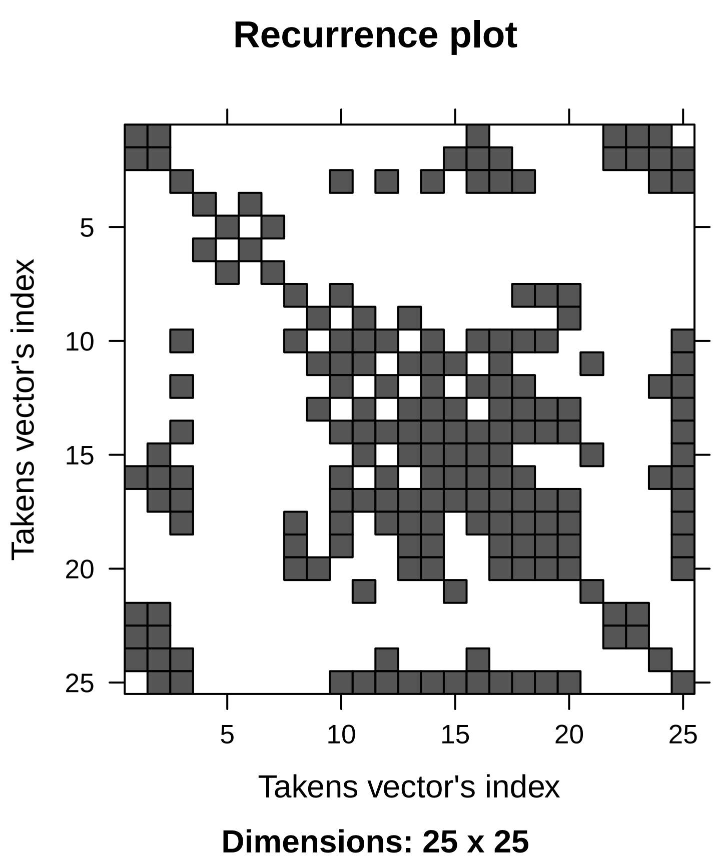

```
# plot(rqa.analysis)
# rqa.analysis
```

recurrence plot for month cancel percentages

```
df = hotel_data %>%
    filter(country %ni% c('NULL' ) ) %>%
    group_by(arrival_date_year, arrival_date_month) %>%
    summarise(Cancels_per = sum(is_canceled==1) / n(),
              Cancels = sum(is_canceled==1)) %>%
    arrange(arrival_date_year, arrival_date_month)
```

```
## `summarise()` has grouped output by 'arrival_date_year'. You can override using the `.groups` argumen
```

```r
time_s <- df$Cancels_per
rqa.analysis=rqa(time.series = time_s, embedding.dim=2, time.lag=1,
                 radius=.05,lmin=2,do.plot=TRUE,distanceToBorder=2)
```




```r
# plot(rqa.analysis)
```

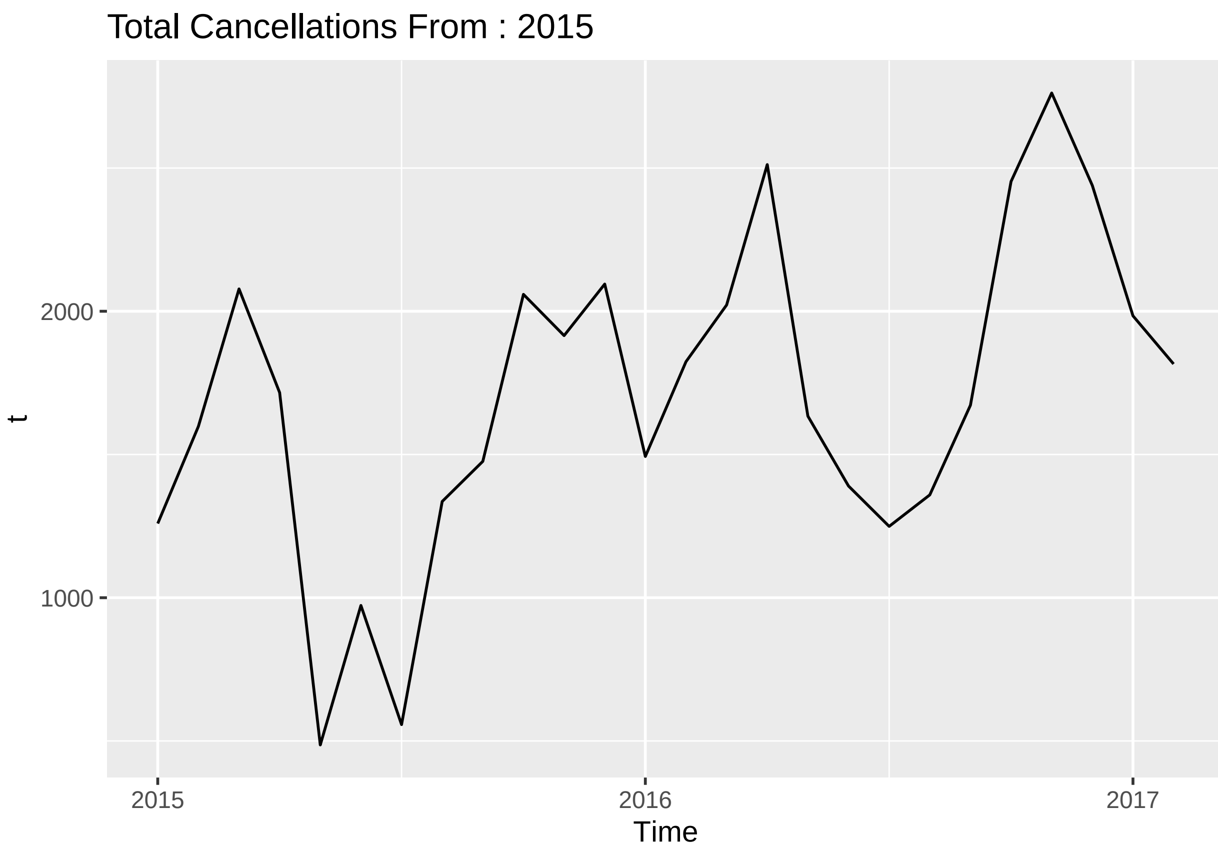
```r
library(seasonal)
```

```
##
## Attaching package: 'seasonal'

## The following object is masked from 'package:tibble':
##
##     view
```

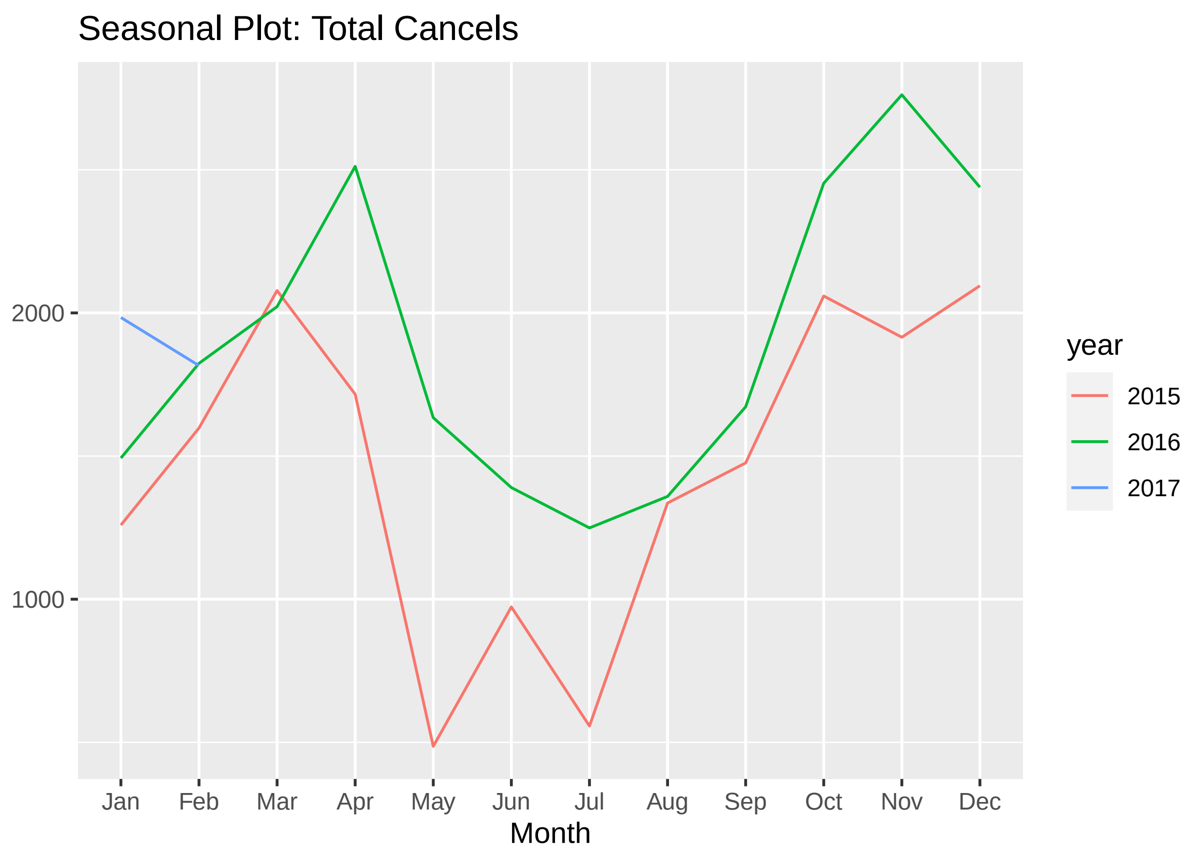
```r
library(fpp2)
```

```
## -- Attaching packages --------------------------------------------- fpp2 2.4 --

## v forecast  8.15     v expsmooth 2.3
## v fma       2.4

##
```

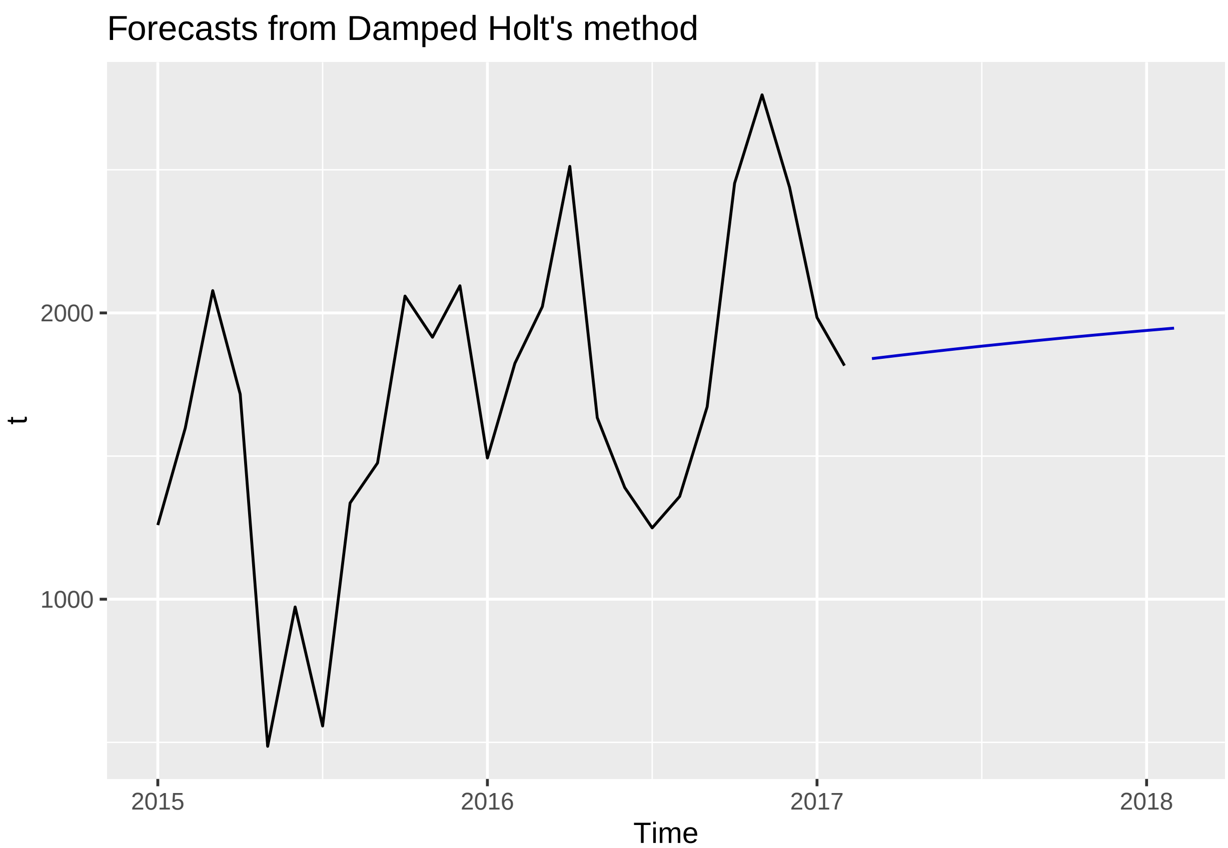
```
t = ts(data = df$Cancels, frequency=12, start = 2015 )
autoplot(t, main = "Total Cancellations From : 2015")
```



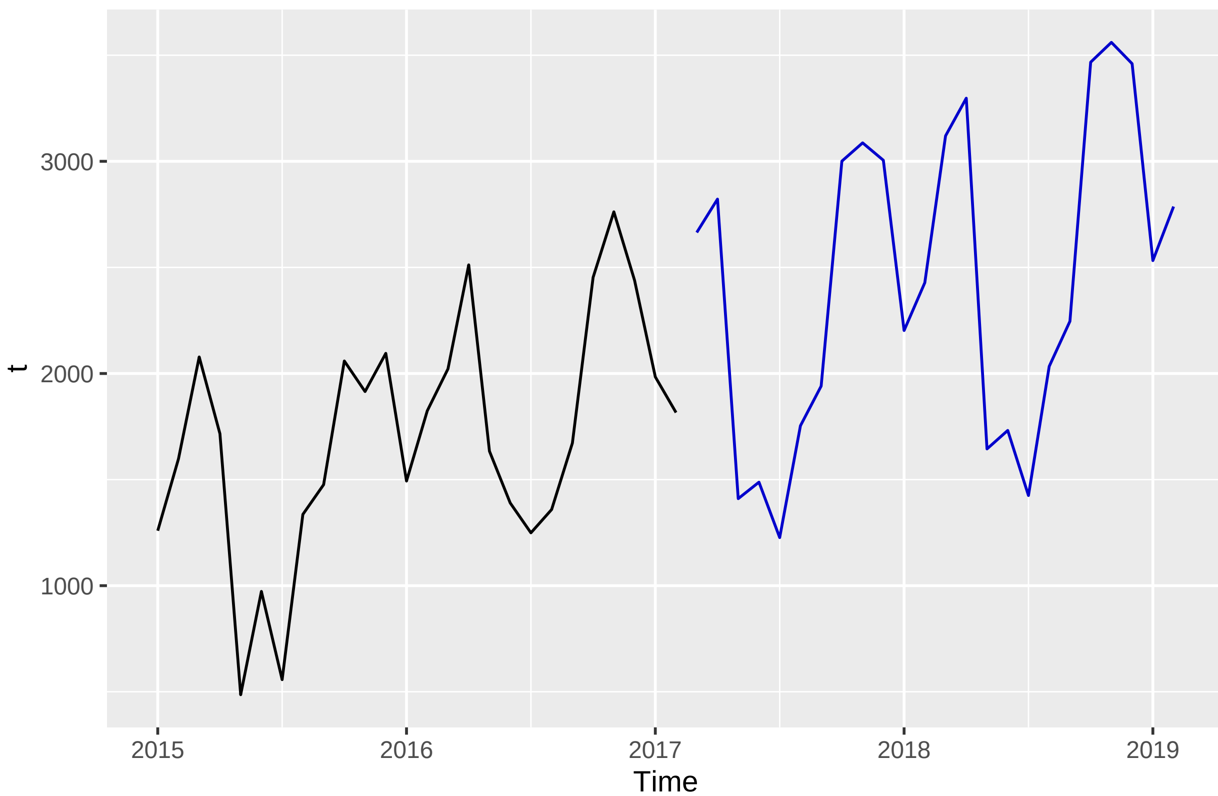

```
ggseasonplot(t, main = "Seasonal Plot: Total Cancels")
```



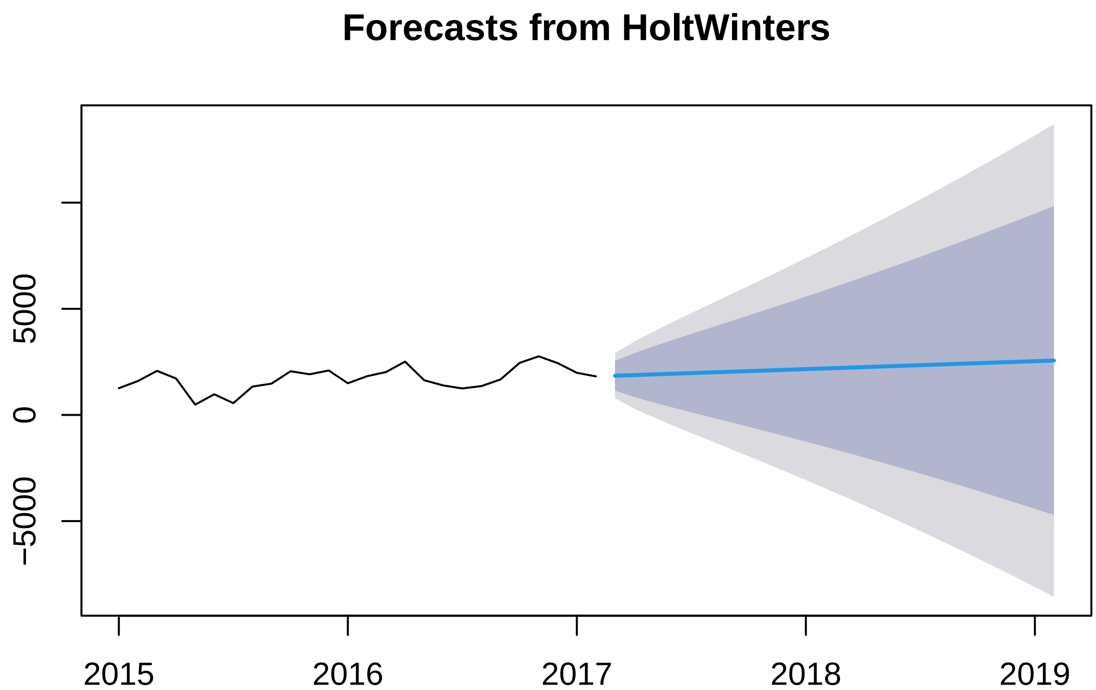

```
autoplot(holt(t, damped = TRUE, h = 12), PI = FALSE)
```




```
autoplot(t) +
  autolayer(hw(t, seasonal = "multiplicative", PI=FALSE))
```




```
plot(forecast(HoltWinters(t, gamma=FALSE)))
```


Forecasts from HoltWinters
−5000
0
5000
2015
2016
2017
2018
2019

Permutation Entropy of Cancels

Entropy is ~0.93 very high indicating, randomness of cancel behaviour

Complexity is ~ 0.05

```
import pandas as pd
from datetime import datetime
import seaborn as sns

hotel_df = pd.read_csv('hotel_bookings.csv')
for i in range(len(hotel_df)):
  hotel_df.loc[i, 'arrival_date_month'] = datetime.strptime(hotel_df.loc[i, 'arrival_date_month'], "%B").month

hotel_df1 = hotel_df[["arrival_date_year", "arrival_date_month", "is_canceled"]]
hotel_df2 =  hotel_df1[hotel_df1["is_canceled"] == 1].groupby(["arrival_date_year", "arrival_date_month"]).agg({'is_canceled': 'sum'})

sns.lineplot(x=range(1, len(hotel_df2['is_canceled'])+1), y=hotel_df2['is_canceled'])

op_cancles = ordinal_patterns(hotel_df2['is_canceled'],3,1)
print("Permutation Entropy Cancels=", p_entropy(op_cancles))
print("Complexity Cancels=", complexity(op_cancles))
```

```
Permutation Entropy Cancels= 0.937550625656298
Complexity Cancels= 0.0555136269980162
```

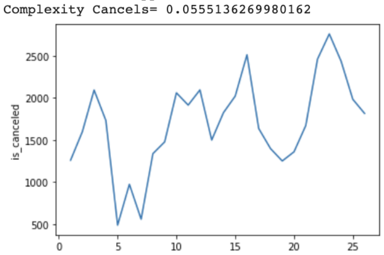


Figure 2: Lineplot for cancels per month from Oct-2014 - Aug 2017

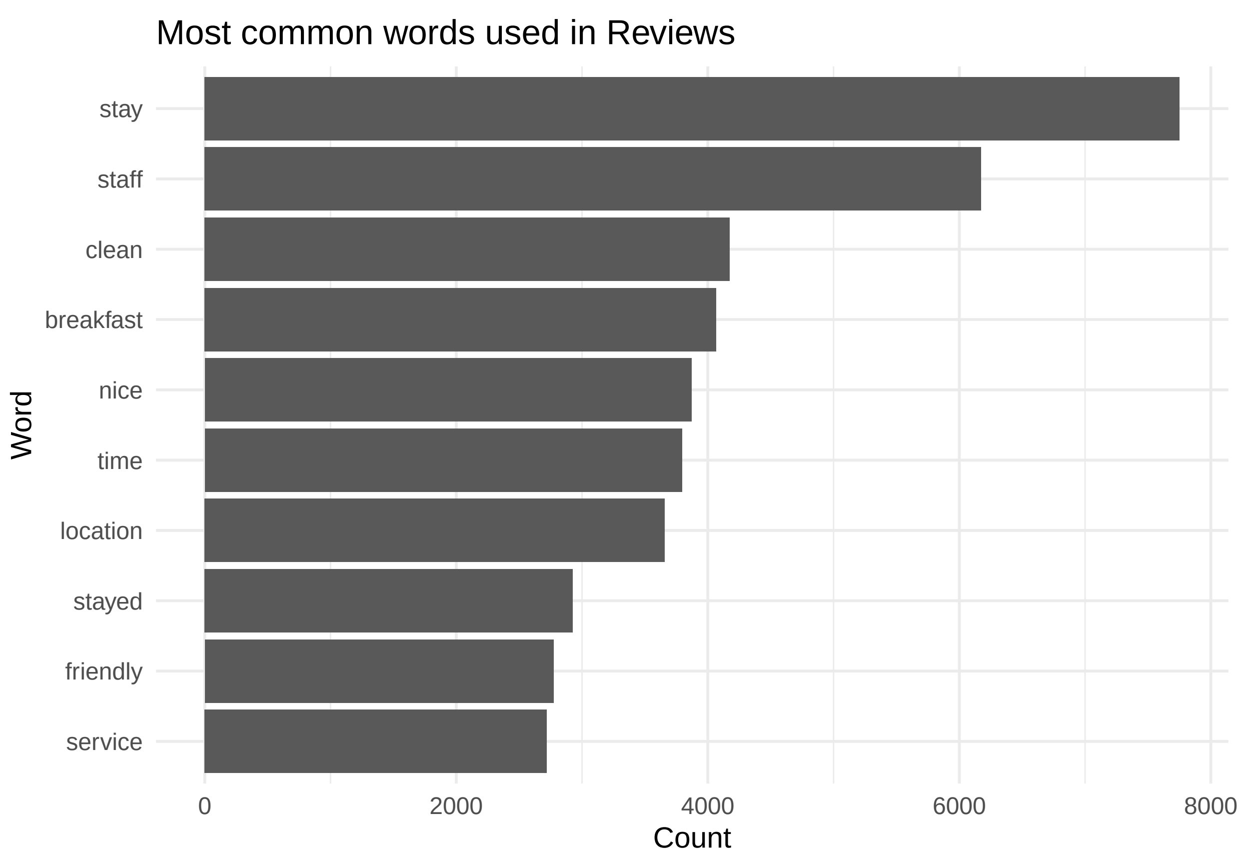
Text Analysis

```
## [1] "/Users/mohammad.rafi.shaik/Desktop/NEU/FDA - IE_5374/Project 2"

## Rows: 10000 Columns: 26

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (19): id, address, categories, primaryCategories, city, country, keys, ...
## dbl   (3): latitude, longitude, reviews.rating
## lgl   (1): reviews.dateAdded
## dttm  (3): dateAdded, dateUpdated, reviews.date

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Removing stop words and tokenizing the text

bar plot for most common words

```
## Selecting by n
```



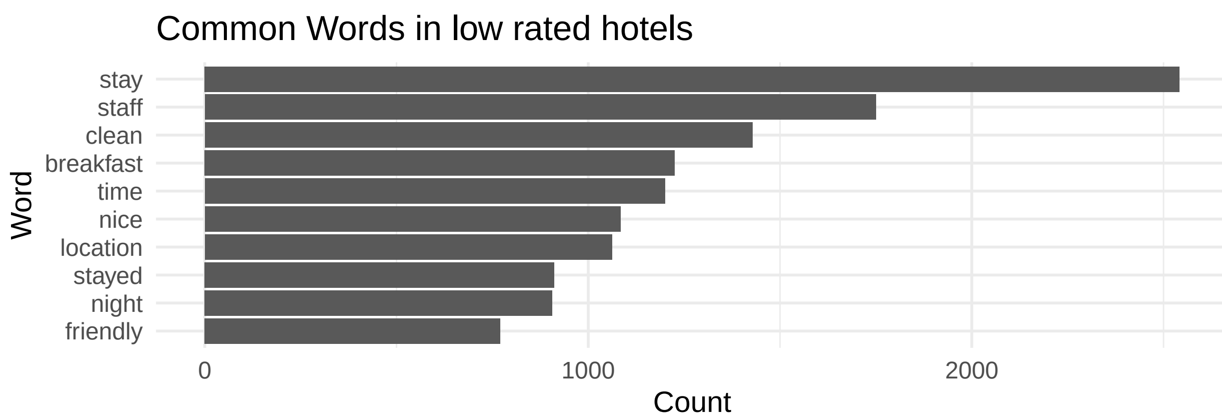
Wordcloud with top 100 words

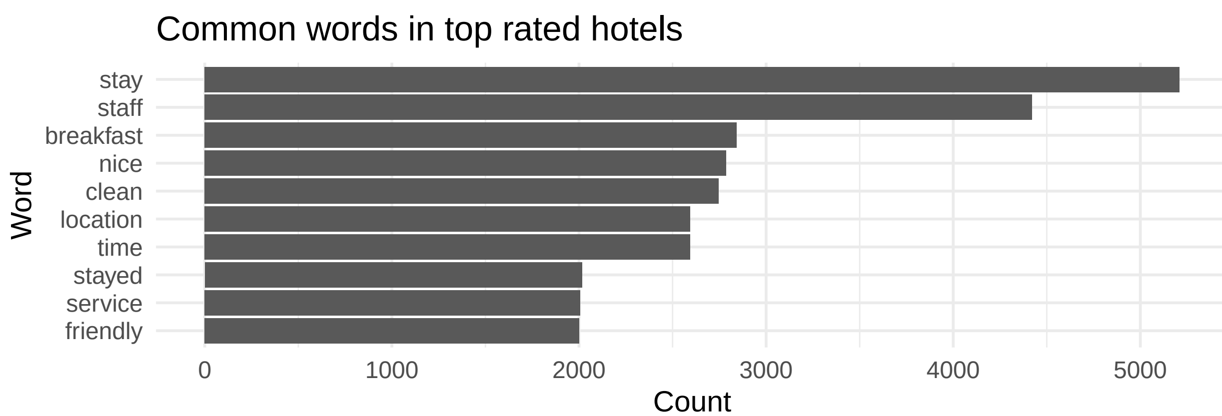
```
library(wordcloud)

pal <- brewer.pal(10, "BrBG")
tidy_reviews %>%
  count(word) %>%
  with(wordcloud(word, n, max.words = 100, random.order=FALSE, colors = pal))
```

As seen from the bar graph and word cloud, the most common words in reviews are stay, staff, location, time,clean, breakfast among.

So these are the topics of interest amon consumers and the hotels should look to make improvements on them.

```
## Selecting by n
## Selecting by n

##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```





Sentiment

```
## Selecting by n
```

```
## # A tibble: 10 x 2
##    word2     `sentiment value from prev word`
##    <chr>                                <dbl>
##  1 breakfast                             1.46
##  2 clean                                 2.57
##  3 friendly                              2.62
##  4 location                              3.00
##  5 nice                                  2.02
##  6 service                               2.69
##  7 staff                                 2.25
##  8 stay                                  2.84
##  9 stayed                                1.38
## 10 time                                  2.19
```

```r
top_provinces <- reviews %>%
  count(province, sort=TRUE) %>%
  top_n(6) %>%  select(province)
```

```
## Selecting by n
```

```r
top_words <-tidy_reviews %>%
   count(word, sort=TRUE) %>%
   top_n(100) %>%  select(word)
```

```
## Selecting by n
```

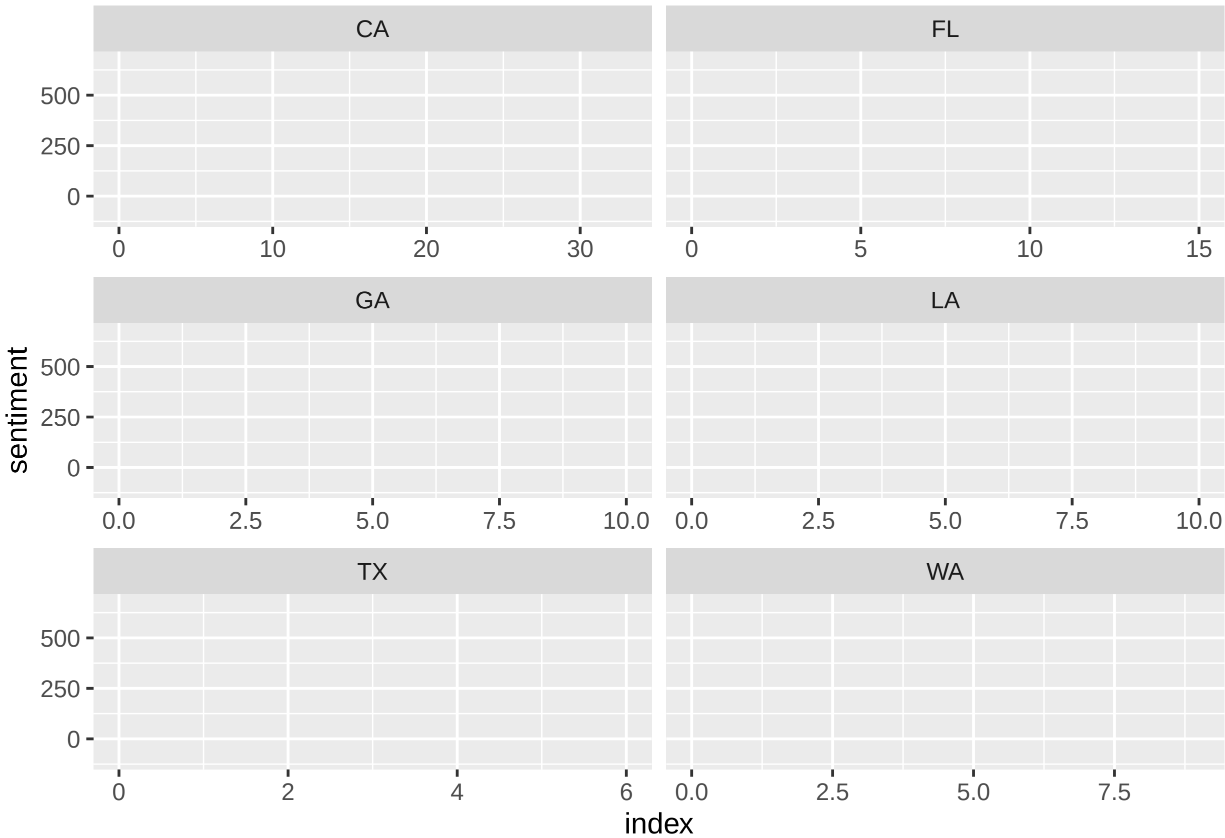
```r
tidy_sub_df <- reviews %>%
  select(province, reviews.text) %>%
  unnest_tokens(word, reviews.text, token = "words") %>%
  filter(!word %in% stop_words$word,
         !word %in% str_remove_all(stop_words$word, "'"),
         !word == 'hotel',
         str_detect(word, "[a-z]"))

c_df <- reviews %>%
  filter(province %in% top_provinces$province) %>%
  select(province, reviews.text, reviews.date) %>%
  group_by(province) %>%
  mutate(review_number = row_number()) %>%
  ungroup()


library(tidytext)
tidy_words <- c_df %>%
  unnest_tokens(word, reviews.text)


data(stop_words)

tidy_words <- tidy_words %>%
  anti_join(stop_words)
```

```
## Joining, by = "word"
```

```r
df_sentiment <- tidy_words %>%
  inner_join(get_sentiments("bing")) %>%
  count(province, index = review_number %/% 80, sentiment) %>%
  pivot_wider(names_from = sentiment, values_from = n, values_fill = 0) %>%
  mutate(sentiment = positive - negative)
```

```
## Joining, by = "word"
```

```r
ggplot(df_sentiment, aes(index, sentiment, fill = province)) +
  facet_wrap(~province, ncol = 2, scales = "free_x")
```


CA
FL
GA
LA
TX
WA
sentiment
index
500
250
0
0
10
20
30
5
15
0.0
2.5
5.0
7.5
10.0
2
4
6