R Notebook

```
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(jsonlite)
library(tidytext)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:randomForest':
##
##     margin
```

```
library(curl)
```

```
## Using libcurl 7.79.1 with LibreSSL/3.3.6
```

```
library(RCurl)
library(urltools)
library(httr)
```

```
##
## Attaching package: 'httr'
```

```
## The following object is masked from 'package:curl':
##
##     handle_reset
```

```
library(rvest)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

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

```
## The following objects are masked from 'package:stats':
##
```

```
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(textutils)
library(stringr)
library(tm)
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

```
## The following object is masked from 'package:httr':
##
##     content
```

```
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
library(tidyr)
```

```
##
## Attaching package: 'tidyr'
```

```
## The following object is masked from 'package:RCurl':
##
##     complete
```

```
library(broom)
library(purrr)
```

```
##
## Attaching package: 'purrr'
```

```
## The following object is masked from 'package:jsonlite':
##
##     flatten
```

```
library(modelr)
```

```
##
## Attaching package: 'modelr'
```

```
## The following object is masked from 'package:broom':
##
##     bootstrap
```

```
library(ggbeeswarm)
library(magrittr)
```

```
##
## Attaching package: 'magrittr'
```

```
## The following object is masked from 'package:purrr':
##
##     set_names
```

```
## The following object is masked from 'package:tidyr':
```

```
##
##     extract
```

```
library(rsample)
```

Web Scrapping

```
#To get WebAPI
#Create an account on https://www.scrapingbee.com/ and get the API key. Its a very simple process. The
```

```
# URL parsing to make a call to scrapingbee

my_url <- 'https://app.scrapingbee.com/api/v1'
my_api_key <- 'HMO9LWHYYM3R66P93XAJBRRFTJXPKQ3GZ9HHMERD8ORE7AFB4N4ZWD9A62RUC4R1FNTBEQZ2SOOQ5ZQN'
my_custom_google <- TRUE
```

```
states <- c("Alabama","Alaska","Arizona","Arkansas","California","Colorado","Connecticut","Delaware","D
```

```
length(states)
```

```
## [1] 51
```

```
# additional URL needs to be passed to scrapingbee API call so that it gets the required page

final_url <- 'https://www.airbnb.com/s/{state}--United-States/homes?tab_id=home_tab&refinement_paths%5B
```

```
scrape_airbnb <- function(my_api_key, state_url) {
  api_base_url <- "https://app.scrapingbee.com/api/v1/"
  api_response <- getForm(
   uri = api_base_url,
  .params = c(
    api_key = my_api_key,
    url = state_url
   )
  )
  return(api_response)
}
```

```
result_df <- data.frame(place_name = character(),
                        description = character(),
                        state = character(),
                        views = numeric(),
                        from_date = as.Date(character()),
                        to_date = as.Date(character()),
                        price = numeric(),
                        original_price = numeric(),
                        price_str = character(),
                        rating = numeric(),
                        no_of_rates = numeric())
```

```
extractDate <- function(date){
  date_list<-list()
  year <-format(Sys.Date(), "%Y")
  month1 <- word(date,1)
  month2 <- ifelse(is.na(as.numeric(word(date,4))),word(date,4),month1 )
  ddate <- stringr::str_extract_all(date, "\\d+")
```

```
  ddate <- strsplit(ddate[[1]]," ")
  #ddate<-sapply(strsplit(date, " - "), "[[", 1)
  if(nchar(ddate[[1]])==1)
    ddate[[1]]<-paste("0",ddate[[1]],sep='')
  if(nchar(ddate[[2]])==1)
    ddate[[2]]<-paste("0",ddate[[2]],sep='')
  Date_1<-as.Date(paste(month1,ddate[[1]],year,sep = ''),"%B%d%Y")
  Date_2<-as.Date(paste(month2,ddate[[2]],year,sep = ''),"%B%d%Y")
  date_list[[1]]<-Date_1
  date_list[[2]]<-Date_2
  return(date_list)
 #Date_1<-as.Date(with(df1,paste(Year,Month,Day,sep="-")),"%Y-%m-%d")
}
```

```
retrieve_airbnb_details <- function(attributes_extracted, state){
    for (i in seq_along(attributes_extracted)) {
      item <- attributes_extracted[[i]]
      place_name <- item[1]
      description <- item[2]
      view <- as.numeric(str_extract(item[3], "\\d+"))
      date<-extractDate(item[4])
      from_date <- date[[1]]
      to_date <- date[[2]]
      price_str <- item[5]
      price <- as.numeric(str_extract(str_extract(price_str, "\\$\\d+\\s*per night"), "\\d+"))
      if (!is.na(price)){
        if (str_detect(price_str, "originally")){
          original_price <- as.numeric(str_extract(price_str, "(?<=originally \\$)\\d+"))
        } else{
          original_price <- price
        }
      } else{
         original_price <- NA
      }

      rating <- as.numeric(str_extract(item[6], "\\d+\\.\\d"))
      no_of_rates <- as.numeric(str_extract(item[6], "(?<=\\()\\d+(?=\\))"))
      result_df <- rbind(result_df, data.frame(place_name, description, state, view, from_date, to_date
    }
    return(result_df)
}
```

```
#for (state in states) {
#  state_url <- gsub("\\{state\\}", state, final_url)
#  print(state)
#  added_places <- list()
#  attributes_extracted_all <- list()
#  for (offset in seq(0, 90, by = 18)){
#      offset_url <- gsub("\\{page_no\\}", offset, state_url)
#      offset_response <- scrape_airbnb(my_api_key, offset_url)
#      elements_extracted <- read_html(offset_response) %>%
#                            html_elements(".g1qv1ctd.cb4nyux.dir.dir-ltr") %>%
#                           html_text2()
#      attributes_extracted <- strsplit(elements_extracted, "\n")
```

```
#     i <- 1
#     while (i < length(attributes_extracted)) {
#       place_name <- attributes_extracted[[i]][1]
#       if (place_name %in% added_places) {
#         attributes_extracted <- attributes_extracted[-i]
#       } else {
#         added_places <- c(added_places, place_name)
#       i <- i + 1
#       }
#     }
#     attributes_extracted_all <- c(attributes_extracted_all, attributes_extracted)
#  }
#  result_df <- retrieve_airbnb_details(attributes_extracted_all, state)
#}
```

```
df_airbnb_data <- read.csv('airbnb_data_scrapped_final.csv')
df_airbnb_data <- subset(df_airbnb_data, select = -c(from_date_1, to_date_1))
```

```
head(df_airbnb_data)
```

```
##                      place_name
## 1 Private room in New Brunswick
## 2      Home in Neptune Township
## 3     Private room in Manasquan
## 4         Cottage in Lavallette
## 5    Guesthouse in Seaside Park
## 6      Condo in Seaside Heights
##                                        description   state view from_date
## 1 Rm #3 Lg Private Rm by Rutgers+NYC + Jersey Shore Alabama  146 4/27/2023
## 2  The Stockton - Victorian Ocean Grove near Asbury Alabama  237 4/27/2023
## 3                                     The Sea Loft Alabama  321 5/29/2023
## 4                                     Cute Cottage Alabama  312 5/14/2023
## 5            Unique Spanish Villa (no loud parties) Alabama  270 4/23/2023
## 6 Bayside bungalow just a few blocks from the beach Alabama  128 4/23/2023
##     to_date price original_price                                        price_str
## 1  5/2/2023    55             55                          $55 night$55 per night
## 2  5/3/2023   163            163                         $163 night$163 per night
## 3  6/5/2023   133            148 $148 $133 night$133 per night, originally $148
## 4 5/19/2023   118            155 $155 $118 night$118 per night, originally $155
## 5 4/28/2023   199            199                         $199 night$199 per night
## 6 4/28/2023   154            154                         $154 night$154 per night
##   rating no_of_rates
## 1    4.9         664
## 2    4.9         272
## 3    4.9          63
## 4    4.9          91
## 5    4.9         462
## 6    4.9          71
```

1.0 EDA

```
df_airbnb_data_2 <- df_airbnb_data
```

```r
df_airbnb_data_2$binned_season <- cut(month(as.Date(df_airbnb_data$to_date, "%m/%d/%y")) + 1, seq(1, 13
head(df_airbnb_data_2)
```

```
##                      place_name
## 1 Private room in New Brunswick
## 2      Home in Neptune Township
## 3     Private room in Manasquan
## 4         Cottage in Lavallette
## 5    Guesthouse in Seaside Park
## 6      Condo in Seaside Heights
##                                       description   state view from_date
## 1 Rm #3 Lg Private Rm by Rutgers+NYC + Jersey Shore Alabama  146 4/27/2023
## 2  The Stockton - Victorian Ocean Grove near Asbury Alabama  237 4/27/2023
## 3                                    The Sea Loft Alabama  321 5/29/2023
## 4                                    Cute Cottage Alabama  312 5/14/2023
## 5            Unique Spanish Villa (no loud parties) Alabama  270 4/23/2023
## 6 Bayside bungalow just a few blocks from the beach Alabama  128 4/23/2023
##     to_date price original_price                                      price_str
## 1  5/2/2023    55             55                        $55 night$55 per night
## 2  5/3/2023   163            163                       $163 night$163 per night
## 3  6/5/2023   133            148 $148 $133 night$133 per night, originally $148
## 4 5/19/2023   118            155 $155 $118 night$118 per night, originally $155
## 5 4/28/2023   199            199                       $199 night$199 per night
## 6 4/28/2023   154            154                       $154 night$154 per night
##   rating no_of_rates binned_season
## 1    4.9         664        Spring
## 2    4.9         272        Spring
## 3    4.9          63        Spring
## 4    4.9          91        Spring
## 5    4.9         462        Spring
## 6    4.9          71        Spring
```

```r
df_airbnb_data_2 %>% filter(state == 'Alaska') %>% summarise(num_listings = n(),
              lowest_view = min(view),
              highest_view = max(view),
              view_range = highest_view - lowest_view)
```

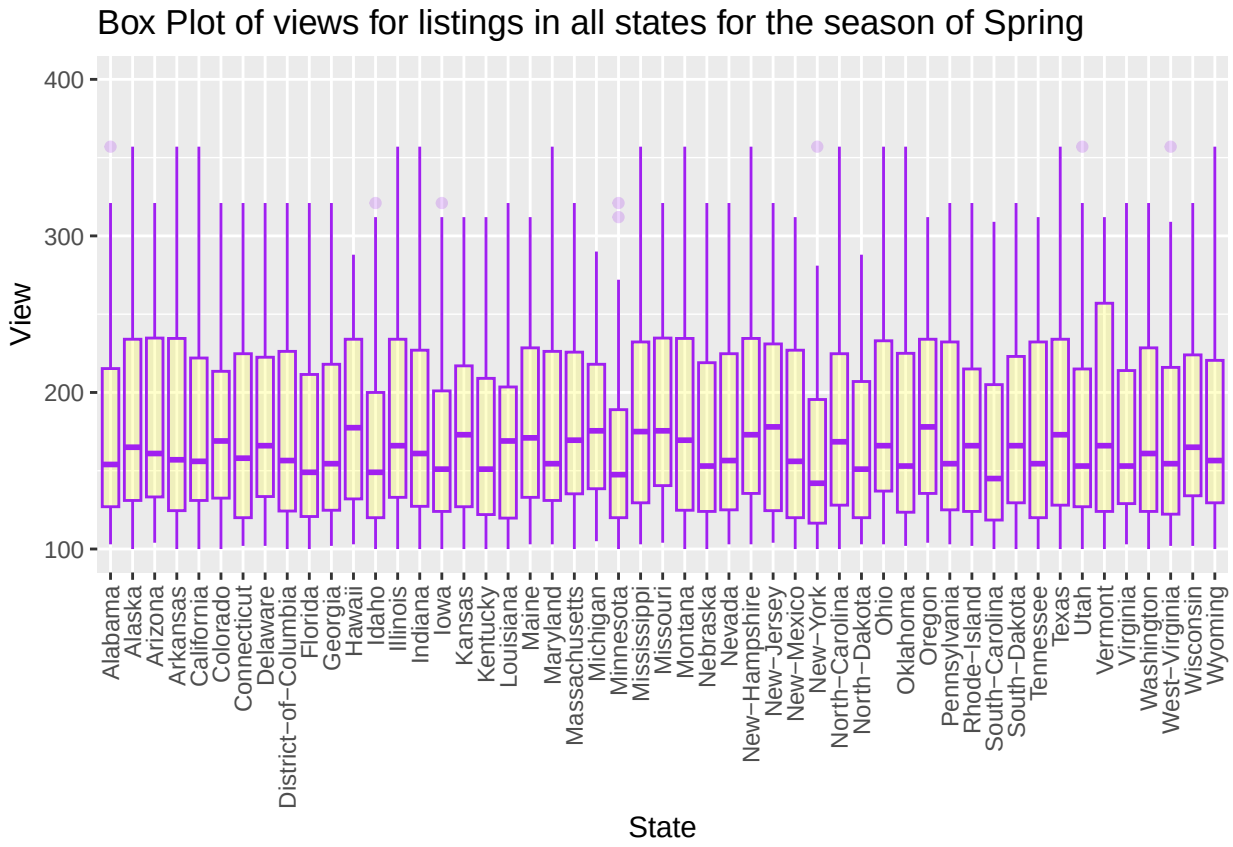
```
##   num_listings lowest_view highest_view view_range
## 1           63         100          701        601
```

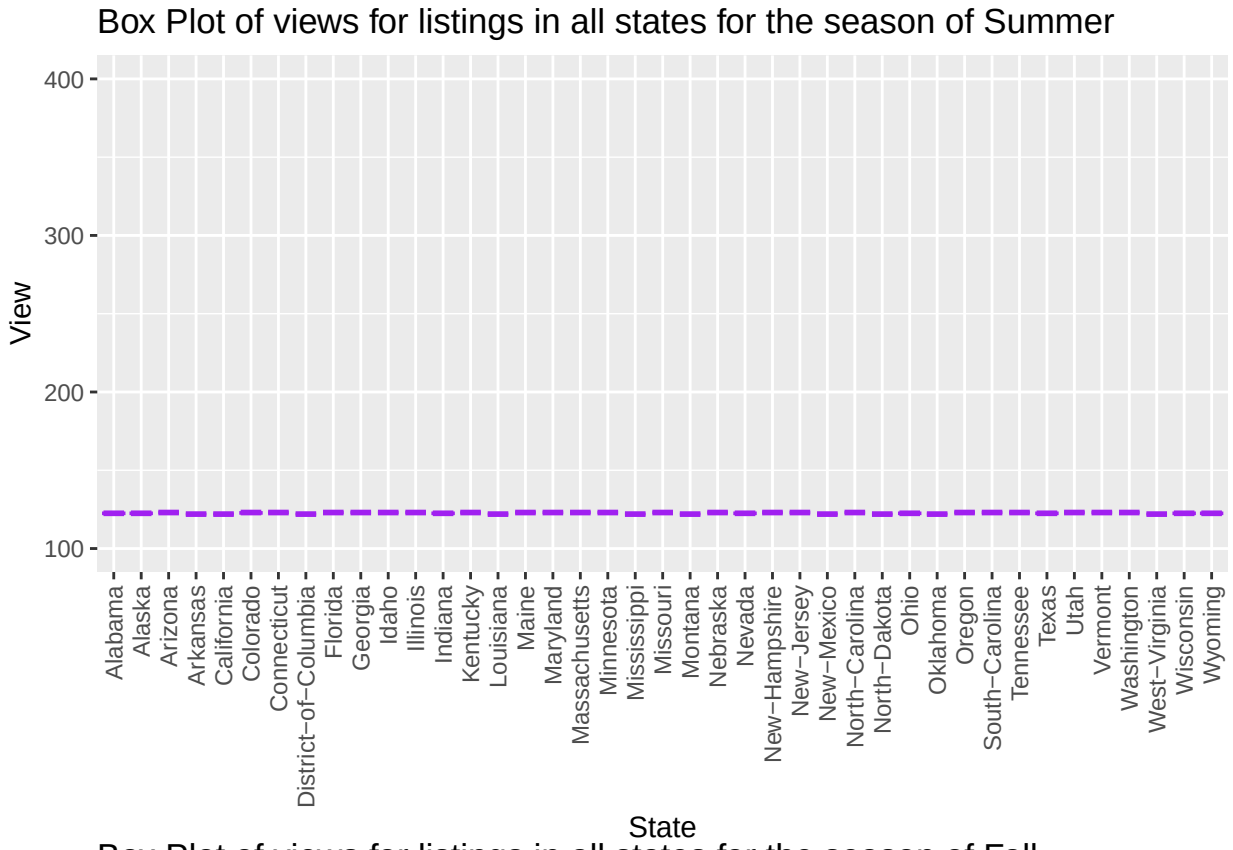
```r
df_airbnb_data_2 %>% filter(state == 'Minnesota') %>% summarise(num_listings = n(),
              lowest_view = min(view),
              highest_view = max(view),
              view_range = highest_view - lowest_view)
```

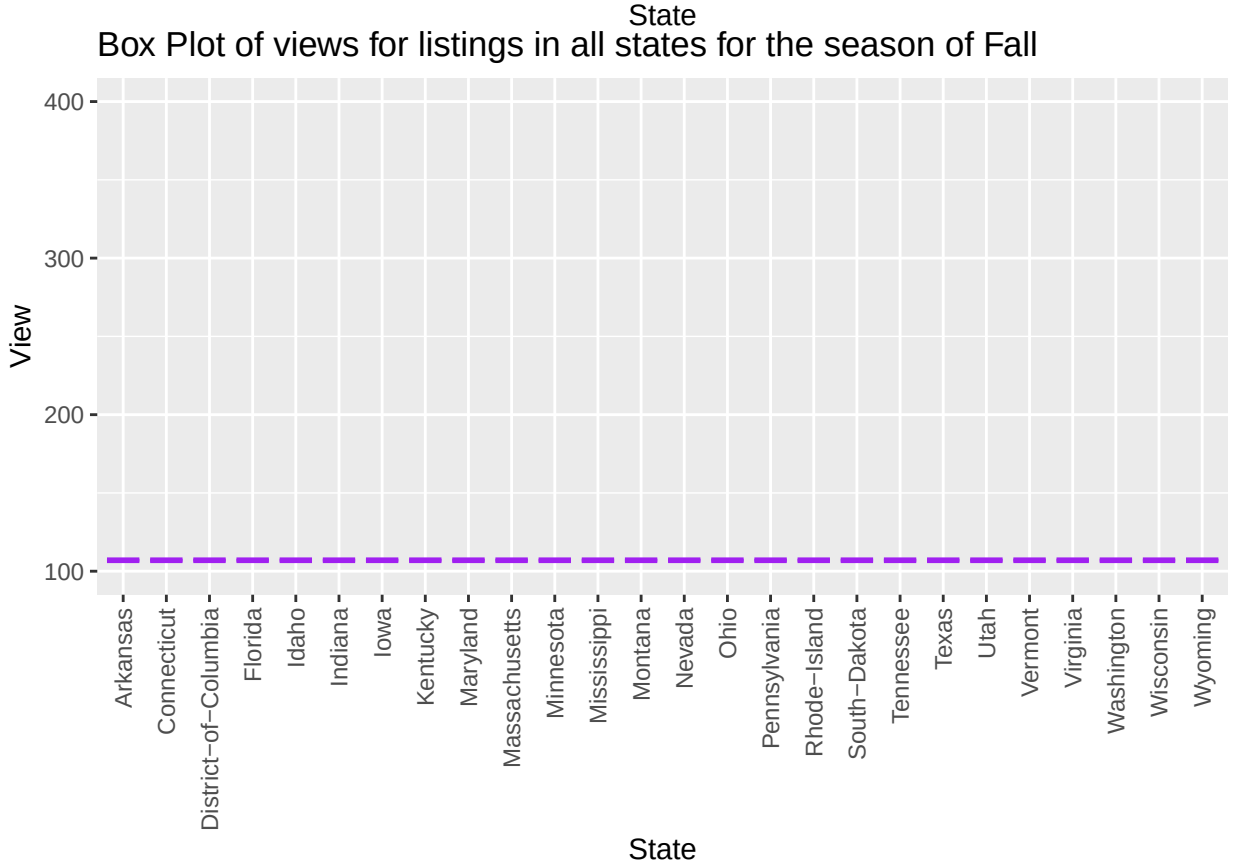
```
##   num_listings lowest_view highest_view view_range
## 1           57         100          701        601
```

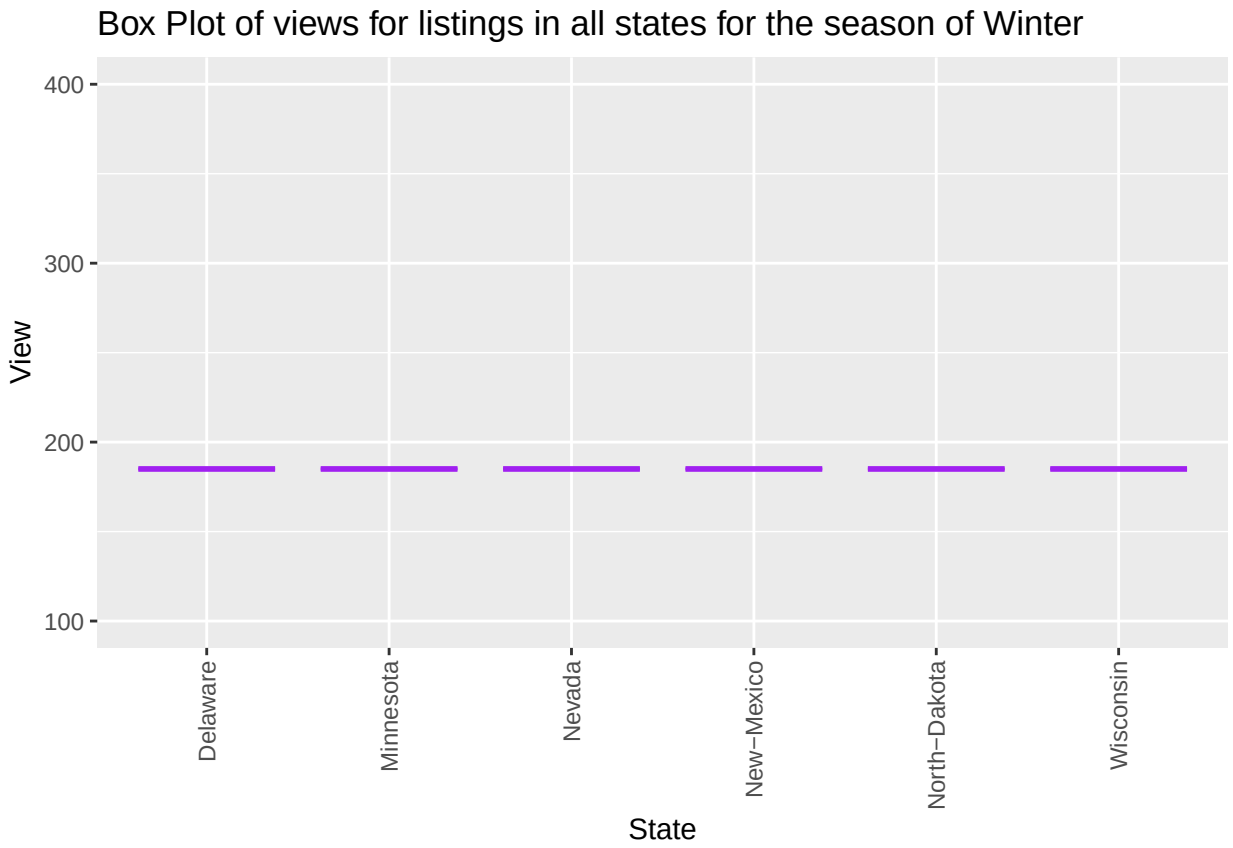
```r
for (season in c("Spring", "Summer", "Fall", "Winter")) {
  df_airbnb_data_2 %>%
    filter(df_airbnb_data_2$binned_season == season) %>%
    ggplot(., aes(x = state, y = view)) +
      geom_boxplot(color="purple", fill="yellow", alpha=0.2) +
      coord_cartesian(ylim=c(100, 400)) +
      xlab("State") +
      ylab("View") +
```

```
      ggtitle(paste0("Box Plot of views for listings in all states for the season of ", season)) +
      theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) -> g
    print(g)
}
```



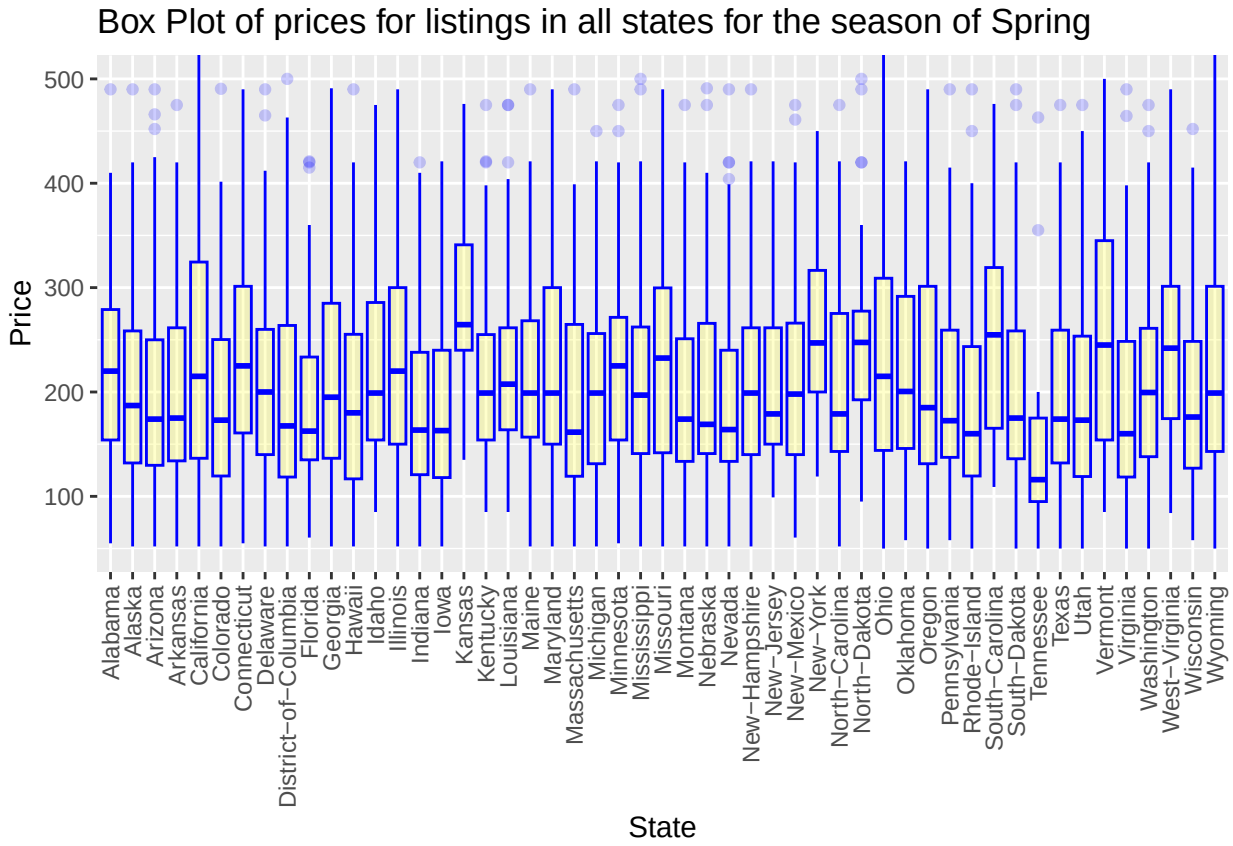

Box Plot of views for listings in all states for the season of Summer
View
400
300
200
100
Alabama
Alaska
Arizona
Arkansas
California
Colorado
Connecticut
District−of−Columbia
Florida
Georgia
Idaho
Illinois
Indiana
Kentucky
Louisiana
Maine
Maryland
Massachusetts
Minnesota
Mississippi
Missouri
Montana
Nebraska
Nevada
New−Hampshire
New−Jersey
New−Mexico
North−Carolina
North−Dakota
Ohio
Oklahoma
Oregon
South−Carolina
Tennessee
Texas
Utah
Vermont
Washington
West−Virginia
Wisconsin
Wyoming
State


Box Plot of views for listings in all states for the season of Fall
View
400
300
200
100
Arkansas
Connecticut
District−of−Columbia
Florida
Idaho
Indiana
Iowa
Kentucky
Maryland
Massachusetts
Minnesota
Mississippi
Montana
Nevada
Ohio
Pennsylvania
Rhode−Island
South−Dakota
Tennessee
Texas
Utah
Vermont
Virginia
Washington
Wisconsin
Wyoming
State

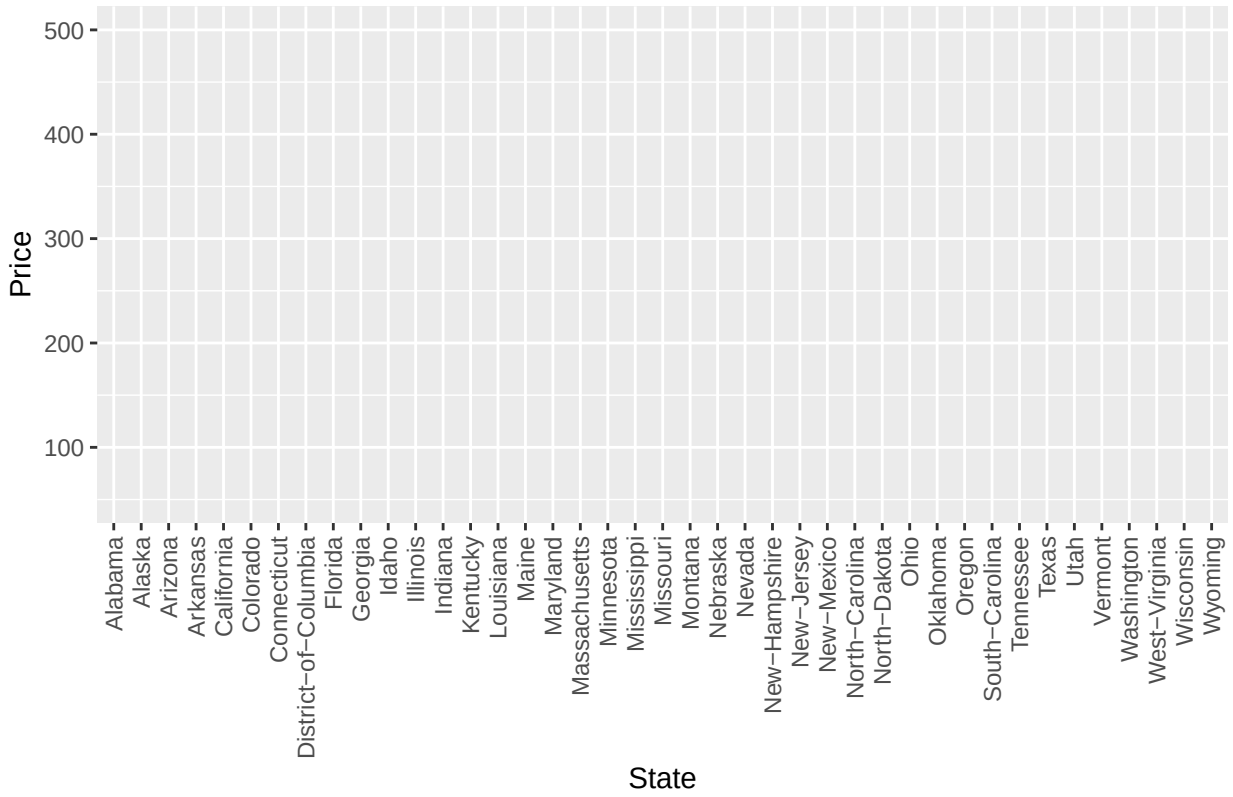



```
for (season in c("Spring", "Summer", "Fall", "Winter")) {
  df_airbnb_data_2 %>%
    filter(df_airbnb_data_2$binned_season == season) %>%
    ggplot(., aes(x = state, y = price)) +
      geom_boxplot(color="blue", fill="yellow", alpha=0.2) +
      coord_cartesian(ylim=c(50, 500)) +
      xlab("State") +
      ylab("Price") +
      ggtitle(paste0("Box Plot of prices for listings in all states for the season of ", season)) +
      theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) -> g
    print(g)
}
```

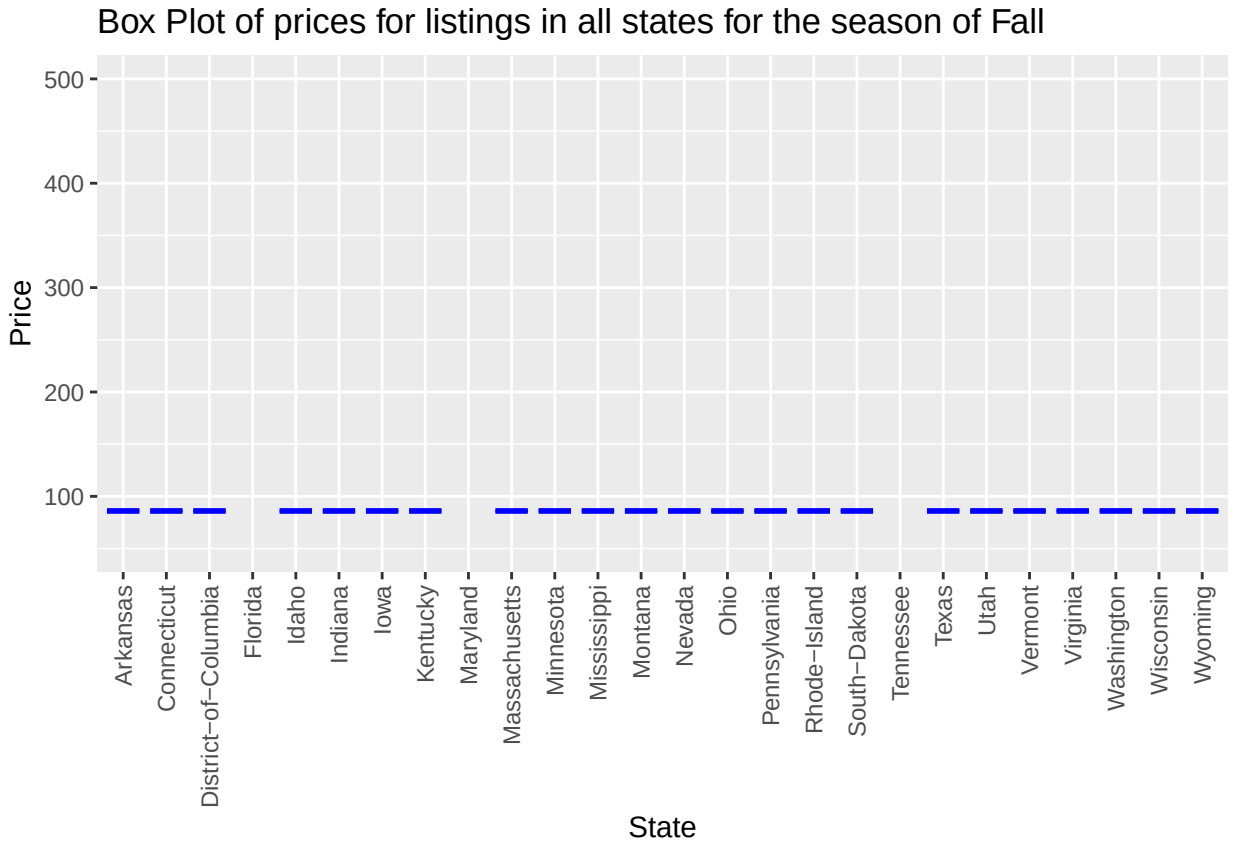
```
## Warning: Removed 177 rows containing non-finite values (`stat_boxplot()`).
```




```
## Warning: Removed 25 rows containing non-finite values (`stat_boxplot()`).
```

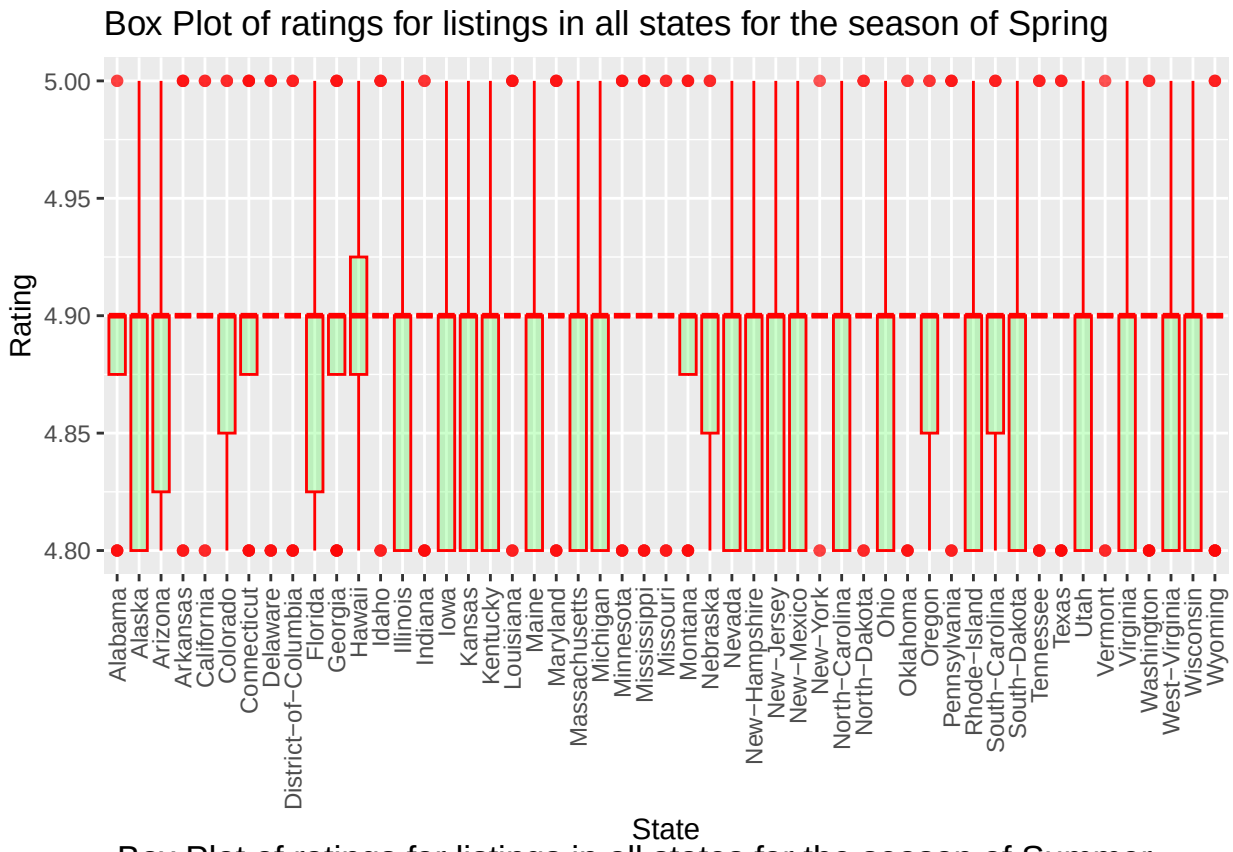


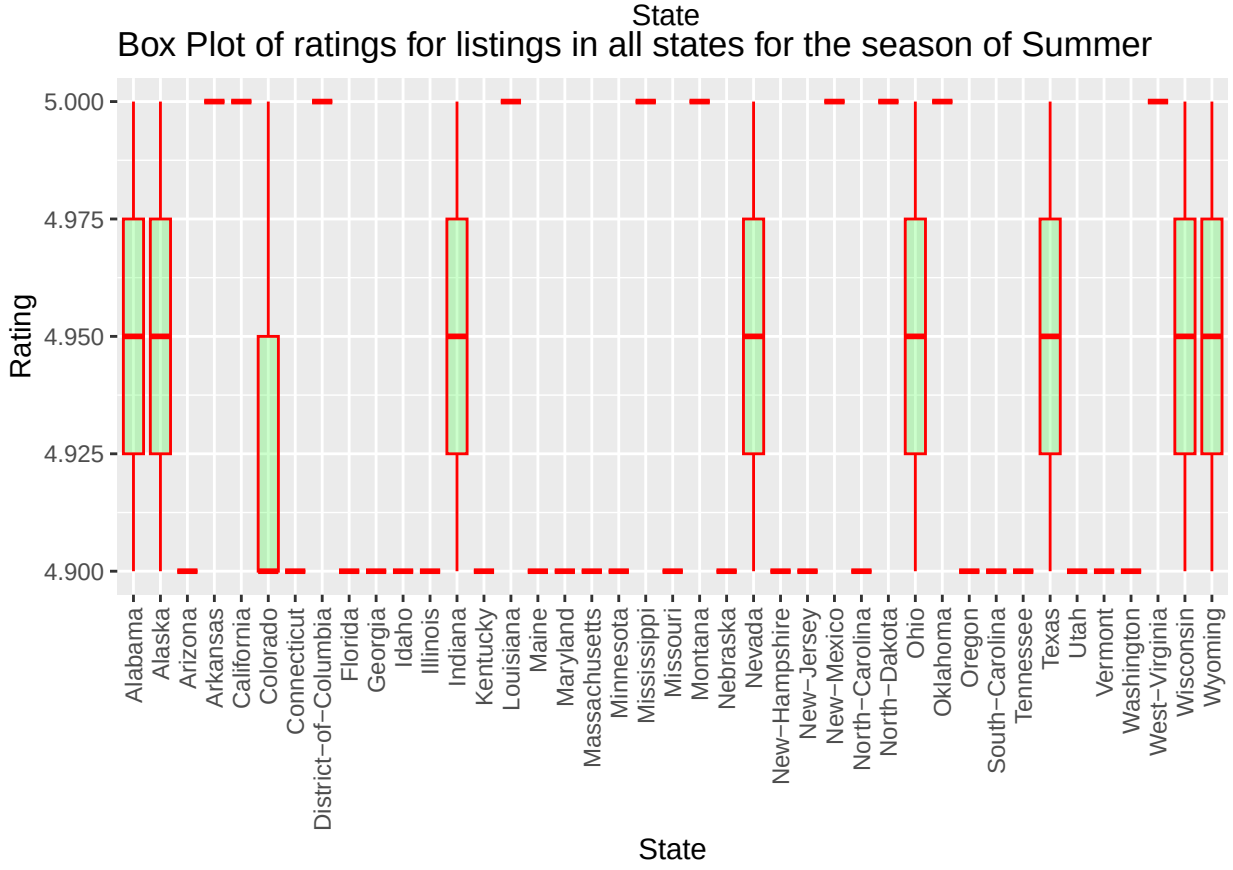

```
## Warning: Removed 4 rows containing non-finite values (`stat_boxplot()`).
```

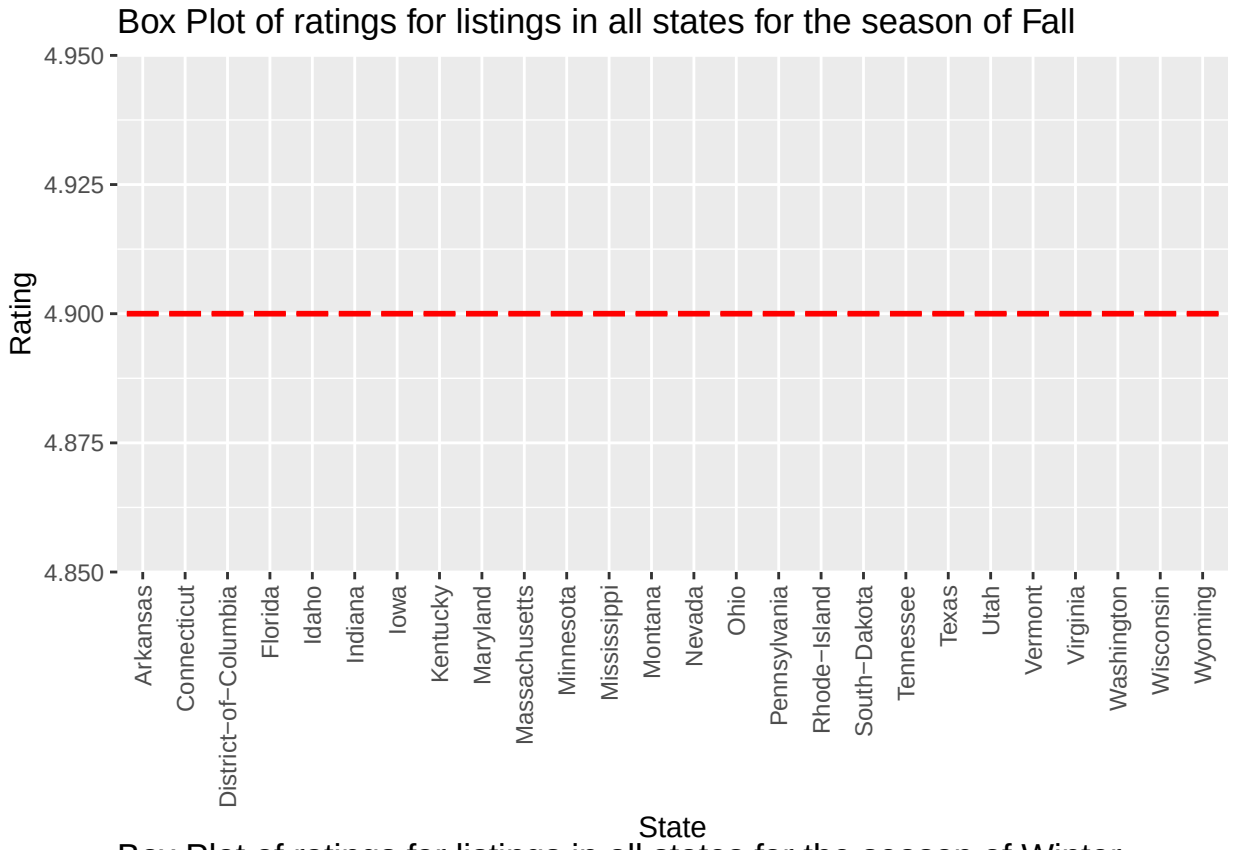


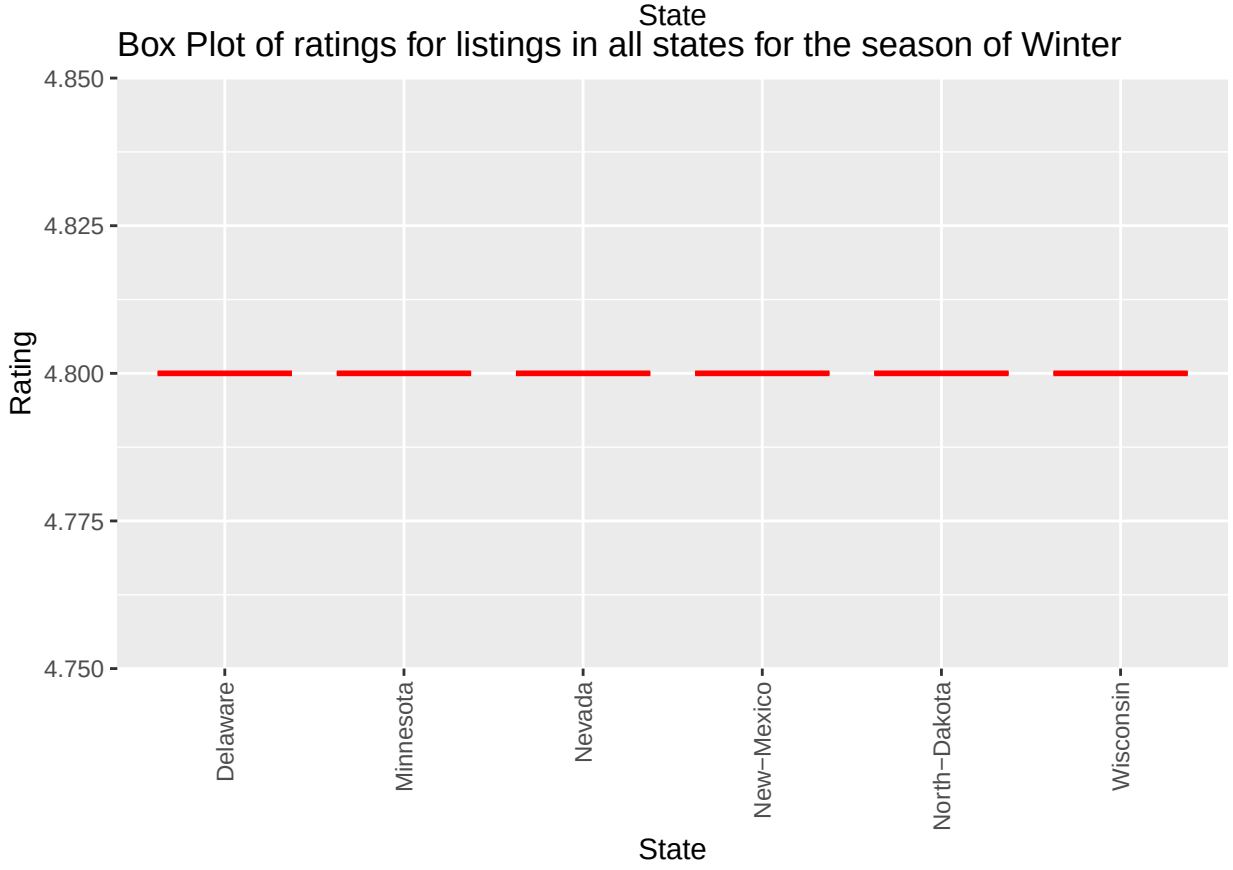

```
## Warning: Removed 1 rows containing non-finite values (`stat_boxplot()`).
```


```
for (season in c("Spring", "Summer", "Fall", "Winter")) {
  df_airbnb_data_2 %>%
    filter(df_airbnb_data_2$binned_season == season) %>%
    ggplot(., aes(x = state, y = rating)) +
      geom_boxplot(color="red", fill="green", alpha=0.2) +
      xlab("State") +
      ylab("Rating") +
      ggtitle(paste0("Box Plot of ratings for listings in all states for the season of ", season)) +
      theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) -> g
    print(g)
}
```


Box Plot of ratings for listings in all states for the season of Spring
Rating
5.00
4.95
4.90
4.85
4.80
Alabama
Alaska
Arizona
Arkansas
California
Colorado
Connecticut
Delaware
District−of−Columbia
Florida
Georgia
Hawaii
Idaho
Illinois
Indiana
Iowa
Kansas
Kentucky
Louisiana
Maine
Maryland
Massachusetts
Michigan
Minnesota
Mississippi
Missouri
Montana
Nebraska
Nevada
New−Hampshire
New−Jersey
New−Mexico
New−York
North−Carolina
North−Dakota
Ohio
Oklahoma
Oregon
Pennsylvania
Rhode−Island
South−Carolina
South−Dakota
Tennessee
Texas
Utah
Vermont
Virginia
Washington
West−Virginia
Wisconsin
Wyoming
State

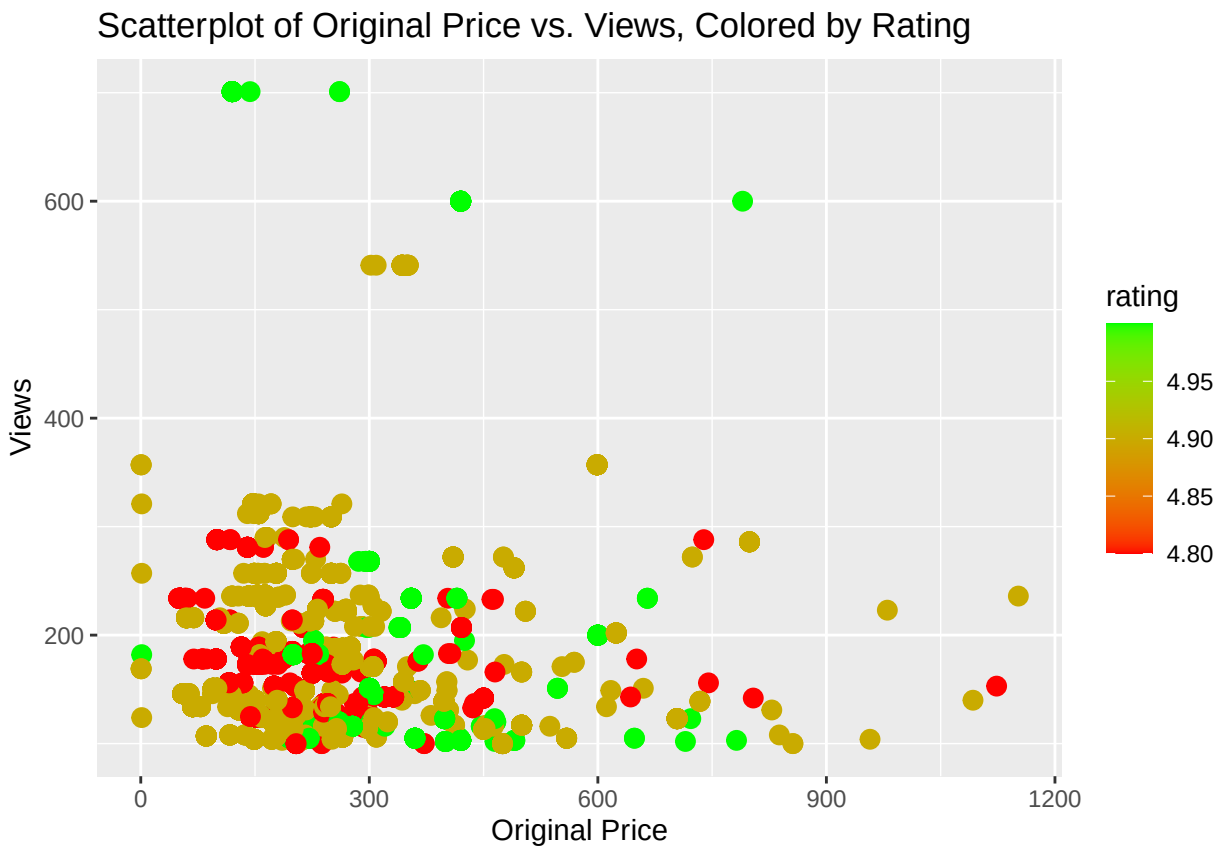

Box Plot of ratings for listings in all states for the season of Summer
Rating
5.000
4.975
4.950
4.925
4.900
Alabama
Alaska
Arizona
Arkansas
California
Colorado
Connecticut
District−of−Columbia
Florida
Georgia
Idaho
Illinois
Indiana
Kentucky
Louisiana
Maine
Maryland
Massachusetts
Minnesota
Mississippi
Missouri
Montana
Nebraska
Nevada
New−Hampshire
New−Jersey
New−Mexico
North−Carolina
North−Dakota
Ohio
Oklahoma
Oregon
South−Carolina
Tennessee
Texas
Utah
Vermont
Washington
West−Virginia
Wisconsin
Wyoming
State


Box Plot of ratings for listings in all states for the season of Fall
Rating
4.950
4.925
4.900
4.875
4.850
Arkansas
Connecticut
District−of−Columbia
Florida
Idaho
Indiana
Iowa
Kentucky
Maryland
Massachusetts
Minnesota
Mississippi
Montana
Nevada
Ohio
Pennsylvania
Rhode−Island
South−Dakota
Tennessee
Texas
Utah
Vermont
Virginia
Washington
Wisconsin
Wyoming
State


Box Plot of ratings for listings in all states for the season of Winter
Rating
4.850
4.825
4.800
4.775
4.750
Delaware
Minnesota
Nevada
New−Mexico
North−Dakota
Wisconsin
State

```
ggplot(df_airbnb_data_2, aes(x = original_price, y = view, color = rating)) +
  geom_point(size = 3) +
  scale_color_gradient(low = "red", high = "green") +
  xlab("Original Price") +
  ylab("Views") +
  ggtitle("Scatterplot of Original Price vs. Views, Colored by Rating")
```

```
## Warning: Removed 212 rows containing missing values (`geom_point()`).
```

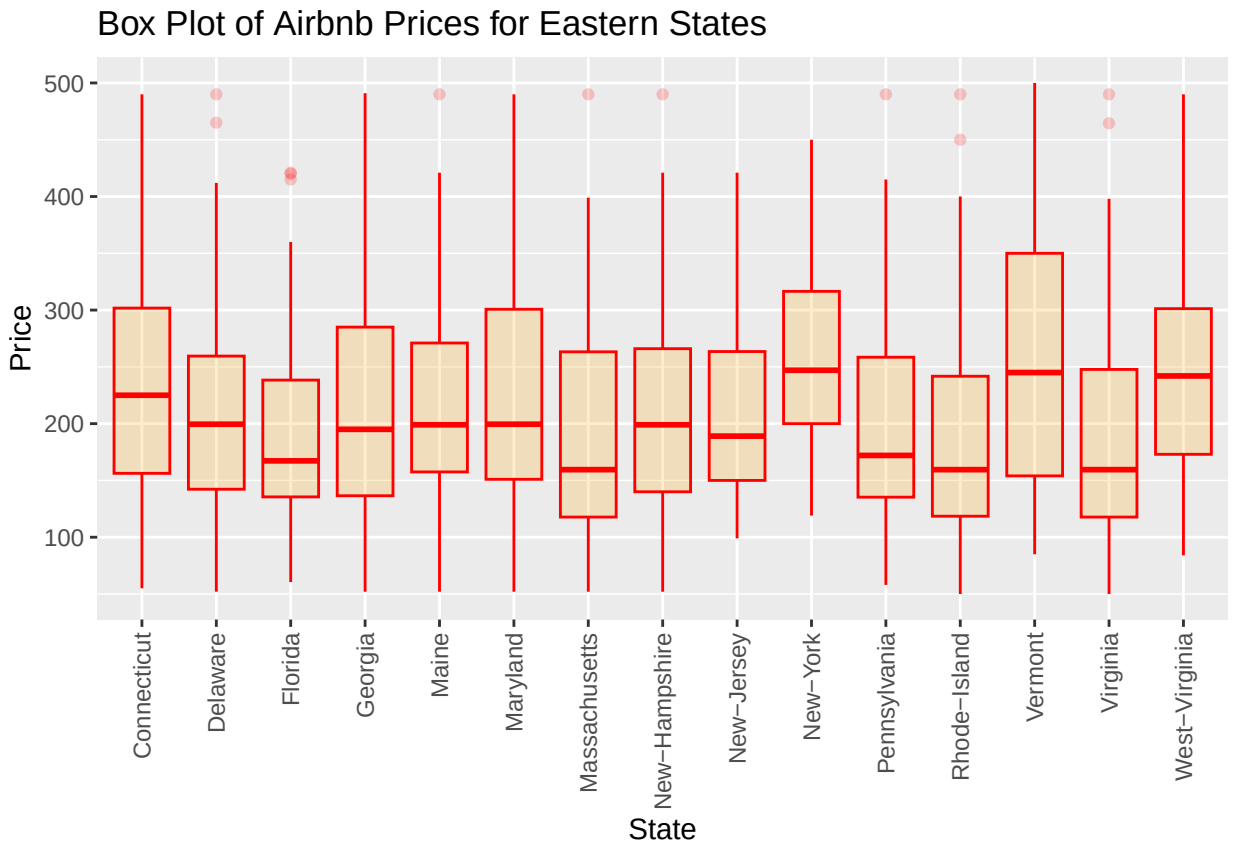



```
eastern_states <- c("Maine", "New-Hampshire", "Vermont", "Massachusetts", "Rhode-Island", "Connecticut"

df_eastern <- df_airbnb_data_2 %>%
  filter(state %in% eastern_states)

ggplot(df_eastern, aes(x = state, y = price)) +
  geom_boxplot(color="red", fill="orange", alpha=0.2) +
  coord_cartesian(ylim=c(50, 500)) +
  xlab("State") +
  ylab("Price") +
  ggtitle("Box Plot of Airbnb Prices for Eastern States") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```

```
## Warning: Removed 45 rows containing non-finite values (`stat_boxplot()`).
```

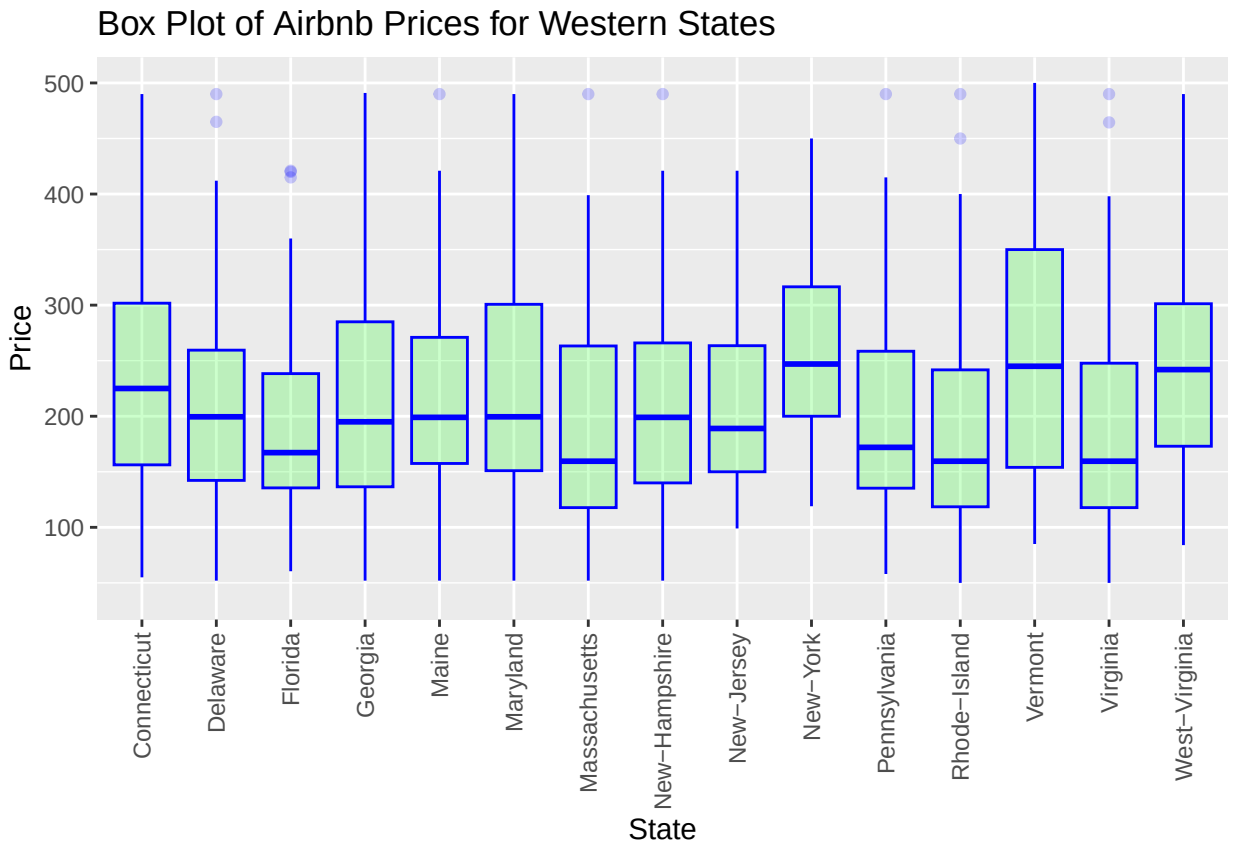



```
western_states <- c("Alaska", "Arizona", "California", "Colorado", "Hawaii", "Idaho",
                    "Montana", "Nevada", "New Mexico", "Oregon", "Utah", "Washington", "Wyoming")

df_western <- df_airbnb_data_2 %>%
  filter(state %in% western_states)

ggplot(df_eastern, aes(x = state, y = price)) +
  geom_boxplot(color="blue", fill="green", alpha=0.2) +
  coord_cartesian(ylim=c(40, 500)) +
  xlab("State") +
  ylab("Price") +
  ggtitle("Box Plot of Airbnb Prices for Western States") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```

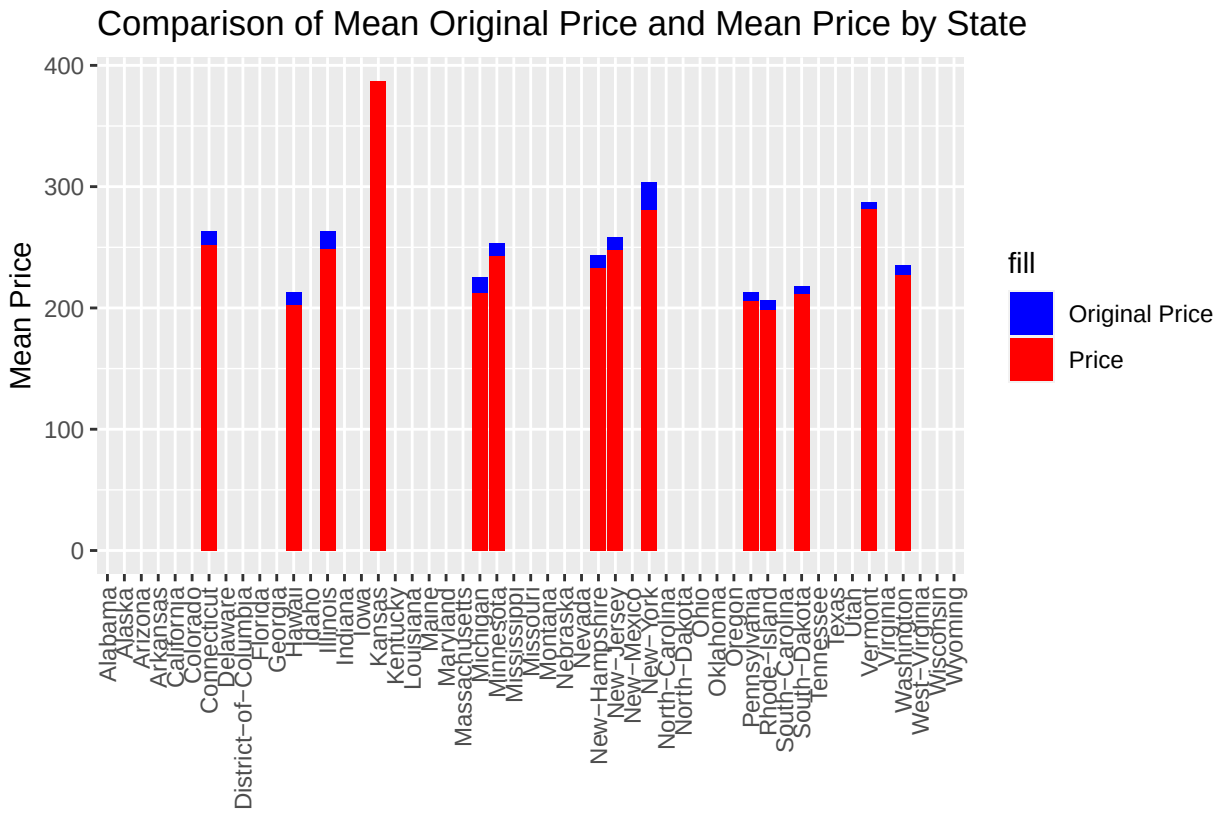
```
## Warning: Removed 45 rows containing non-finite values (`stat_boxplot()`).
```



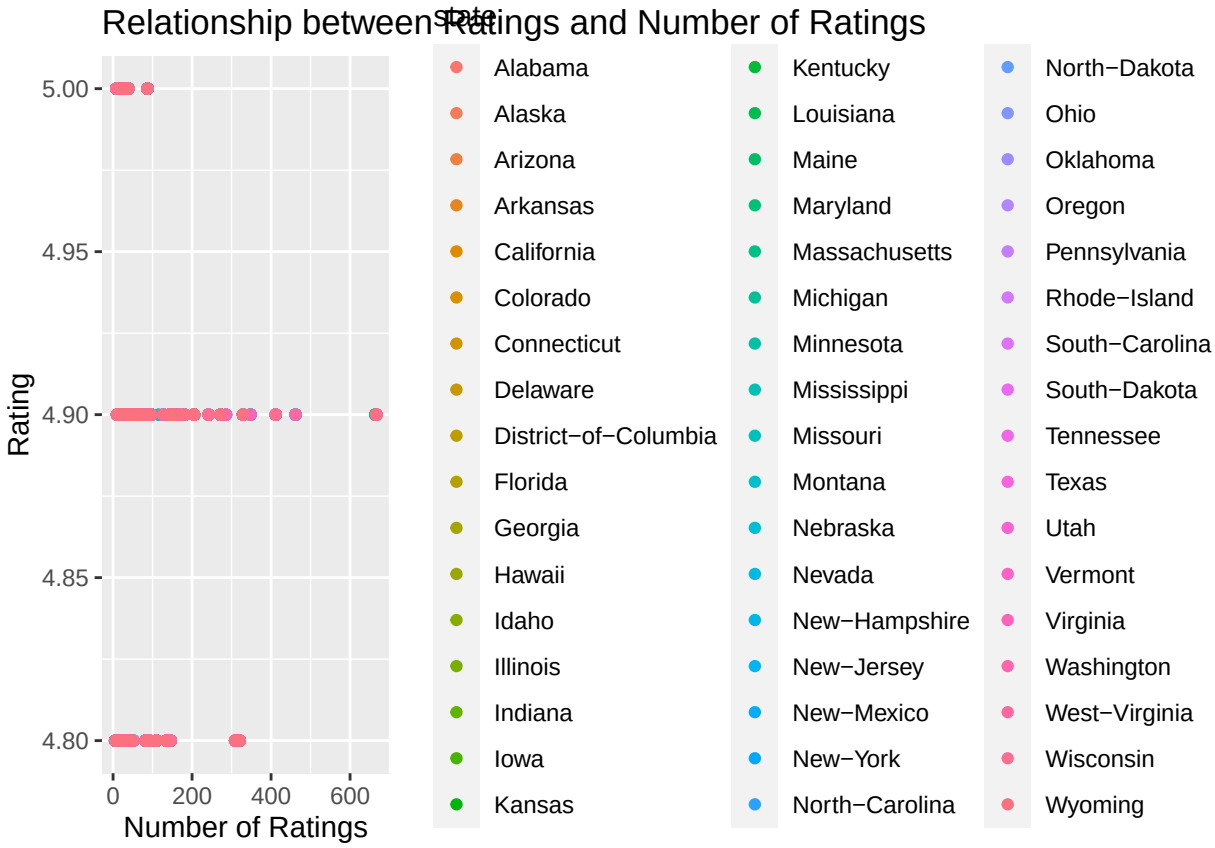

```
df_prices <- df_airbnb_data_2 %>%
  group_by(state) %>%
  summarise(mean_original_price = mean(original_price),
            mean_price = mean(price)) %>%
  ungroup()
```

```
ggplot(df_prices, aes(x = state)) +
  geom_bar(aes(y = mean_original_price, fill = "Original Price"), stat = "identity", position = "dodge"
  geom_bar(aes(y = mean_price, fill = "Price"), stat = "identity", position = "dodge") +
  scale_fill_manual(values = c("Original Price" = "blue", "Price" = "red")) +
  xlab("") +
  ylab("Mean Price") +
  ggtitle("Comparison of Mean Original Price and Mean Price by State") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```

```
## Warning: Removed 37 rows containing missing values (`geom_bar()`).
## Removed 37 rows containing missing values (`geom_bar()`).
```




```
ggplot(df_airbnb_data_2, aes(x = no_of_rates, y = rating, color = state)) +
  geom_point() +
  xlab("Number of Ratings") +
  ylab("Rating") +
  ggtitle("Relationship between Ratings and Number of Ratings")
```



2.0 Sentiment Analysis on Description

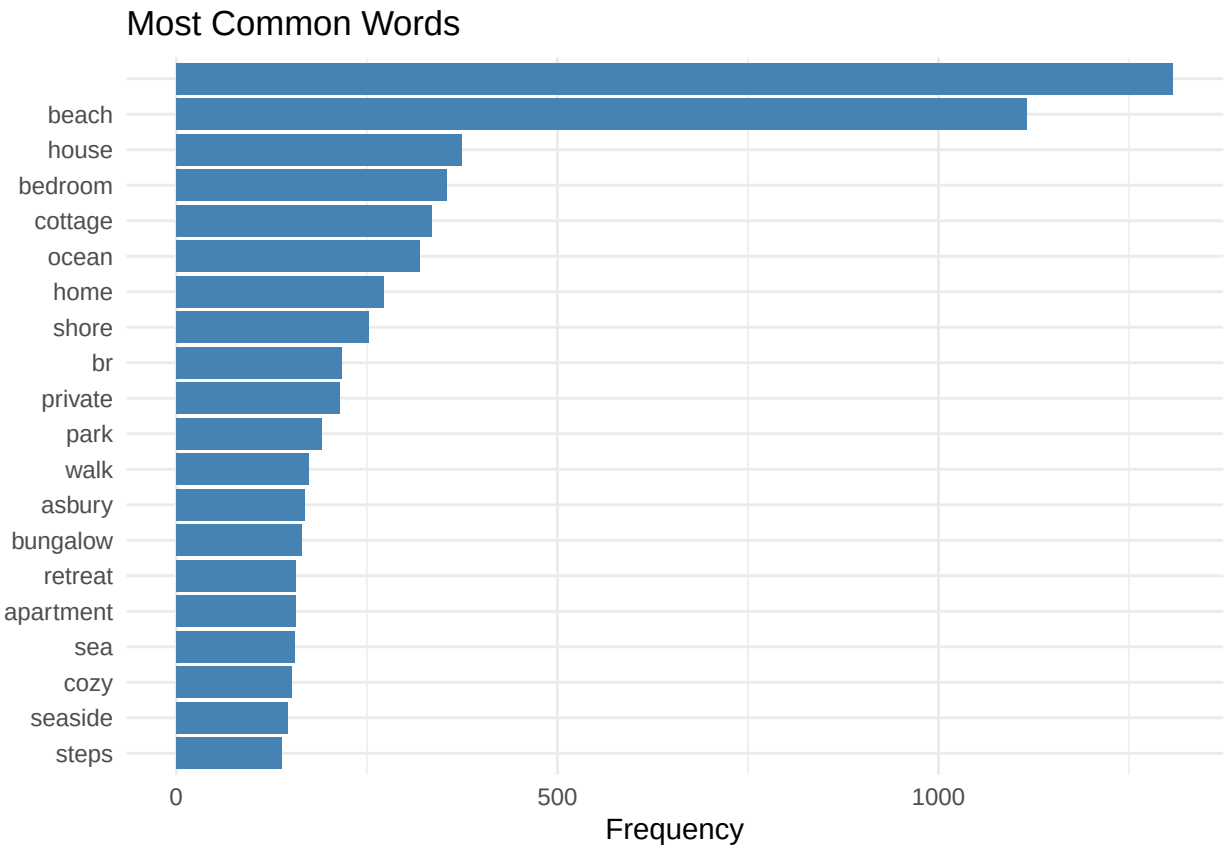
```
tidy_desc <- df_airbnb_data %>%
  select(description) %>%
  unnest_tokens(word, description) %>%
  anti_join(stop_words) %>%
  mutate(word = str_replace_all(word, "[^[:alpha:]\\s']", ""))
```

```
## Joining with `by = join_by(word)`
```

```
# Calculate the frequency of each word and select the top 20
top_words <- tidy_desc %>%
  count(word, sort = TRUE) %>%
  top_n(20)
```

```
## Selecting by n
```

```
ggplot(top_words, aes(x = reorder(word, n), y = n)) +
  geom_col(fill = "steelblue") +
  labs(x = NULL, y = "Frequency") +
  coord_flip() +
  ggtitle("Most Common Words") +
  theme_minimal()
```




```
bigrams <- df_airbnb_data %>%
  unnest_tokens(bigram, description, token = "ngrams", n = 2)
```

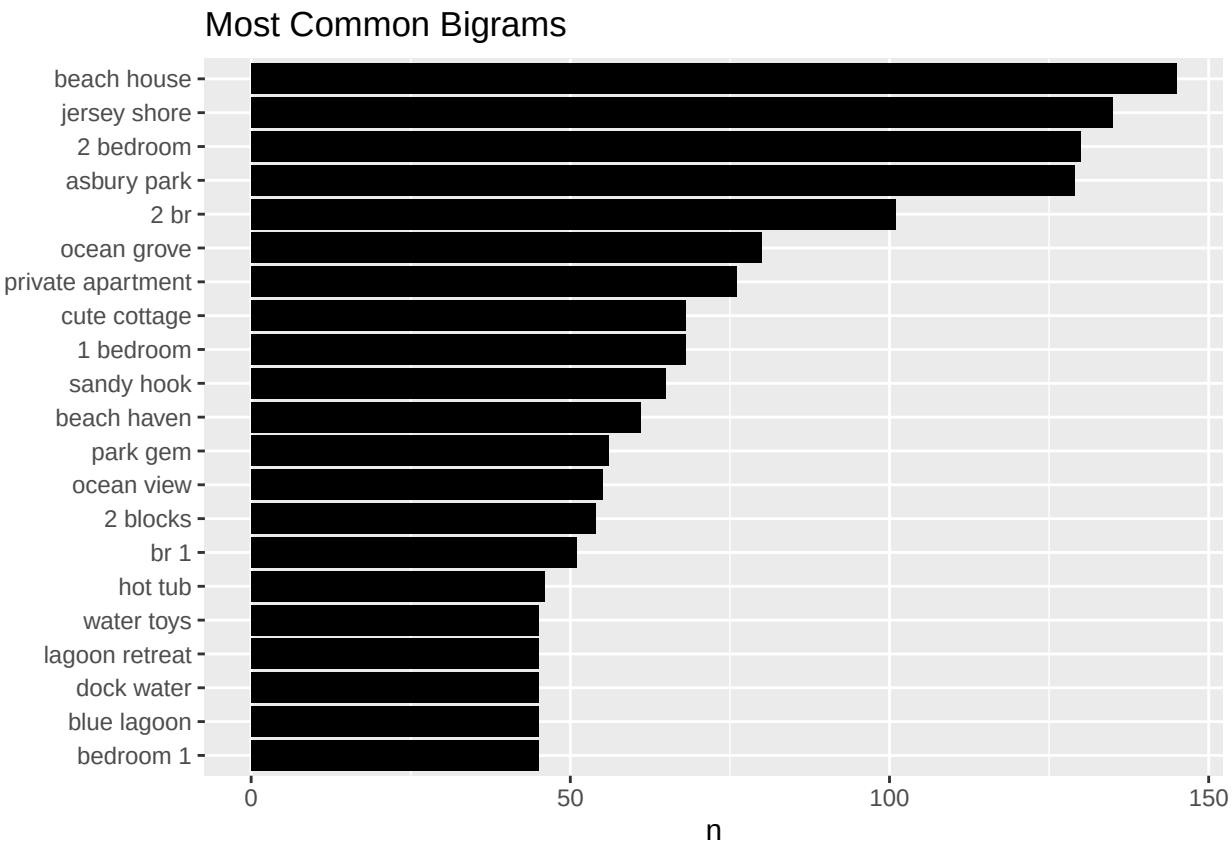
```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v tibble  3.1.8     v forcats 0.5.2
## v readr   2.1.3
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x NLP::annotate()          masks ggplot2::annotate()
## x lubridate::as.difftime() masks base::as.difftime()
## x modelr::bootstrap()      masks broom::bootstrap()
## x dplyr::combine()         masks randomForest::combine()
## x tidyr::complete()        masks RCurl::complete()
## x NLP::content()           masks httr::content()
## x lubridate::date()        masks base::date()
## x magrittr::extract()      masks tidyr::extract()
## x dplyr::filter()          masks stats::filter()
## x purrr::flatten()         masks jsonlite::flatten()
## x readr::guess_encoding()  masks rvest::guess_encoding()
## x httr::handle_reset()     masks curl::handle_reset()
## x lubridate::intersect()   masks base::intersect()
## x dplyr::lag()             masks stats::lag()
## x ggplot2::margin()        masks randomForest::margin()
## x readr::parse_date()      masks curl::parse_date()
## x magrittr::set_names()    masks purrr::set_names()
## x lubridate::setdiff()     masks base::setdiff()
```

```
## x lubridate::union()       masks base::union()
```

```
bigrams %>%
  separate(bigram, c("word1", "word2"), sep = " ") %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word) %>%
  unite(bigram, word1, word2, sep=" ") %>%
  count(bigram, sort = TRUE) %>%
  filter(n > 20 ) %>%
  top_n(20) %>%
  mutate(bigram = reorder(bigram, n)) %>%
  ggplot(aes(bigram, n)) +
  geom_bar(stat="identity") +
  geom_col(fill = "black") +
  ggtitle("Most Common Bigrams") +
  xlab(NULL) + coord_flip()
```

```
## Selecting by n
```




```
df_states <- df_airbnb_data %>%
  filter(state %in% c("New-York", "California", "Florida", "Hawaii"))
head(df_states)
```

```
##                         place_name
## 1 Private room in Seaside Heights
## 2                  Home in Berlin
## 3       Private room in Beachwood
## 4          Home in Barnegat Light
## 5            Cottage in Lake Como
```

```
## 6 Guesthouse in Lawrence Township
##                                     description      state view from_date
## 1                                  Ortley Beach California  234  5/7/2023
## 2                100% Private Fits 1 to 9 Guests California  178 4/23/2023
## 3   Almost "down by the river side" (green room) California  134 4/28/2023
## 4 2 Night Minimum LBI Barnegat Light NJ 2 BR 1 B California  257 4/23/2023
## 5                   Sea Glass & Lavender Cottage California  236  5/8/2023
## 6         Adorable studio cottage near Princeton California  156 4/29/2023
##     to_date price original_price                                        price_str
## 1 5/13/2023    52             52                          $52 night$52 per night
## 2 4/28/2023    69             99     $99 $69 night$69 per night, originally $99
## 3  5/5/2023    68             68                          $68 night$68 per night
## 4 4/30/2023   140            178 $178 $140 night$140 per night, originally $178
## 5 5/14/2023   103            142 $142 $103 night$103 per night, originally $142
## 6  5/4/2023   116            116                        $116 night$116 per night
##   rating no_of_rates
## 1    4.8         321
## 2    4.8          41
## 3    4.9         181
## 4    4.9         205
## 5    4.9          70
## 6    4.8          93
```

```
ny_ca_words <- df_states %>%
  select(state, description) %>%
  unnest_tokens(word, description) %>%
  anti_join(stop_words) %>%
  mutate(word = str_replace_all(word, "[^[:alpha:]\\s']", "")) %>%
  filter(!word %in% stop_words$word) %>%
  count(state, word) %>%
  group_by(state) %>%
  mutate(proportion = n / sum(n)) %>%
  select(-n) %>%
  pivot_wider(names_from = "state", values_from = "proportion")
```

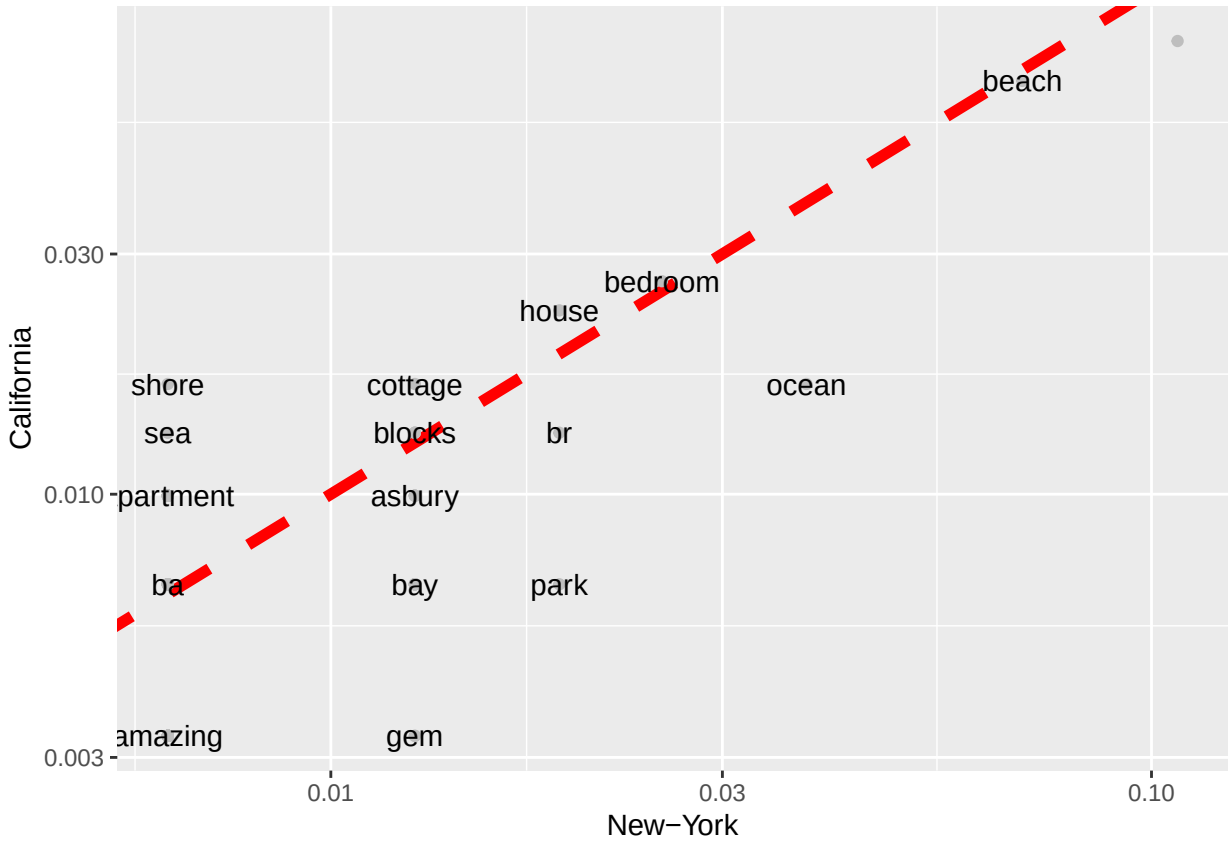
```
## Joining with `by = join_by(word)`
```

```
ggplot(ny_ca_words, aes(x = `New-York`,
y = `California`)) +
geom_abline(color = "red", lty = 2,
lwd=2) +
geom_point(color="grey")+
geom_text(aes(label = word),
check_overlap = TRUE) +
scale_x_log10() +
scale_y_log10()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
```

```
## Warning: Removed 152 rows containing missing values (`geom_point()`).
```

```
## Warning: Removed 152 rows containing missing values (`geom_text()`).
```



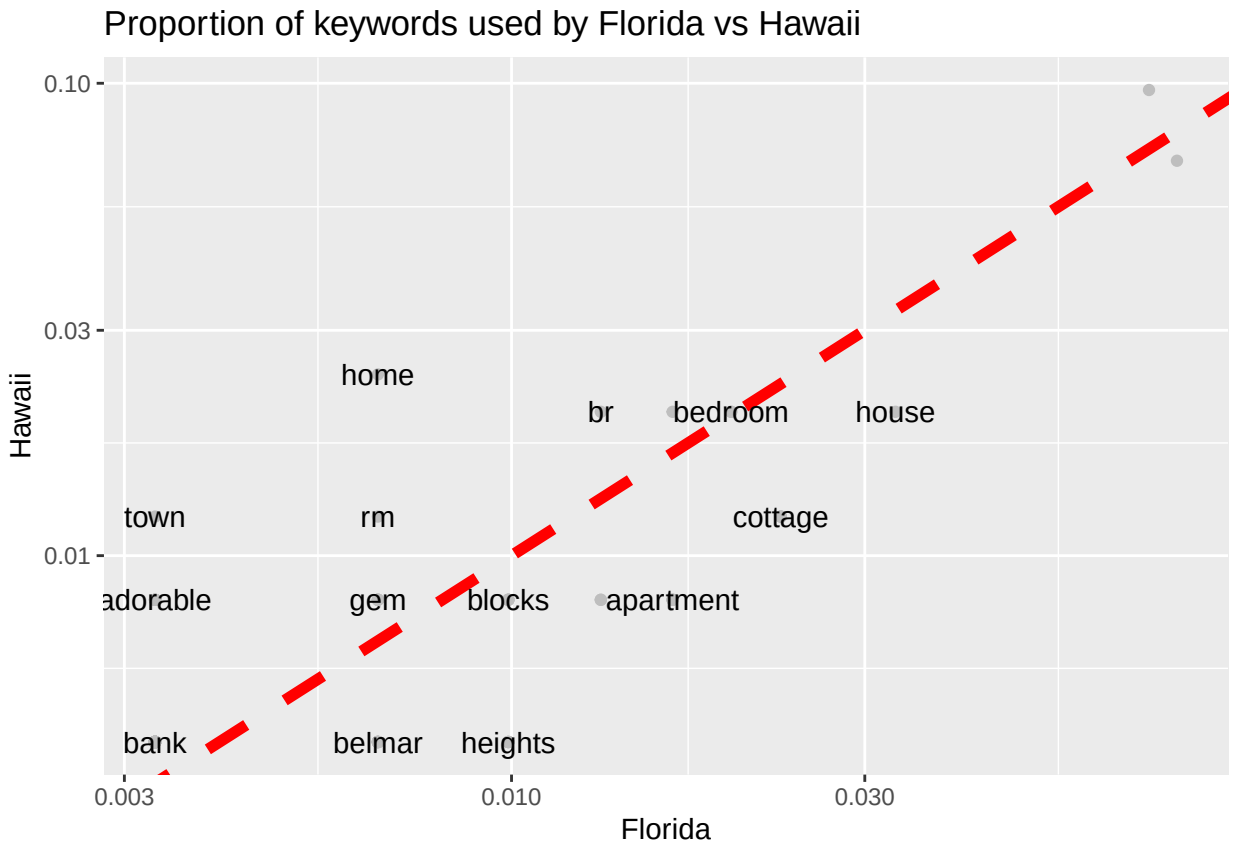

```
fl_hw_words <- df_states %>%
  select(state, description) %>%
  unnest_tokens(word, description) %>%
  anti_join(stop_words) %>%
  mutate(word = str_replace_all(word, "[^[:alpha:]\\s']", "")) %>%
  filter(!word %in% stop_words$word) %>%
  count(state, word) %>%
  group_by(state) %>%
  mutate(proportion = n / sum(n)) %>%
  select(-n) %>%
  pivot_wider(names_from = "state", values_from = "proportion")
```

```
## Joining with `by = join_by(word)`
```

```
ggplot(fl_hw_words, aes(x = `Florida`,
y = `Hawaii`)) +
geom_abline(color = "red", lty = 2,
lwd=2) +
geom_point(color="grey")+
geom_text(aes(label = word),
check_overlap = TRUE) +
ggtitle("Proportion of keywords used by Florida vs Hawaii") +
scale_x_log10() +
scale_y_log10()
```

```
## Warning: Removed 103 rows containing missing values (`geom_point()`).
```

```
## Warning: Removed 103 rows containing missing values (`geom_text()`).
```




```
text_corpus <- Corpus(VectorSource(df_airbnb_data$description))
# Perform text preprocessing
text_corpus <- tm_map(text_corpus, removeNumbers) # Remove numbers
```

```
## Warning in tm_map.SimpleCorpus(text_corpus, removeNumbers): transformation drops
## documents
```

```
text_corpus <- tm_map(text_corpus, content_transformer(tolower)) # Convert to lowercase
```

```
## Warning in tm_map.SimpleCorpus(text_corpus, content_transformer(tolower)):
## transformation drops documents
```

```
text_corpus <- tm_map(text_corpus, removePunctuation) # Remove punctuation marks
```

```
## Warning in tm_map.SimpleCorpus(text_corpus, removePunctuation): transformation
## drops documents
```

```
text_corpus <- tm_map(text_corpus, removeWords, stopwords()) # Remove stop words
```

```
## Warning in tm_map.SimpleCorpus(text_corpus, removeWords, stopwords()):
## transformation drops documents
```

```
text_corpus <- tm_map(text_corpus, stemDocument) # Perform stemming
```

```
## Warning in tm_map.SimpleCorpus(text_corpus, stemDocument): transformation drops
## documents
```

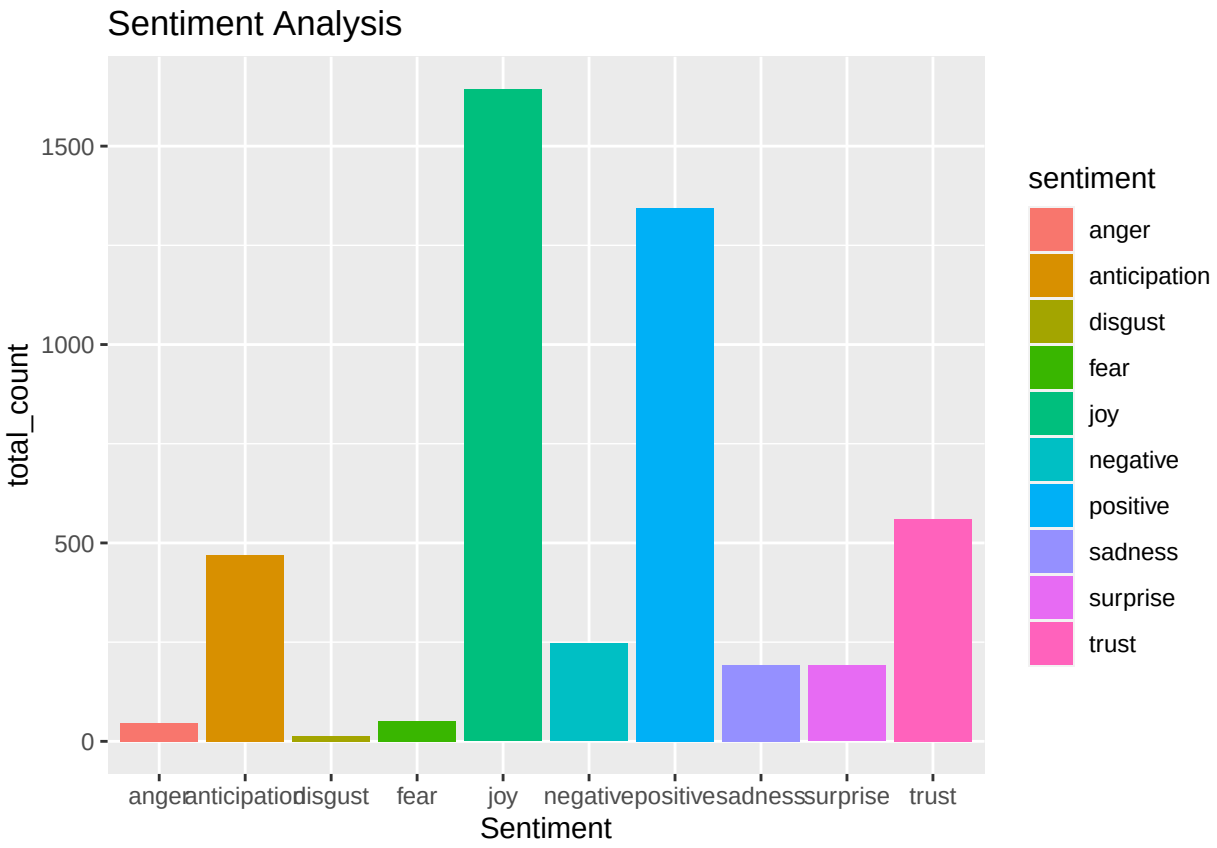
```
text_corpus <- tm_map(text_corpus, stripWhitespace)
```

```
## Warning in tm_map.SimpleCorpus(text_corpus, stripWhitespace): transformation
## drops documents
```

```
dtm <- DocumentTermMatrix(text_corpus)
df_dtm <- as.data.frame(as.matrix(dtm))
```

```
df_dtm %>%
  rownames_to_column(var = "document") %>%
  pivot_longer(cols = -document, names_to = "term", values_to = "count") %>%
  filter(count > 0) %>%
  inner_join(get_sentiments("nrc"), by = c("term" = "word")) %>%
  group_by(sentiment) %>%
  summarise(total_count = sum(count)) %>%
  ggplot(aes(x = sentiment, y = total_count, fill = sentiment)) +
  geom_bar(stat = "identity") +
  labs(title = "Sentiment Analysis",
       x = "Sentiment")
```

```
## Warning in inner_join(., get_sentiments("nrc"), by = c(term = "word")): Each row in `x` is expected t
## i Row 26 of `x` matches multiple rows.
## i If multiple matches are expected, set `multiple = "all"` to silence this
##   warning.
```




```
# Get the sentiment score
df_sentiments <- df_airbnb_data %>%
  select(description, state) %>%
  mutate(id = row_number()) %>%
  unnest_tokens(word, description) %>%
  left_join(get_sentiments("nrc"), by = "word") %>%
  group_by(id, state) %>%
  summarise(sentiment_score = sum(ifelse(sentiment %in% c("positive", "surprise", "joy", "trust", "anti
```

```
                                        ifelse(sentiment %in% c("negative", "sadness", "disgust", "a
  select(-state)
```

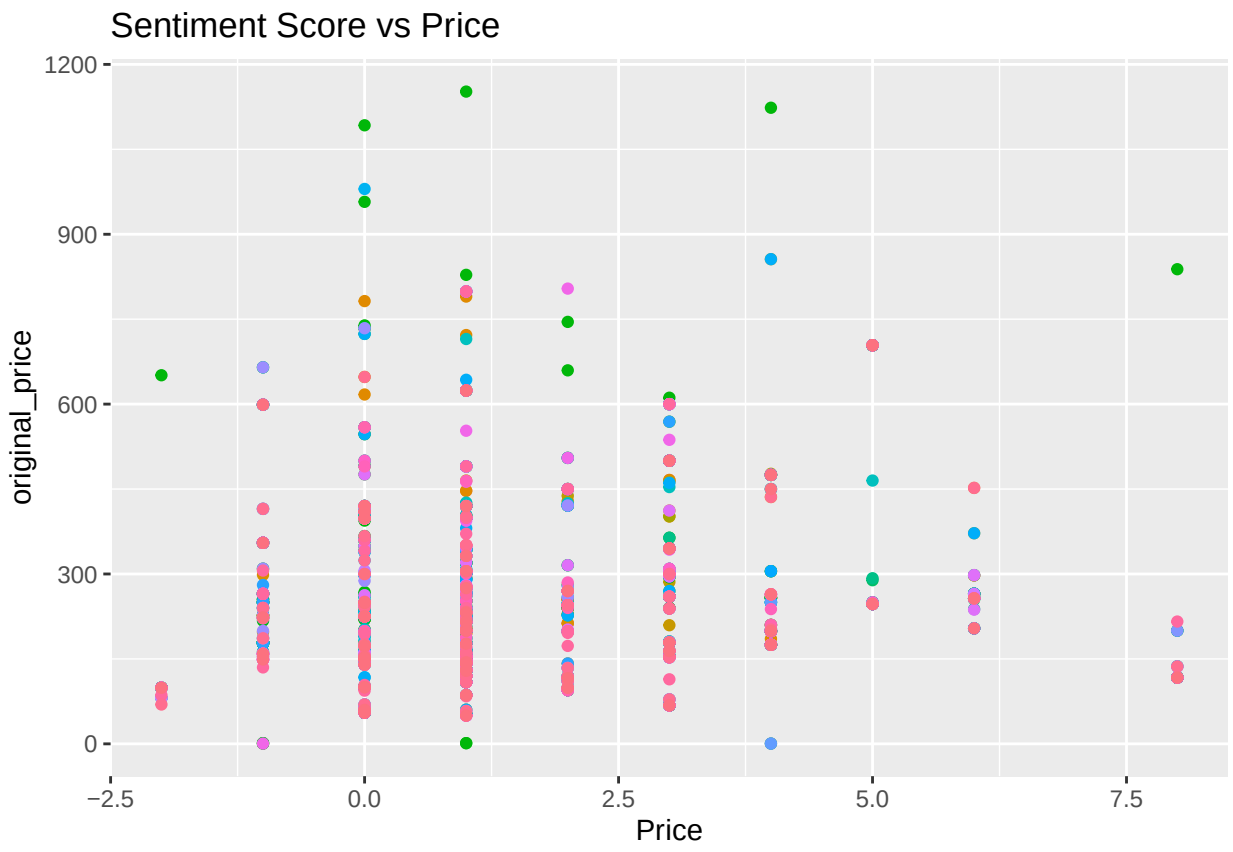
```
## Warning in left_join(., get_sentiments("nrc"), by = "word"): Each row in `x` is expected to match at
## i Row 23 of `x` matches multiple rows.
## i If multiple matches are expected, set `multiple = "all"` to silence this
##   warning.
```

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

```
df_airbnb_data <- df_airbnb_data %>%
  mutate(id = row_number()) %>%
  left_join(df_sentiments, by = "id") %>%
  select(-id)
```

```
df_airbnb_data %>% ggplot(aes(y = original_price, x = sentiment_score, color = state)) +
  geom_point() +
  theme(legend.position = "none") +
  labs(title = "Sentiment Score vs Price",
       x = "Price")
```

```
## Warning: Removed 212 rows containing missing values (`geom_point()`).
```



3.0 Ml Modeling for Predicting Price

```
df_airbnb_data_2 <- df_airbnb_data
df_airbnb_data_2$state_encoded <- as.numeric(factor(df_airbnb_data_2$state))
```

```
df_airbnb_data_2 <- na.omit(df_airbnb_data_2)
```

```
train_index <- sample(1:nrow(df_airbnb_data_2), 0.7*nrow(df_airbnb_data_2))
train_data <- df_airbnb_data_2[train_index, ]
test_data <- df_airbnb_data_2[-train_index, ]
```

Fitting Linear Regression

```
lm_model <- lm(original_price ~ state_encoded + view + rating + no_of_rates + sentiment_score, data = t
lm_pred <- predict(lm_model, newdata = test_data)
sqrt(mean((lm_pred - test_data$original_price)^2))
```

```
## [1] 131.3833
```

Fitting Random Forest Regressor

```
rf_model <- randomForest(original_price ~ state_encoded + view + rating + no_of_rates + sentiment_score
```

```
rf_pred <- predict(rf_model, newdata = test_data)
sqrt(mean((rf_pred - test_data$original_price)^2))
```

```
## [1] 68.58096
```

Doing for each state using ModelR

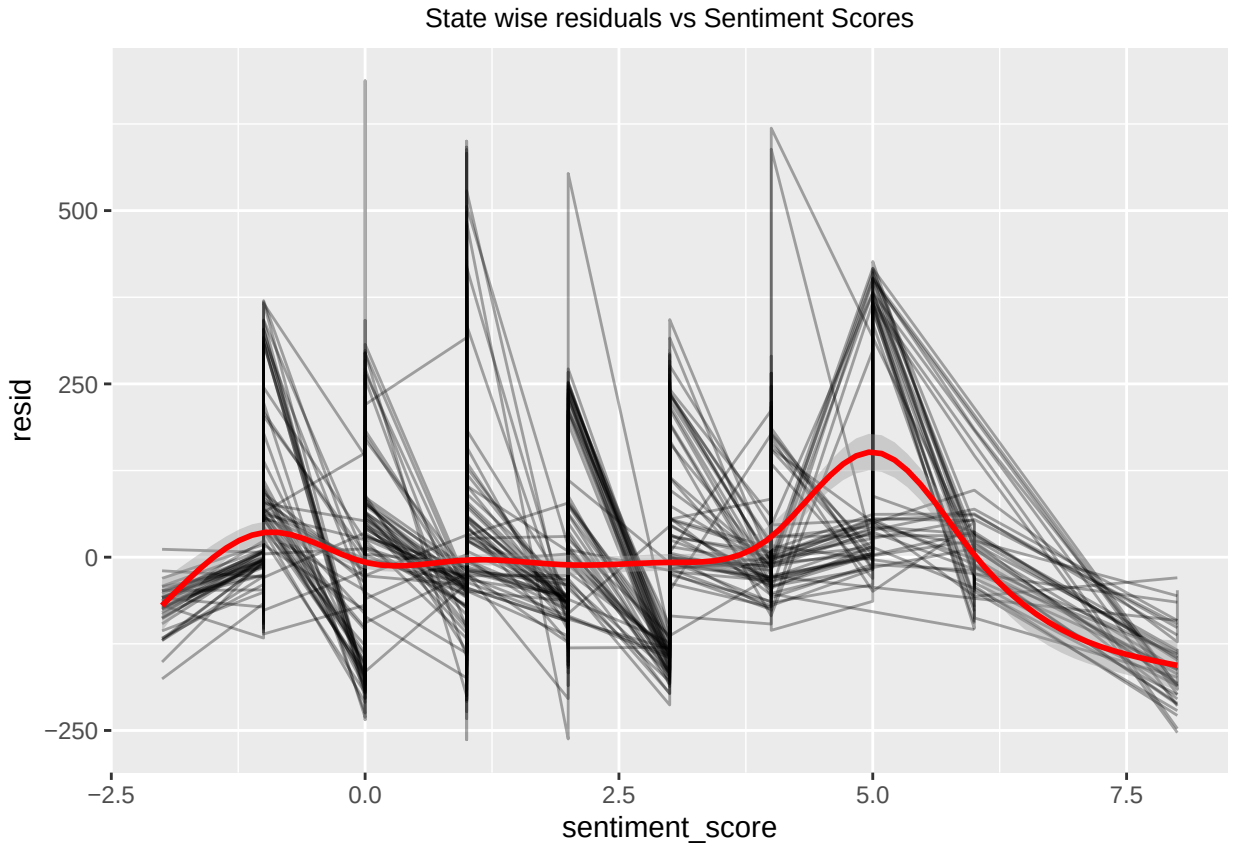
```
model_lm <- function(df) {
  lm(original_price ~ view + rating + no_of_rates + sentiment_score, data = df)
}
```

```
airbnb_data_nested <- df_airbnb_data_2 %>%
  group_by(state) %>%
  nest()
```

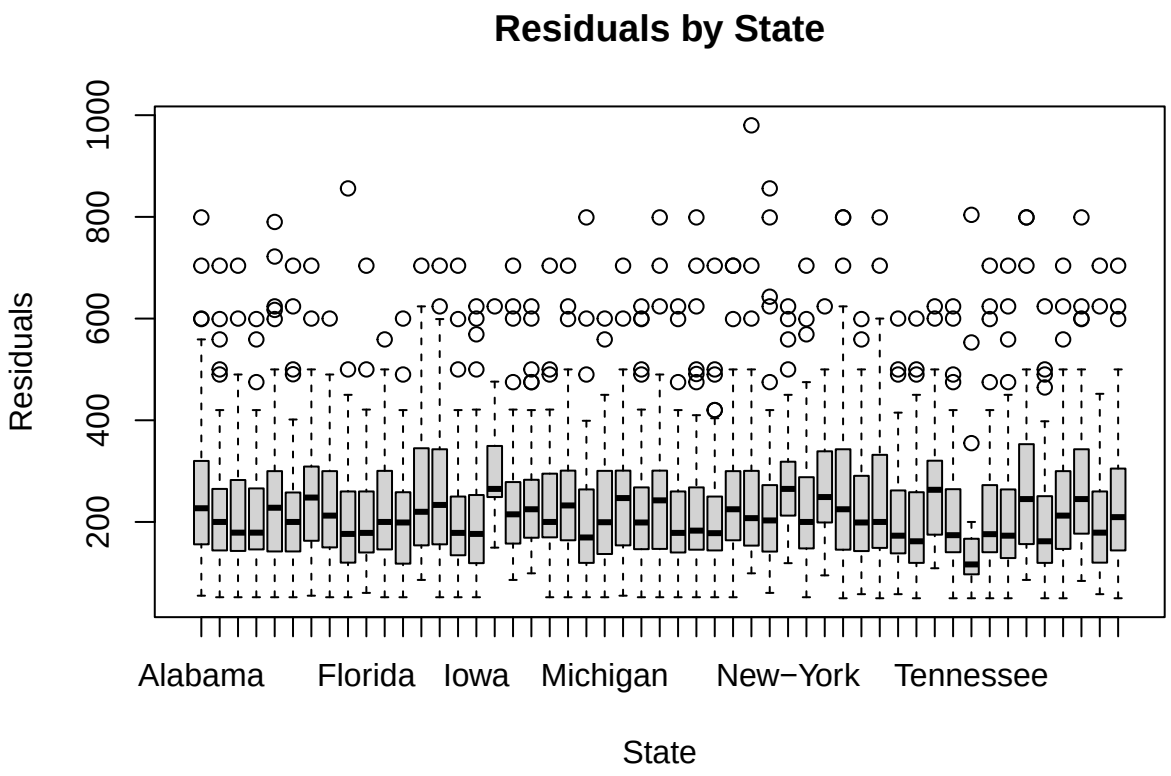
```
airbnb_data_nested_models <- airbnb_data_nested %>%
  mutate(model = map(data, model_lm)) %>%
  mutate(lm_glance = map(model, augment))  %>%
  mutate(resid = map2(data, model, add_residuals))
```

```
resid <- unnest(airbnb_data_nested_models, resid)
resid %>% ggplot(aes(sentiment_score, resid)) +
                  geom_line(alpha = 1/3, aes(group=state)) +
                  geom_smooth(color = "red") + ggtitle("State wise residuals vs Sentiment Scores") + th
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```




```
# create a boxplot of residuals by state
boxplot(resid$original_price ~ resid$state,
        xlab = "State",
        ylab = "Residuals",
        main = "Residuals by State")
```



TFIDF Extraction

Extract TFIDF for each word in description to add it dataframe

```
tdm <- TermDocumentMatrix(text_corpus)
tfidf <- weightTfIdf(tdm)
df_tfidf <- as.data.frame(as.matrix(tfidf))
df_tfidf_t <- data.frame(t(df_tfidf))
df_tfidf_t_filtered <- df_tfidf_t[, colSums(df_tfidf_t != 0) > 0]
```

```
library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
library(RColorBrewer)

# Calculate the sum of TF-IDF scores for each word
word_freq <- colSums(df_tfidf_t)

# Sort the words based on their frequency
word_freq <- sort(word_freq, decreasing = TRUE)

# Create a word cloud using the top 100 words
wordcloud(names(word_freq)[1:75], word_freq[1:75], colors = brewer.pal(8, "Dark2"))
```

```
## Warning in wordcloud(names(word_freq)[1:75], word_freq[1:75], colors =
## brewer.pal(8, : hous could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(names(word_freq)[1:75], word_freq[1:75], colors =
## brewer.pal(8, : cottag could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(names(word_freq)[1:75], word_freq[1:75], colors =
## brewer.pal(8, : beach could not be fit on page. It will not be plotted.
```


```
# PCA
pca <- prcomp(df_tfidf_t, center = TRUE, scale. = TRUE)

# Extract the first 10 principal components
pc_scores <- data.frame(pca$x[, 1:10])
```

```
df_airbnb_data_tfidf <- df_airbnb_data %>%
  select(-description) %>%
  bind_cols(pc_scores)
```

```
df_airbnb_data_3 <- df_airbnb_data_tfidf %>%
  select(-place_name, -from_date, -to_date, -price, -price_str)
df_airbnb_data_3 <- na.omit(df_airbnb_data_3)
```

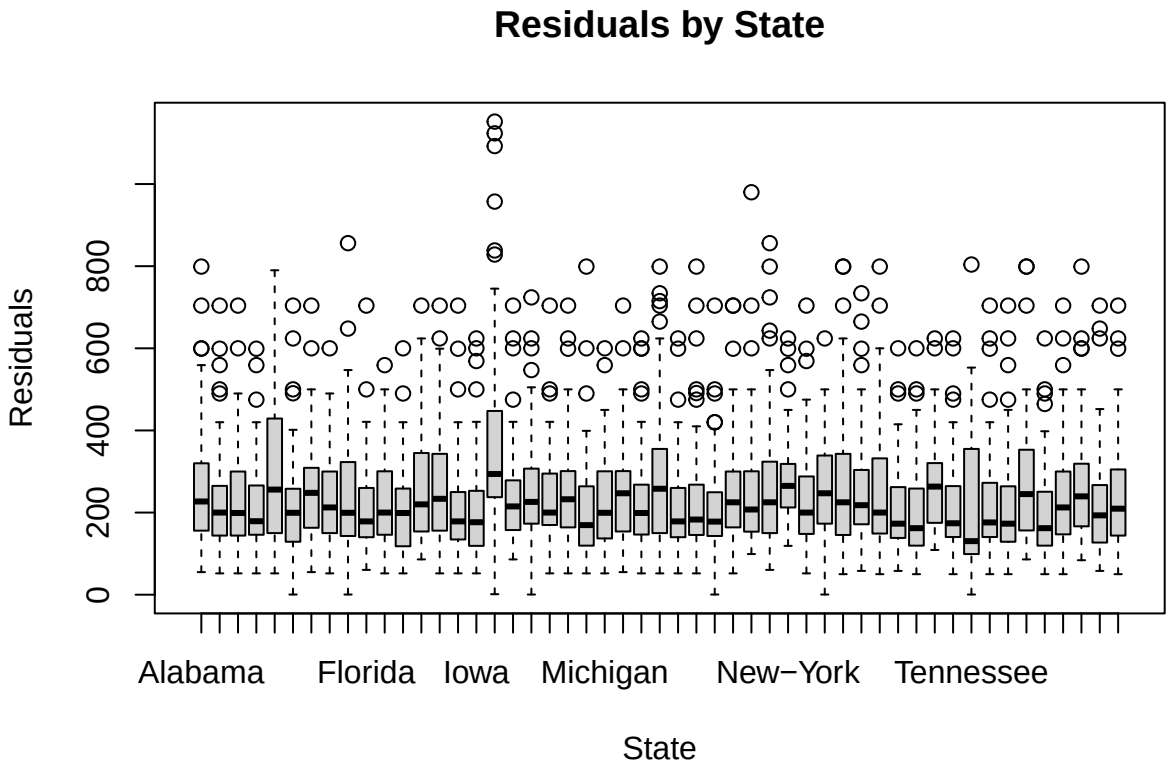
```
model_lm <- function(df) {
  lm(original_price ~ ., data = df)
}
```

```
airbnb_data_nested <- df_airbnb_data_3 %>%
  group_by(state) %>%
  nest()
```

```
airbnb_data_nested_models <- airbnb_data_nested %>%
  mutate(model = map(data, model_lm)) %>%
  mutate(lm_glance = map(model, augment))  %>%
  mutate(resid = map2(data, model, add_residuals))
```

```
resid <- unnest(airbnb_data_nested_models, resid)
```

```
# create a boxplot of residuals by state
boxplot(resid$original_price ~ resid$state,
        xlab = "State",
        ylab = "Residuals",
        main = "Residuals by State")
```




```
df_airbnb_data_3$state <- as.numeric(factor(df_airbnb_data_3$state))
```

```
train_index <- sample(1:nrow(df_airbnb_data_3), 0.7*nrow(df_airbnb_data_3))
train_data <- df_airbnb_data_3[train_index, ]
test_data <- df_airbnb_data_3[-train_index, ]
```

Fitting Linear Model

```
lm_model <- lm(rating ~ ., data = train_data)
lm_pred <- predict(lm_model, newdata = test_data)
```

```
sqrt(mean((lm_pred - test_data$original_price)^2))
```

```
## [1] 284.2316
```

Fitting Random Forest Regressor

```
rf_model <- randomForest(original_price ~ ., data = train_data, ntree = 500, importance = TRUE)
```

```
rf_pred <- predict(rf_model, newdata = test_data)
sqrt(mean((rf_pred - test_data$original_price)^2))
```

```
## [1] 70.2831
```