# finalproject2

Andro Manukov

May 10, 2020

Installing Packages:

```r
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.5.3
```

```
## -- Attaching packages --------------------------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  3.0.0     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## Warning: package 'ggplot2' was built under R version 3.5.3
```

```
## Warning: package 'tibble' was built under R version 3.5.3
```

```
## Warning: package 'tidyr' was built under R version 3.5.3
```

```
## Warning: package 'readr' was built under R version 3.5.3
```

```
## Warning: package 'purrr' was built under R version 3.5.3
```

```
## Warning: package 'dplyr' was built under R version 3.5.3
```

```
## Warning: package 'stringr' was built under R version 3.5.3
```

```
## Warning: package 'forcats' was built under R version 3.5.3
```

```
## -- Conflicts ------------------------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(GGally)
```

```
## Warning: package 'GGally' was built under R version 3.5.3
```

```
## 
## Attaching package: 'GGally'
```

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

```r
library(ggplot2)
library(scales)
```

```
## Warning: package 'scales' was built under R version 3.5.3
```

```
## 
## Attaching package: 'scales'
```

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

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```r
library(gridExtra)
```

```
## Warning: package 'gridExtra' was built under R version 3.5.3
```

```
##
## Attaching package: 'gridExtra'
```

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

```r
library(grid)
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.5.3
```

```
## randomForest 4.6-14
```

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

```
##
## Attaching package: 'randomForest'
```

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

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

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

```r
library(lmtest)
```

```
## Warning: package 'lmtest' was built under R version 3.5.3
```

```
## Loading required package: zoo
```

```
## Warning: package 'zoo' was built under R version 3.5.3
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

Reading csv and subsetting to just city hotels:

```r
Hotel <- read_csv("D:/UIUC/Spring2020/stat425/finalproject/stat425_fpdata.csv")
```

```
## Parsed with column specification:
## cols(
##   hotel = col_character(),
##   is_canceled = col_double(),
##   lead_time = col_double(),
##   arrival_date_year = col_double(),
##   arrival_date_month = col_character(),
##   arrival_date_week_number = col_double(),
##   arrival_date_day_of_month = col_double(),
##   stays_in_weekend_nights = col_double(),
##   stays_in_week_nights = col_double(),
##   adults = col_double(),
##   children = col_double(),
##   babies = col_double(),
##   meal = col_character(),
##   market_segment = col_character(),
##   reserved_room_type = col_character(),
##   customer_type = col_character(),
##   adr = col_double(),
##   total_of_special_requests = col_double()
## )
```

```r
city_hotels <- subset(Hotel, Hotel$hotel == "City Hotel") #Subset just city hotels because im section two.
```

Creating seasons variable:

```r
city_hotels$arrival_season = "" #init variable
i = 1 #index for rows
for(val in city_hotels$arrival_date_month) #increment through each row
{
  if(val == "January" | val == "February" | val == "December") { #conditional
  city_hotels$arrival_season[i] = "Winter" #change to season
} else if(val == "March" | val == "May" | val == "April") {
  city_hotels$arrival_season[i] = "Spring"
} else if(val == "June" | val == "July" | val == "August") {
  city_hotels$arrival_season[i] = "Summer"
} else {
  city_hotels$arrival_season[i] = "Fall"
}

i = i + 1 #incremenet index
}
```

```r
head(city_hotels)
```

```
## # A tibble: 6 x 19
##   hotel is_canceled lead_time arrival_date_ye~ arrival_date_mo~
##   <chr>       <dbl>     <dbl>            <dbl> <chr>
## 1 City~           0         0             2015 August
## 2 City~           0         1             2015 August
## 3 City~           0         5             2015 August
## 4 City~           0        39             2015 August
## 5 City~           0         3             2015 September
## 6 City~           0        82             2015 September
## # ... with 14 more variables: arrival_date_week_number <dbl>,
## #   arrival_date_day_of_month <dbl>, stays_in_weekend_nights <dbl>,
## #   stays_in_week_nights <dbl>, adults <dbl>, children <dbl>,
## #   babies <dbl>, meal <chr>, market_segment <chr>,
## #   reserved_room_type <chr>, customer_type <chr>, adr <dbl>,
## #   total_of_special_requests <dbl>, arrival_season <chr>
```

Creating total stays variable:

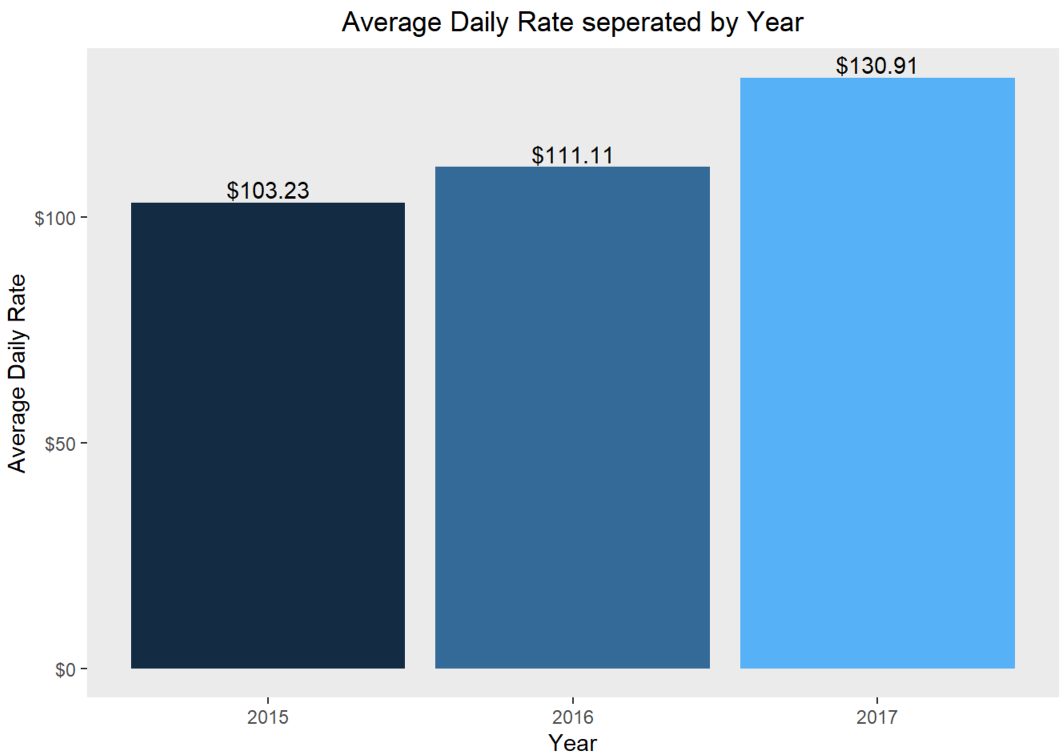
```r
city_hotels$total_stays <- city_hotels$stays_in_weekend_nights + city_hotels$stays_in_week_nights #creating total stays variable
```

Creating Bar Plots:

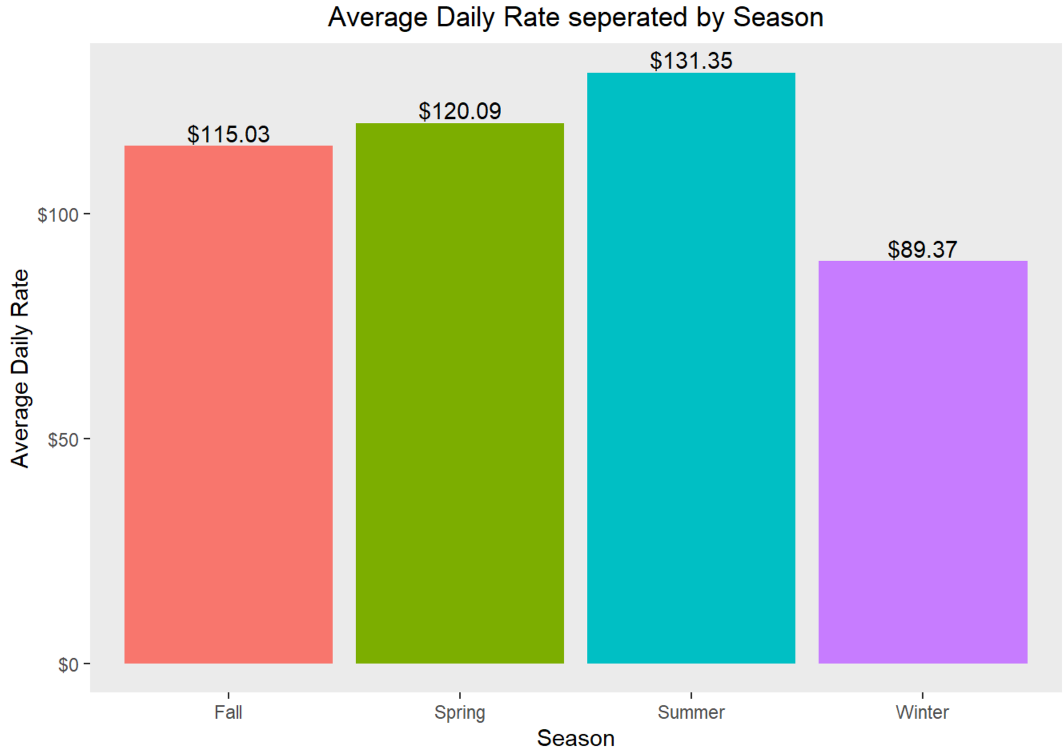
```
#READ THIS ONE FOR A GENERAL TREND OF ALL OF THE OTHER BAR PLOTS!!!!!!!!!!!!
#THIS BAR PLOT IS FOR ADR SEPERATED BY YEAR!!!!!!!!!!

year_group <- summarise_at(group_by(city_hotels, arrival_date_year), vars(adr), funs(mean(.,na.rm=FALSE))) #
GROUPBY THE VARIABLE WE ARE INVESTIGATING
```

```
## Warning: funs() is soft deprecated as of dplyr 0.8.0
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once per session.
```

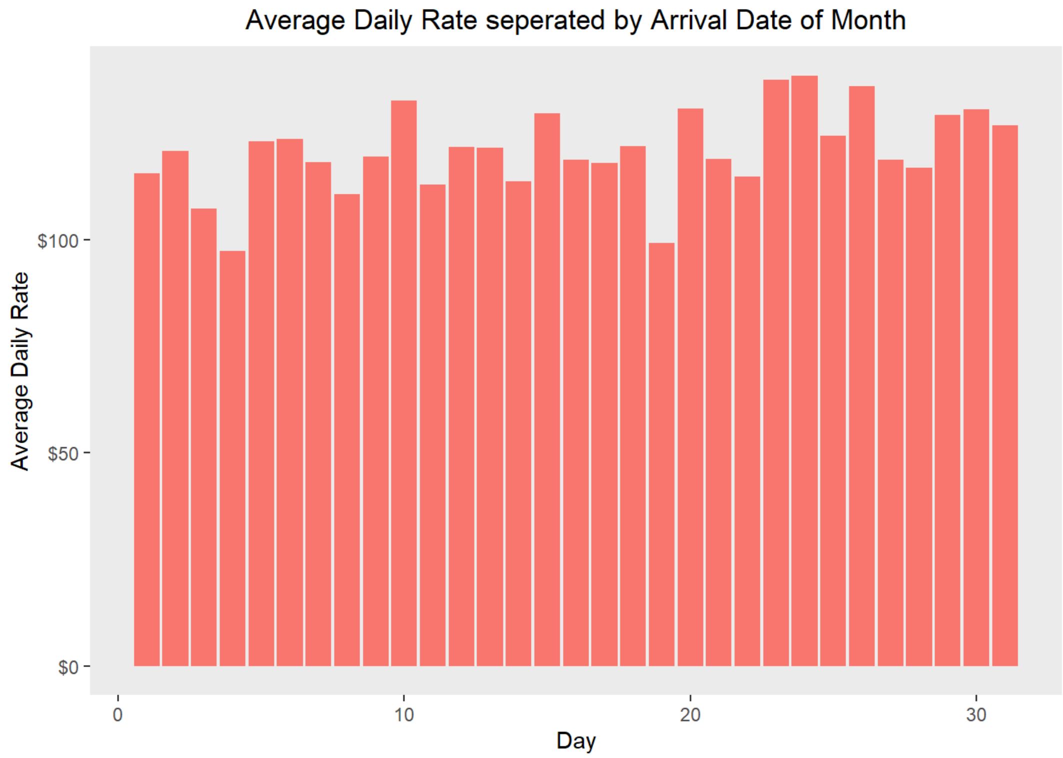
```
ggplot(data = year_group, aes(x = arrival_date_year, y = adr, fill = arrival_date_year)) +
  geom_bar(position = 'dodge', stat='identity') + #create bar plot
  geom_text(aes(label= dollar(adr)), position=position_dodge(width=0.9), vjust=-0.25) + #add text to bar plo
t
  xlab("Year") + ylab("Average Daily Rate") + ggtitle("Average Daily Rate seperated by Year") + scale_y_cont
inuous(labels = scales::dollar) + theme(legend.position = "none", panel.grid =   element_blank(), #add all t
itles and everything
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5)) #ADD X, Y, TITLE LABELS AND ADJUST THEM
```



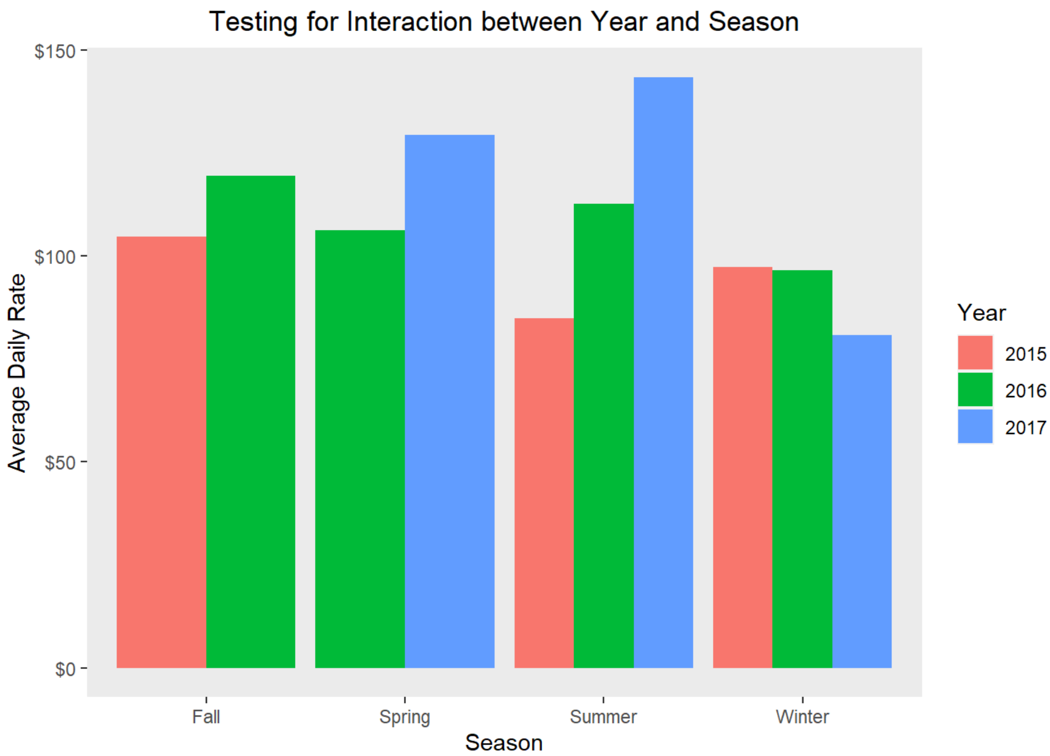

```
#plot for Average Daily Rate seperated by Season
season_group <- summarise_at(group_by(city_hotels, arrival_season), vars(adr), funs(mean(.,na.rm=FALSE)))

ggplot(data = season_group, aes(x = arrival_season, y = adr, fill = arrival_season)) +
  geom_bar(position = 'dodge', stat='identity') +
  geom_text(aes(label= dollar(adr)), position=position_dodge(width=0.9), vjust=-0.25) +
  xlab("Season") + ylab("Average Daily Rate") + ggtitle("Average Daily Rate seperated by Season") +   scale_
y_continuous(labels = scales::dollar) + theme(legend.position = "none", panel.grid =   element_blank(),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5))
```



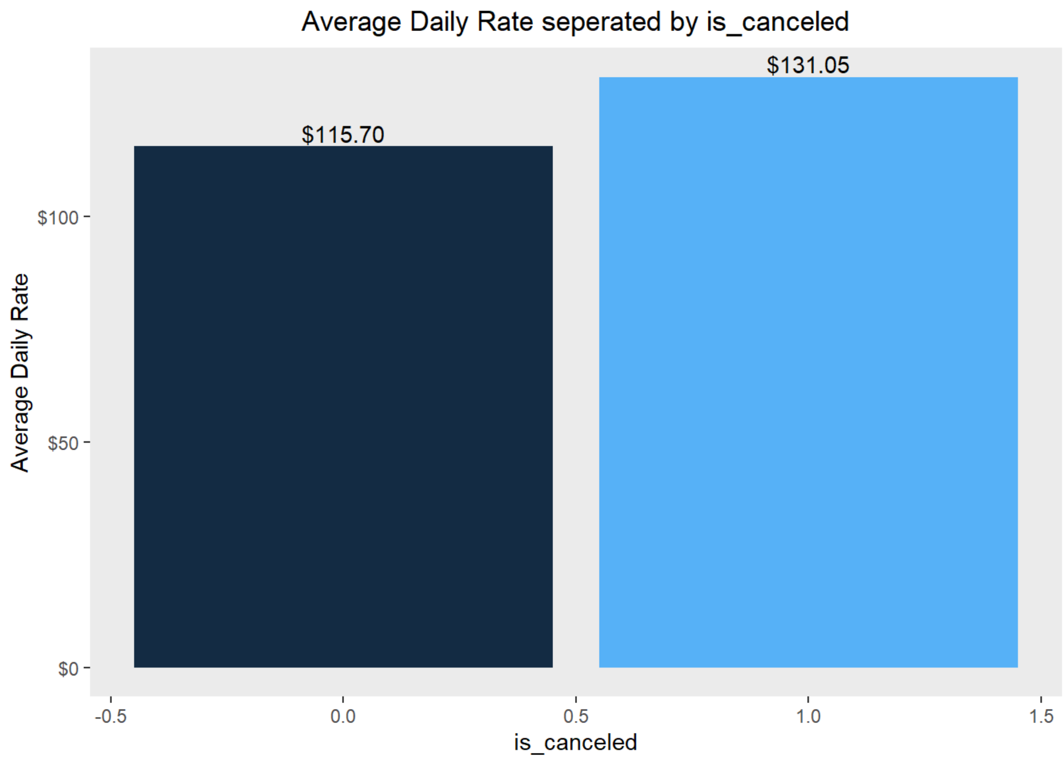

```
#plot for Average Daily Rate seperated by Arrival Date of Month"
day_group <- summarise_at(group_by(city_hotels, arrival_date_day_of_month), vars(adr), funs(mean(.,na.rm=FAL
SE)))

ggplot(data = day_group, aes(x = arrival_date_day_of_month, y = adr, fill = "red")) +
  geom_bar(position = 'dodge', stat='identity') +
  xlab("Day") + ylab("Average Daily Rate") + ggtitle("Average Daily Rate seperated by Arrival Date of Month"
) +   scale_y_continuous(labels = scales::dollar) + theme(legend.position = "none", panel.grid =   element_
blank(),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5))
```



```r
#creating plot to test for interaction between year and season
year_season_group <- summarise_at(group_by(city_hotels, arrival_season, arrival_date_year), vars(adr), funs(
mean(.,na.rm=FALSE)))

ggplot(year_season_group, aes(x=arrival_season, y=adr, fill = as.factor(arrival_date_year))) +
    geom_bar(position = 'dodge', stat='identity') +
  xlab("Season") + ylab("Average Daily Rate") + ggtitle("Testing for Interaction between Year and Season") +
scale_y_continuous(labels = scales::dollar) + theme(panel.grid = element_blank(),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5)) + labs(fill = "Year")
```



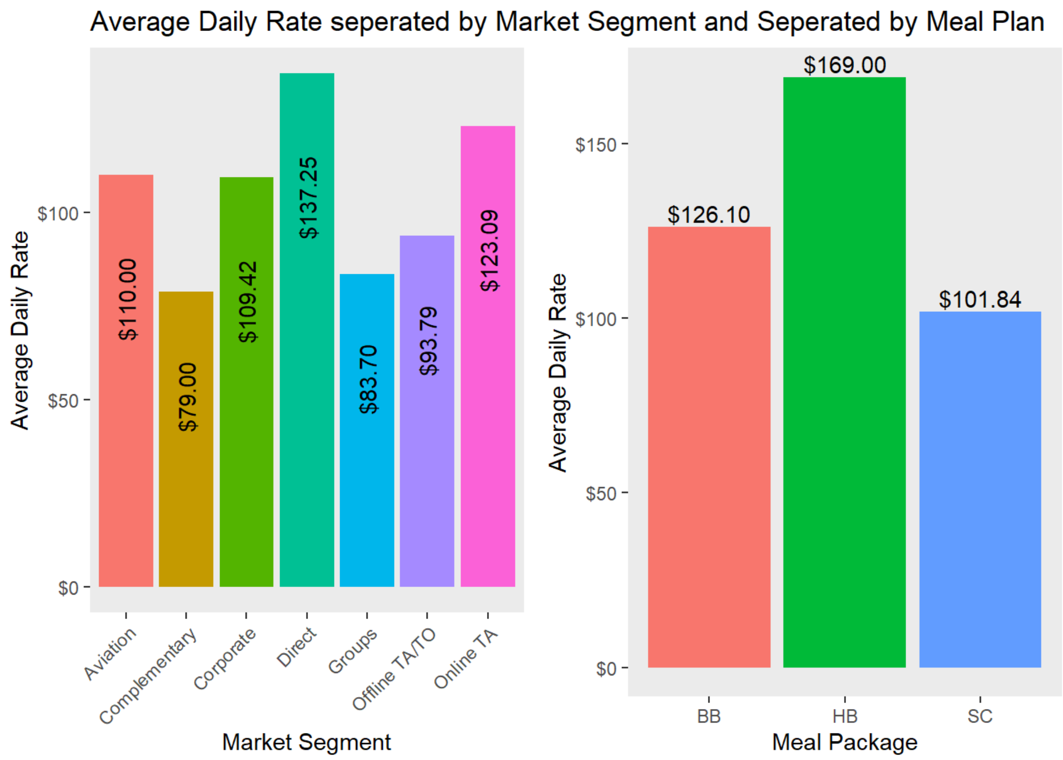

```r
#creating is canceled group plot.
is_canceled_group <- summarise_at(group_by(city_hotels, is_canceled), vars(adr), funs(mean(.,na.rm=FALSE)))

ggplot(data = is_canceled_group, aes(x = is_canceled, y = adr, fill = is_canceled)) +
  geom_bar(position = 'dodge', stat='identity') +
  geom_text(aes(label= dollar(adr)), position=position_dodge(width=0.9), vjust=-0.25) +
  xlab("is_canceled") + ylab("Average Daily Rate") + ggtitle("Average Daily Rate seperated by is_canceled")
+   scale_y_continuous(labels = scales::dollar) + theme(legend.position = "none", panel.grid =   element_bl
ank(),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5))
```




```
#creating plot for meal group
meal_group <- summarise_at(group_by(city_hotels, meal), vars(adr), funs(mean(.,na.rm=FALSE)))

meal_package_plot <- ggplot(data = meal_group, aes(x = meal, y = adr, fill = meal)) +
  geom_bar(position = 'dodge', stat='identity') +
  geom_text(aes(label= dollar(adr)), position=position_dodge(width=0.9), vjust=-0.25) +
  xlab("Meal Package") + ylab("Average Daily Rate") + ggtitle("") +   scale_y_continuous(labels = scales::do
llar) + theme(legend.position = "none", panel.grid =   element_blank(),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5))
```

```
#mcreating plot for market_group
market_group <- summarise_at(group_by(city_hotels, market_segment), vars(adr), funs(mean(.,na.rm=FALSE)))

market_segment_plot <- ggplot(data = market_group, aes(x = market_segment, y = adr, fill = market_segment))
+
  geom_bar(position = 'dodge', stat='identity') +
  geom_text(aes(label= dollar(adr), angle = 90), hjust= 2) +
  xlab("Market Segment") + ylab("Average Daily Rate") + ggtitle("Average Daily Rate seperated by Market Segm
ent and Seperated by Meal Plan") +   scale_y_continuous(labels = scales::dollar) + theme(legend.position = "
none", panel.grid =   element_blank(),
        plot.title = element_text(hjust = 0),
        plot.subtitle = element_text(hjust = 0.5),
        axis.text.x = element_text(angle = 45, hjust = 1))
```

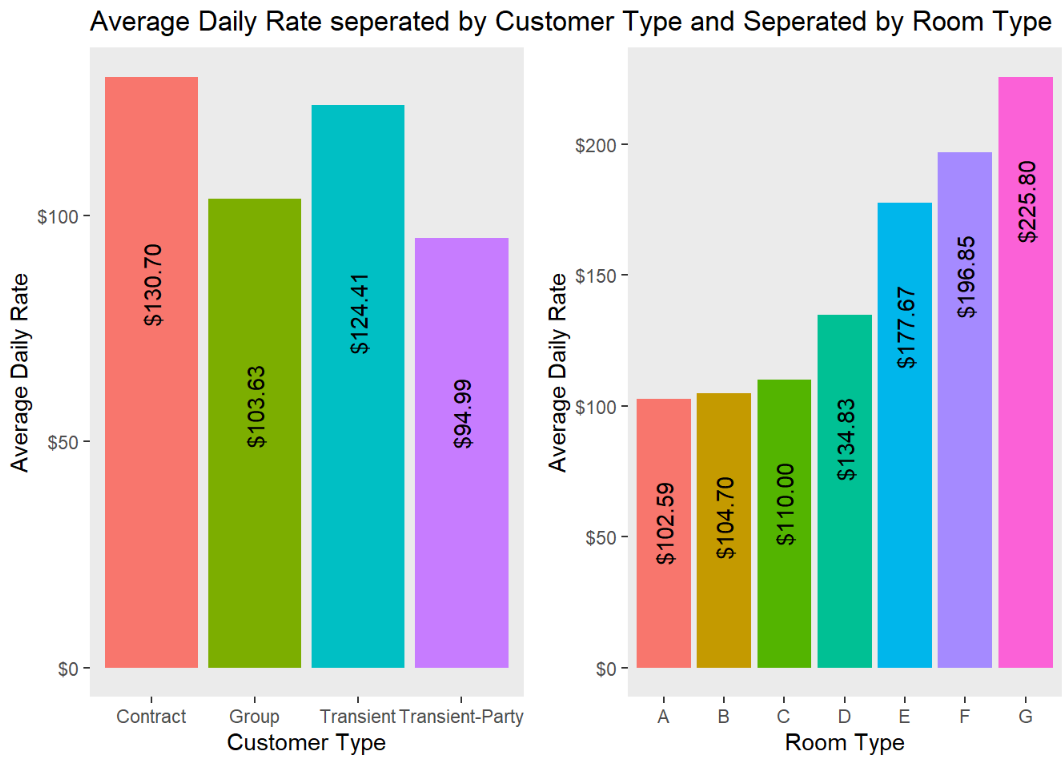
```
grid.arrange(meal_package_plot, market_segment_plot, layout_matrix = cbind(c(2,2), c(1,1))) #combining to sa
ve space
```



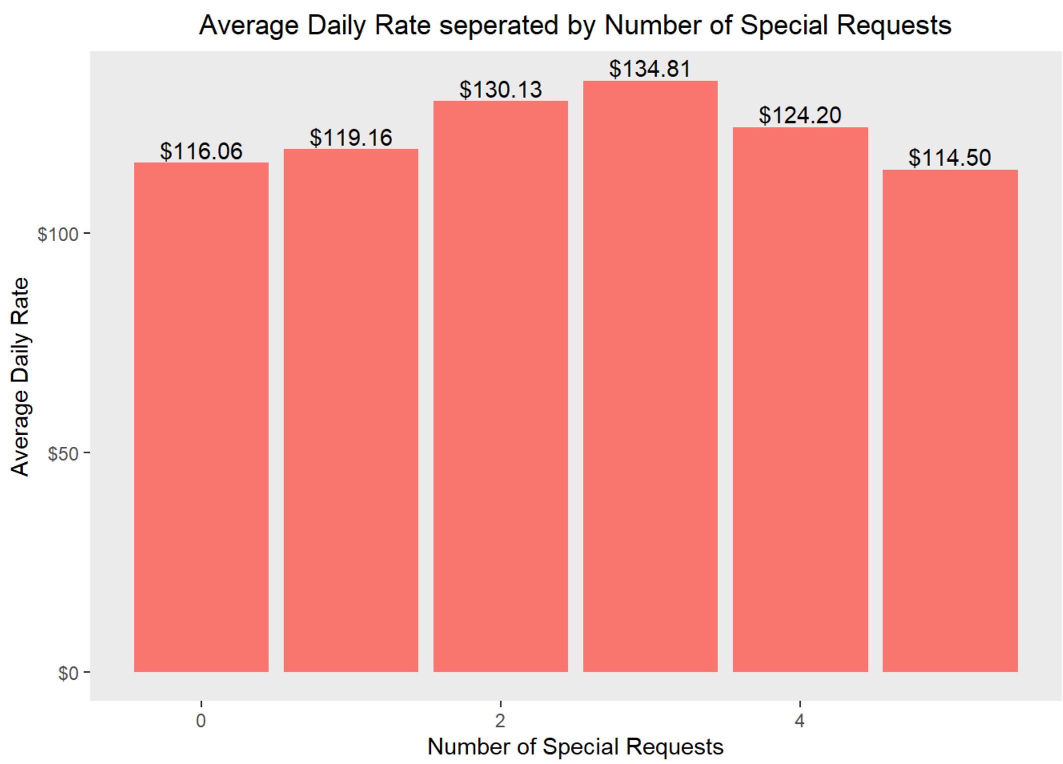

```
#creating room type plot
reserve_group <- summarise_at(group_by(city_hotels, reserved_room_type), vars(adr), funs(mean(.,na.rm=FALSE)
))

reserve_group_plot <- ggplot(data = reserve_group, aes(x = reserved_room_type, y = adr, fill = reserved_room
_type)) +
  geom_bar(position = 'dodge', stat='identity') +
  geom_text(aes(label= dollar(adr), angle = 90), hjust = 2) +
  xlab("Room Type") + ylab("Average Daily Rate") + ggtitle("") +   scale_y_continuous(labels = scales::dolla
r) + theme(legend.position = "none", panel.grid =   element_blank(),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5))
```

```
#creating customer group plot
customer_group <- summarise_at(group_by(city_hotels, customer_type), vars(adr), funs(mean(.,na.rm=FALSE)))

customer_group_plot <- ggplot(data = customer_group, aes(x = customer_type, y = adr, fill = customer_type))
+
  geom_bar(position = 'dodge', stat='identity') +
  geom_text(aes(label= dollar(adr), angle = 90), hjust = 3) +
  xlab("Customer Type") + ylab("Average Daily Rate") + ggtitle("Average Daily Rate seperated by Customer Typ
e and Seperated by Room Type") +   scale_y_continuous(labels = scales::dollar) + theme(legend.position = "no
ne", panel.grid =   element_blank(),
        plot.title = element_text(hjust = 0),
        plot.subtitle = element_text(hjust = 0.5))
```

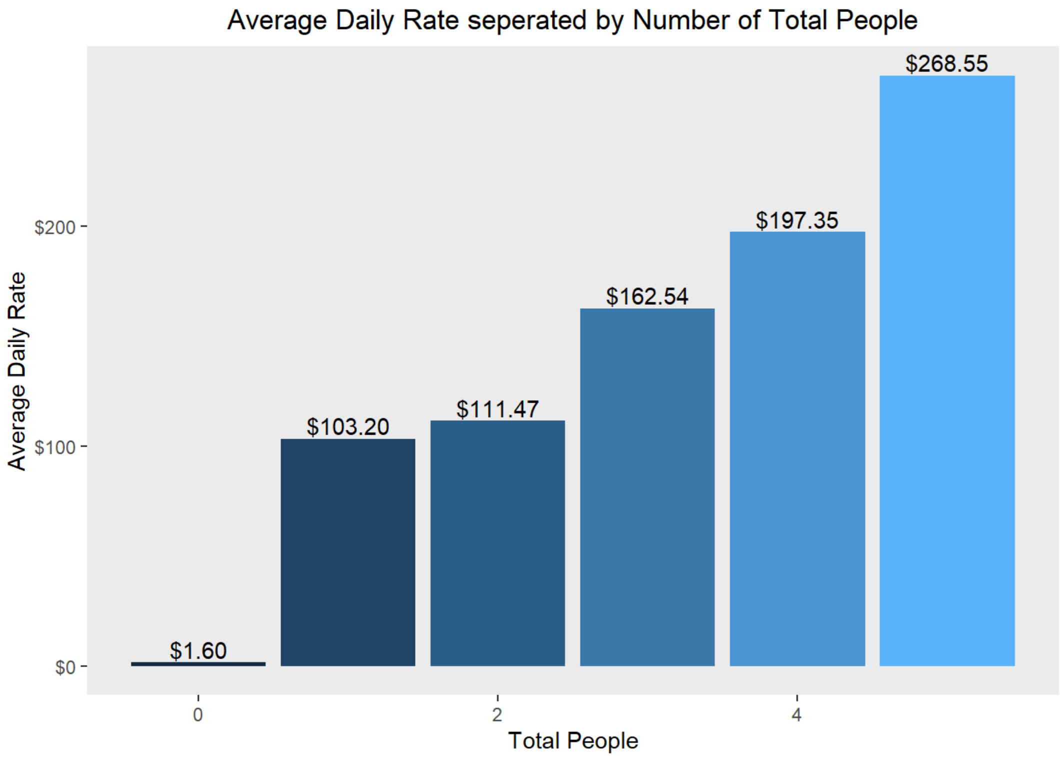
```
#combining two plots to save space :(
grid.arrange(reserve_group_plot, customer_group_plot, layout_matrix = cbind(c(2,2), c(1,1)))
```




```
#looking at Average Daily Rate seperated by Number of Special Requests
special_group <- summarise_at(group_by(city_hotels, total_of_special_requests), vars(adr), funs(mean(.,na.rm
=FALSE)))

ggplot(data = special_group, aes(x = total_of_special_requests, y = adr, fill = "red")) +
  geom_bar(position = 'dodge', stat='identity') +
  geom_text(aes(label= dollar(adr)), position=position_dodge(width=0.9), vjust=-0.25) +
  xlab("Number of Special Requests") + ylab("Average Daily Rate") + ggtitle("Average Daily Rate seperated by
Number of Special Requests") +   scale_y_continuous(labels = scales::dollar) + theme(legend.position = "none
", panel.grid =   element_blank(),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5))
```



```
#looking at Average Daily Rate seperated by Number of Total People"
city_hotels$total_people <- city_hotels$adults + city_hotels$children

people_group <- summarise_at(group_by(city_hotels, total_people), vars(adr), funs(mean(.,na.rm=FALSE)))

ggplot(data = people_group, aes(x = total_people, y = adr, fill = total_people)) +
  geom_bar(position = 'dodge', stat='identity') +
  geom_text(aes(label= dollar(adr)), position=position_dodge(width=0.9), vjust=-0.25) +
  xlab("Total People") + ylab("Average Daily Rate") + ggtitle("Average Daily Rate seperated by Number of Tot
al People") +   scale_y_continuous(labels = scales::dollar) + theme(legend.position = "none", panel.grid =
element_blank(),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5))
```

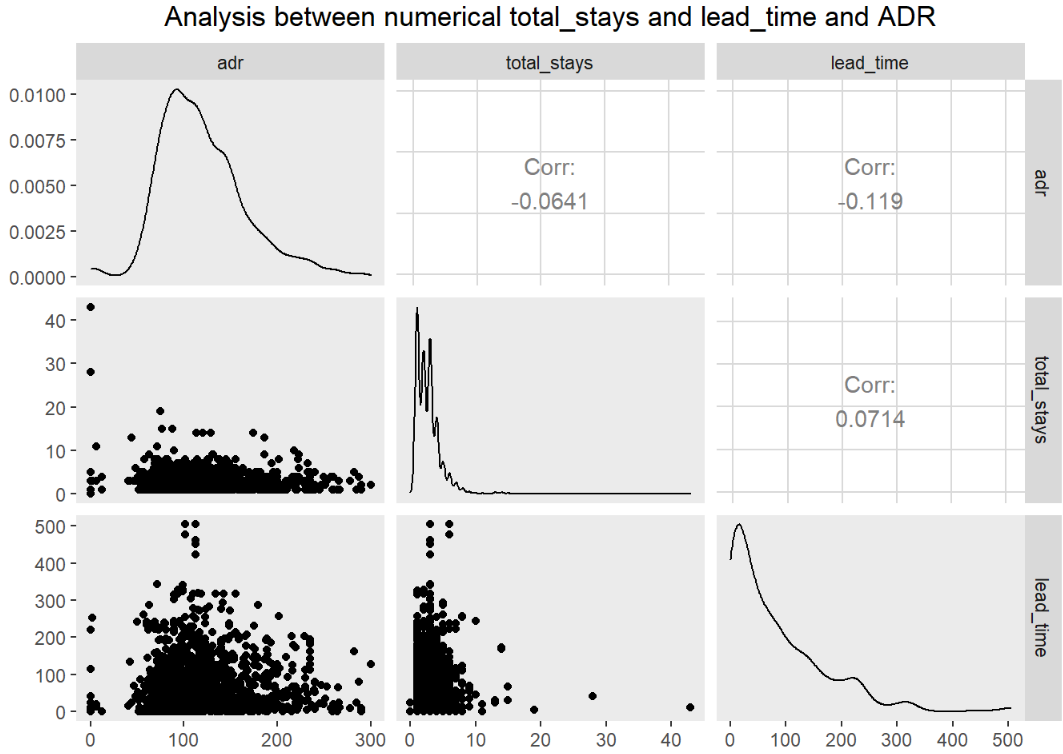



```
head(city_hotels)
```

```
## # A tibble: 6 x 21
##   hotel is_canceled lead_time arrival_date_ye~ arrival_date_mo~
##   <chr>       <dbl>     <dbl>            <dbl> <chr>
## 1 City~           0         0             2015 August
## 2 City~           0         1             2015 August
## 3 City~           0         5             2015 August
## 4 City~           0        39             2015 August
## 5 City~           0         3             2015 September
## 6 City~           0        82             2015 September
## # ... with 16 more variables: arrival_date_week_number <dbl>,
## #   arrival_date_day_of_month <dbl>, stays_in_weekend_nights <dbl>,
## #   stays_in_week_nights <dbl>, adults <dbl>, children <dbl>,
## #   babies <dbl>, meal <chr>, market_segment <chr>,
## #   reserved_room_type <chr>, customer_type <chr>, adr <dbl>,
## #   total_of_special_requests <dbl>, arrival_season <chr>,
## #   total_stays <dbl>, total_people <dbl>
```

Testing Numerical Variables:

```
ggpairs(select(city_hotels, c(adr, total_stays, lead_time))) + ggtitle("Analysis between numerical total_sta
ys and lead_time and ADR") + theme(legend.position = "none", panel.grid =   element_blank(),
        plot.title = element_text(hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5))
```




```
#looking at total stays and lead time
```

Subsetting Dataset once more:

```
city_hotels_removed <- select(city_hotels, is_canceled, reserved_room_type, arrival_season, total_people, me
al, market_segment, customer_type, arrival_date_year, adr) #subsetting data
```

Converting to factor variables:

```
city_hotels_removed$is_canceled = as.factor(city_hotels$is_canceled)
city_hotels_removed$reserved_room_type = as.factor(city_hotels$reserved_room_type)
city_hotels_removed$arrival_season = as.factor(city_hotels$arrival_season)
city_hotels_removed$total_people = as.factor(city_hotels$total_people)
city_hotels_removed$meal = as.factor(city_hotels$meal)
city_hotels_removed$market_segment = as.factor(city_hotels$market_segment)
city_hotels_removed$customer_type = as.factor(city_hotels$customer_type)
city_hotels_removed$arrival_date_year = as.factor(city_hotels$arrival_date_year)
#creating factor variables
```

```
head(city_hotels_removed)
```

```
## # A tibble: 6 x 9
##   is_canceled reserved_room_t~ arrival_season total_people meal
##   <fct>       <fct>            <fct>          <fct>        <fct>
## 1 0           A                Summer         2            SC
## 2 0           A                Summer         2            SC
## 3 0           A                Summer         2            BB
## 4 0           A                Summer         1            HB
## 5 0           A                Fall           2            BB
## 6 0           A                Fall           2            BB
## # ... with 4 more variables: market_segment <fct>, customer_type <fct>,
## #   arrival_date_year <fct>, adr <dbl>
```

Creating simple model:

```
simple_model <- lm(adr ~ . , data = city_hotels_removed)
summary(simple_model) #fitting simple model, looking at summary
```

```
##
## Call:
## lm(formula = adr ~ ., data = city_hotels_removed)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -164.39  -16.12   -0.82   15.27  108.71
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   17.443570  31.508732   0.554   0.5799
## is_canceled1                   5.195600   1.670484   3.110   0.0019 **
## reserved_room_typeB          -10.250066   5.302841  -1.933   0.0534 .
## reserved_room_typeC           10.055988  28.025123   0.359   0.7198
## reserved_room_typeD           17.629094   2.000058   8.814  < 2e-16 ***
## reserved_room_typeE           50.213001   3.634015  13.818  < 2e-16 ***
## reserved_room_typeF           60.216543   4.960476  12.139  < 2e-16 ***
## reserved_room_typeG           96.488950   5.309933  18.171  < 2e-16 ***
## arrival_seasonSpring         -11.333639   2.382911  -4.756 2.15e-06 ***
## arrival_seasonSummer          -7.379294   2.372433  -3.110   0.0019 **
## arrival_seasonWinter         -38.781802   2.741657 -14.145  < 2e-16 ***
## total_people1                105.031219  13.963074   7.522 8.97e-14 ***
## total_people2                106.687744  13.924728   7.662 3.17e-14 ***
## total_people3                133.965301  14.068555   9.522  < 2e-16 ***
## total_people4                131.698746  14.760775   8.922  < 2e-16 ***
## total_people5                162.215765  20.207177   8.028 1.91e-15 ***
## mealHB                        32.179596   5.204559   6.183 7.98e-10 ***
## mealSC                        -8.957874   1.911914  -4.685 3.03e-06 ***
## market_segmentComplementary  -21.541852  39.798005  -0.541   0.5884
## market_segmentCorporate      -11.712512  28.002111  -0.418   0.6758
## market_segmentDirect           4.512965  27.497980   0.164   0.8697
## market_segmentGroups         -24.656194  27.714060  -0.890   0.3738
## market_segmentOffline TA/TO  -20.112166  27.600236  -0.729   0.4663
## market_segmentOnline TA        0.009288  27.476384   0.000   0.9997
## customer_typeGroup           -21.932937  11.369748  -1.929   0.0539 .
## customer_typeTransient       -16.430947   7.287788  -2.255   0.0243 *
## customer_typeTransient-Party -14.520719   7.960738  -1.824   0.0683 .
## arrival_date_year2016          3.956157   3.291770   1.202   0.2296
## arrival_date_year2017         20.260854   3.655920   5.542 3.50e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 27.35 on 1589 degrees of freedom
## Multiple R-squared:  0.6401, Adjusted R-squared:  0.6337
## F-statistic: 100.9 on 28 and 1589 DF,  p-value: < 2.2e-16
```

BIC of simple model:

```
BIC(simple_model) #BIC for simple model
```

```
## [1] 15490.89
```

ADD A TRAIN AND TEST SET FOR 3.2 AND 3.3

Subsetting data to remove marketsegment:

```
city_hotels_removed <- select(city_hotels_removed, is_canceled, reserved_room_type, arrival_season, total_pe
ople, meal, customer_type, arrival_date_year, adr) #cutting down dataset
```

Creating test and train sets:

```r
# creating test and train datasets
## 75% of the sample size
smp_size <- floor(0.75 * nrow(city_hotels_removed))

## set the seed to make your partition reproducible
set.seed(123)
train_ind <- sample(seq_len(nrow(city_hotels_removed)), size = smp_size) #index

train <- city_hotels_removed[train_ind, ]
test <- city_hotels_removed[-train_ind, ]

#https://stackoverflow.com/questions/17200114/how-to-split-data-into-training-testing-sets-using-sample-func
tion
```

Creating complex model:

```r
complex_model <- lm(adr ~ . + arrival_season * arrival_date_year, data = train)
summary(complex_model) #fitting model and looking at summary
```

```
##
## Call:
## lm(formula = adr ~ . + arrival_season * arrival_date_year, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -131.500  -15.119   -0.333   15.476  101.239
##
## Coefficients: (2 not defined because of singularities)
##                                            Estimate Std. Error t value
## (Intercept)                                  22.872     15.721   1.455
## is_canceled1                                  3.935      1.920   2.050
## reserved_room_typeB                          -4.384      5.954  -0.736
## reserved_room_typeC                           1.846     27.492   0.067
## reserved_room_typeD                          20.869      2.319   8.997
## reserved_room_typeE                          52.990      4.068  13.026
## reserved_room_typeF                          69.156      5.644  12.253
## reserved_room_typeG                         101.208      6.150  16.456
## arrival_seasonSpring                         12.176      9.864   1.234
## arrival_seasonSummer                        -22.887     14.140  -1.619
## arrival_seasonWinter                        -30.643      8.898  -3.444
## total_people1                                99.787     14.102   7.076
## total_people2                               100.577     14.061   7.153
## total_people3                               125.362     14.241   8.803
## total_people4                               119.877     15.107   7.935
## total_people5                               159.145     22.018   7.228
## mealHB                                       32.347      6.097   5.306
## mealSC                                       -7.721      2.122  -3.639
## customer_typeGroup                          -30.951     13.019  -2.377
## customer_typeTransient                      -21.975      7.611  -2.887
## customer_typeTransient-Party                -35.415      7.854  -4.509
## arrival_date_year2016                         6.680      4.073   1.640
## arrival_date_year2017                         4.595      9.109   0.504
## arrival_seasonSpring:arrival_date_year2016  -22.015     10.374  -2.122
## arrival_seasonSummer:arrival_date_year2016   14.389     14.467   0.995
## arrival_seasonWinter:arrival_date_year2016    5.353      9.706   0.551
## arrival_seasonSpring:arrival_date_year2017       NA         NA      NA
## arrival_seasonSummer:arrival_date_year2017   39.449     16.632   2.372
## arrival_seasonWinter:arrival_date_year2017       NA         NA      NA
##                                            Pr(>|t|)
## (Intercept)                                0.145973
## is_canceled1                               0.040621 *
## reserved_room_typeB                        0.461657
## reserved_room_typeC                        0.946476
## reserved_room_typeD                         < 2e-16 ***
## reserved_room_typeE                         < 2e-16 ***
## reserved_room_typeF                         < 2e-16 ***
## reserved_room_typeG                         < 2e-16 ***
## arrival_seasonSpring                       0.217320
## arrival_seasonSummer                       0.105807
## arrival_seasonWinter                       0.000594 ***
## total_people1                              2.53e-12 ***
```

```
## total_people2                              1.48e-12 ***
## total_people3                               < 2e-16 ***
## total_people4                              4.83e-15 ***
## total_people5                              8.76e-13 ***
## mealHB                                     1.34e-07 ***
## mealSC                                     0.000286 ***
## customer_typeGroup                         0.017597 *
## customer_typeTransient                     0.003958 **
## customer_typeTransient-Party               7.15e-06 ***
## arrival_date_year2016                      0.101302
## arrival_date_year2017                      0.614057
## arrival_seasonSpring:arrival_date_year2016 0.034030 *
## arrival_seasonSummer:arrival_date_year2016 0.320142
## arrival_seasonWinter:arrival_date_year2016 0.581408
## arrival_seasonSpring:arrival_date_year2017       NA
## arrival_seasonSummer:arrival_date_year2017 0.017855 *
## arrival_seasonWinter:arrival_date_year2017       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 27.37 on 1186 degrees of freedom
## Multiple R-squared:  0.647,  Adjusted R-squared:  0.6392
## F-statistic:  83.6 on 26 and 1186 DF,  p-value: < 2.2e-16
```

BIC of complex model:

```
BIC(complex_model) #BIC of MLR model
```

```
## [1] 11642.17
```

Predict using complex model:

```
#predict_df = data.frame(is_canceled = "0", arrival_season = "Winter", arrival_date_year = #"2016", meal = "
BB", reserved_room_type = "C", customer_type = "Transient", total_people #= "4")

mean(predict(complex_model, test)) #predicting the ADR from the MLR model using test data
```

```
## Warning in predict.lm(complex_model, test): prediction from a rank-
## deficient fit may be misleading
```

```
## [1] 118.9234
```

```
head(city_hotels_removed) #viewing dataset for convience
```

```
## # A tibble: 6 x 8
##   is_canceled reserved_room_t~ arrival_season total_people meal
##   <fct>       <fct>            <fct>          <fct>        <fct>
## 1 0           A                Summer         2            SC
## 2 0           A                Summer         2            SC
## 3 0           A                Summer         2            BB
## 4 0           A                Summer         1            HB
## 5 0           A                Fall           2            BB
## 6 0           A                Fall           2            BB
## # ... with 3 more variables: customer_type <fct>, arrival_date_year <fct>,
## #   adr <dbl>
```

Creating Random Forest:

```
rfModel = randomForest(adr ~ ., ntree = 2000, data = train, seed = 425)
rfModel #creating the RF
```

```
##
## Call:
##  randomForest(formula = adr ~ ., data = train, ntree = 2000, seed = 425)
##                Type of random forest: regression
##                      Number of trees: 2000
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 748.6617
##                     % Var explained: 63.9
```

Importance values of rf:

```
rfModel$importance #importance of RF model variables
```

```
##                    IncNodePurity
## is_canceled             25685.17
## reserved_room_type     736625.71
## arrival_season         172309.42
## total_people           450918.36
## meal                   105655.29
## customer_type           77146.01
## arrival_date_year      128746.55
```

RMSE of RF:

```
mean(sqrt(rfModel$mse)) #rmse of RF model
```

```
## [1] 27.40389
```

Predict using RF:

```
predict_tree = predict(rfModel, data = test, predict.all = TRUE)
mean(predict_tree) #predict using RF
```

```
## [1] 120.5011
```

RMSE of MLR:

```
sqrt(sum(complex_model$residuals^2)/nrow(city_hotels_removed)) #RMSE of MLR model
```

```
## [1] 23.42904
```

SD of ADR:

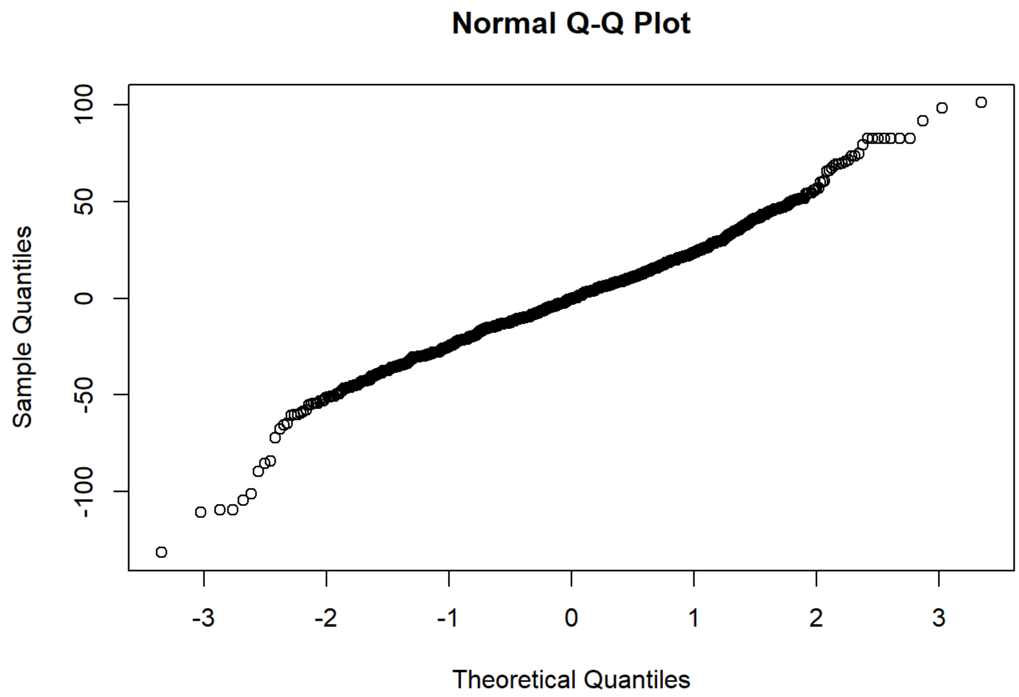
```
sd(test$adr)
```

```
## [1] 44.01975
```

Model Diagnostics:

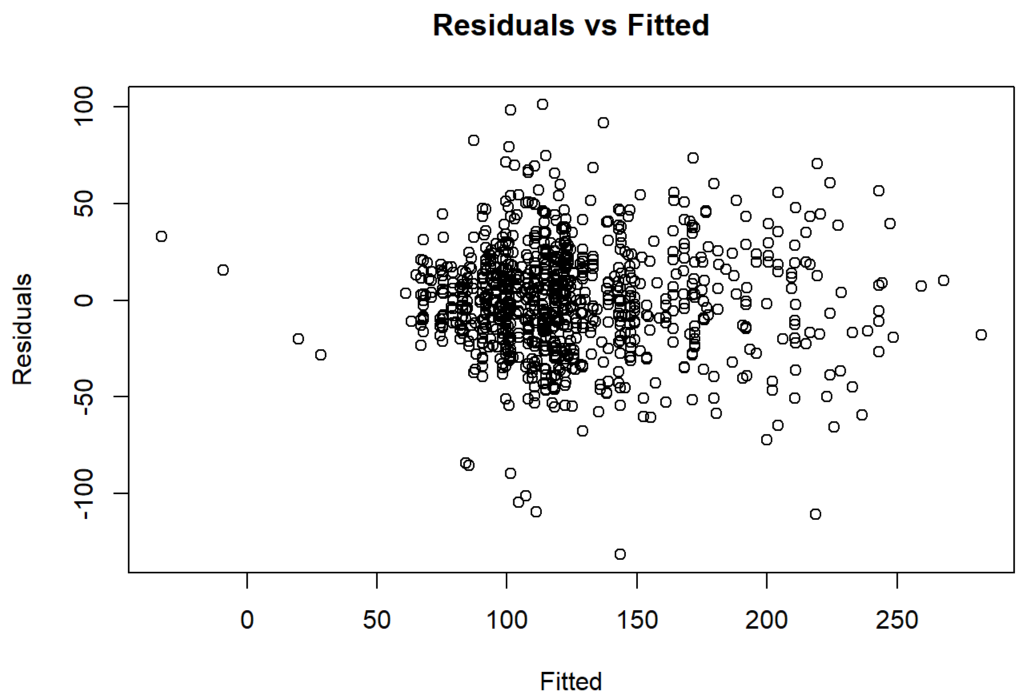
QQ plot

```
qqnorm(complex_model$residuals) #qq plot to test for normality
```



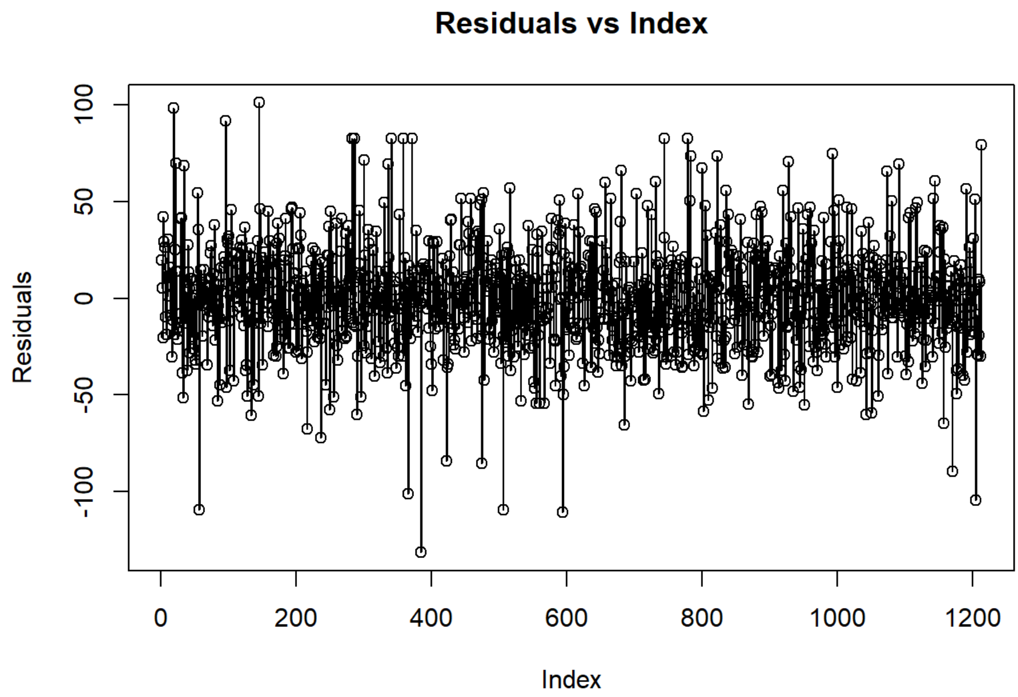

Residual vs Fitted

```
#plot for testing for constance variance assumption
plot(complex_model$fitted.values, complex_model$residuals, xlab = "Fitted", ylab = "Residuals", main = "Resi
duals vs Fitted")
```




Residual vs index

```
#plot to help test for correlated errors
len = 1:1213 #index
plot(len, complex_model$residuals, xlab = "Index", ylab = "Residuals", main = "Residuals vs Index")
lines(len, complex_model$residuals)
```




durbin watson test

```
dwtest(complex_model) #correlated errors test
```

```
##
##  Durbin-Watson test
##
## data:  complex_model
## DW = 2.0429, p-value = 0.7605
## alternative hypothesis: true autocorrelation is greater than 0
```