Restaurants Visitors Forecasting

Load in packages and datasets

```
library(ggplot2)
library(readr)
library(knitr)
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

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

library(fpp)

## Loading required package: forecast

##
## Attaching package: 'forecast'

## The following object is masked from 'package:ggplot2':
##
##     autolayer

## Loading required package: fma

## Loading required package: expsmooth

## Loading required package: lmtest

## Loading required package: zoo

##
## Attaching package: 'zoo'

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

## Loading required package: tseries

df_air <- read_csv('~/Desktop/air_visit_data.csv')
```

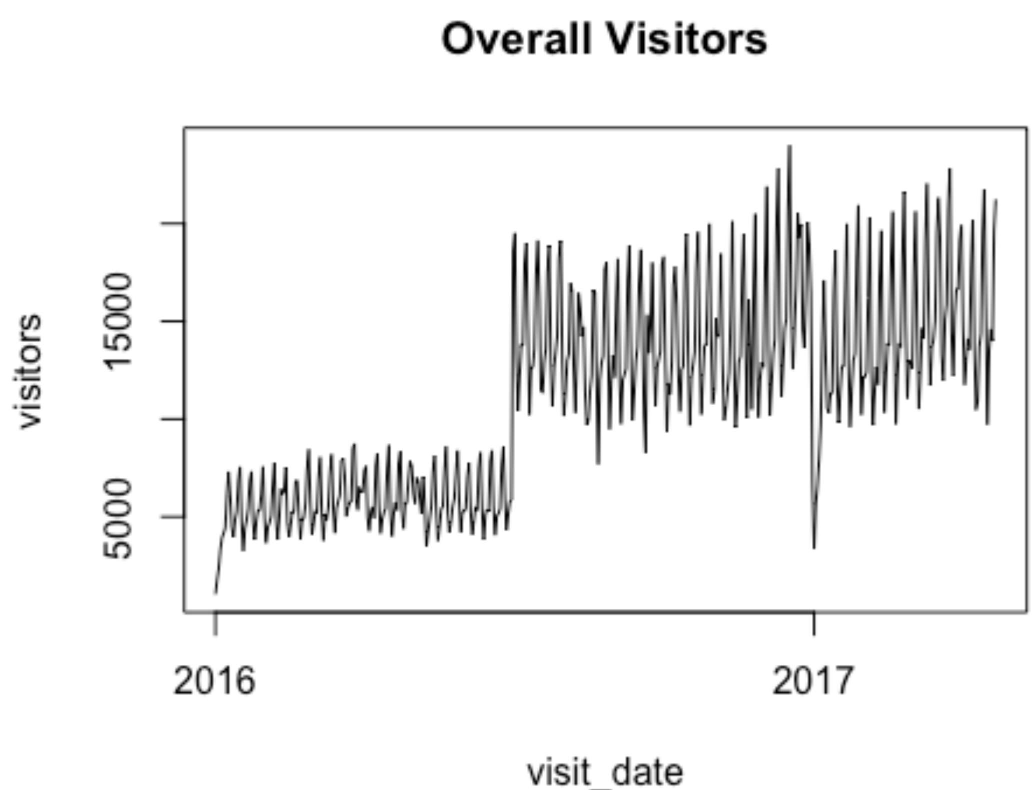
```
## Parsed with column specification:
## cols(
##   air_store_id = col_character(),
##   visit_date = col_date(format = ""),
##   visitors = col_double()
## )
```

```
df_air_store <- read_csv('~/Desktop/air_store_info.csv')
```

```
## Parsed with column specification:
## cols(
##   air_store_id = col_character(),
##   air_genre_name = col_character(),
##   air_area_name = col_character(),
##   latitude = col_double(),
##   longitude = col_double()
## )
```

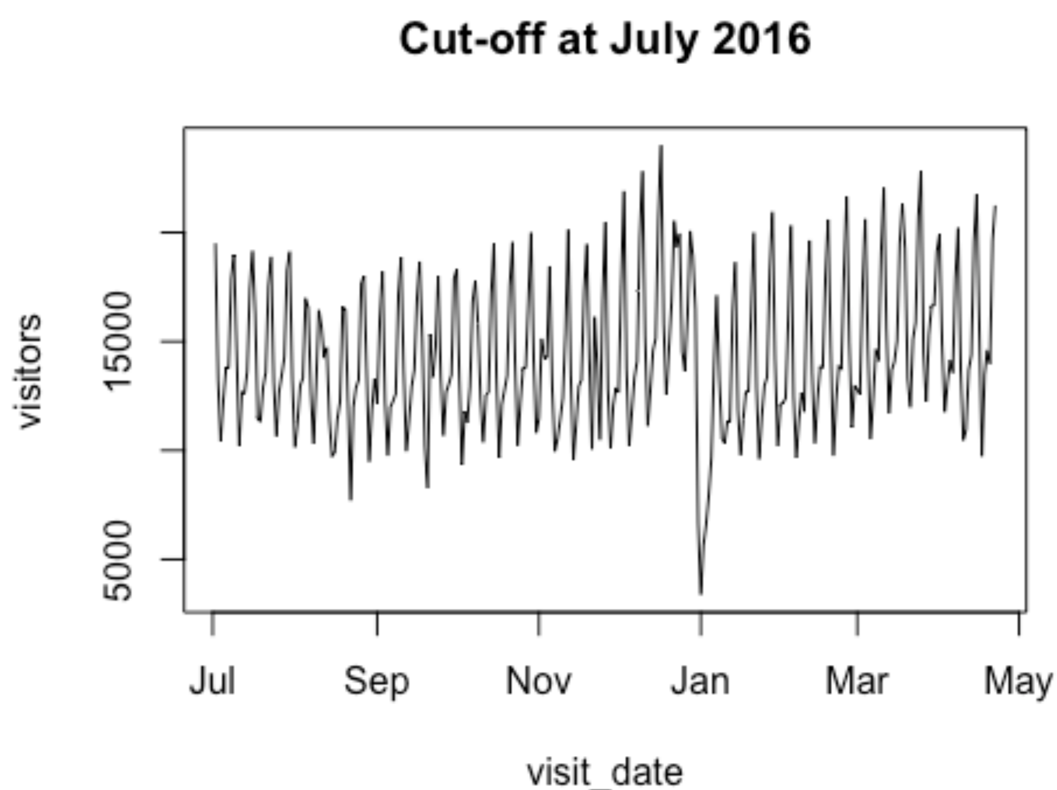
Plot Overall Visitor Distribution

```
df_air %>%
  group_by(visit_date) %>%
  summarize(visitors = sum(visitors)) %>%
  plot(type='l', main='Overall Visitors')
```



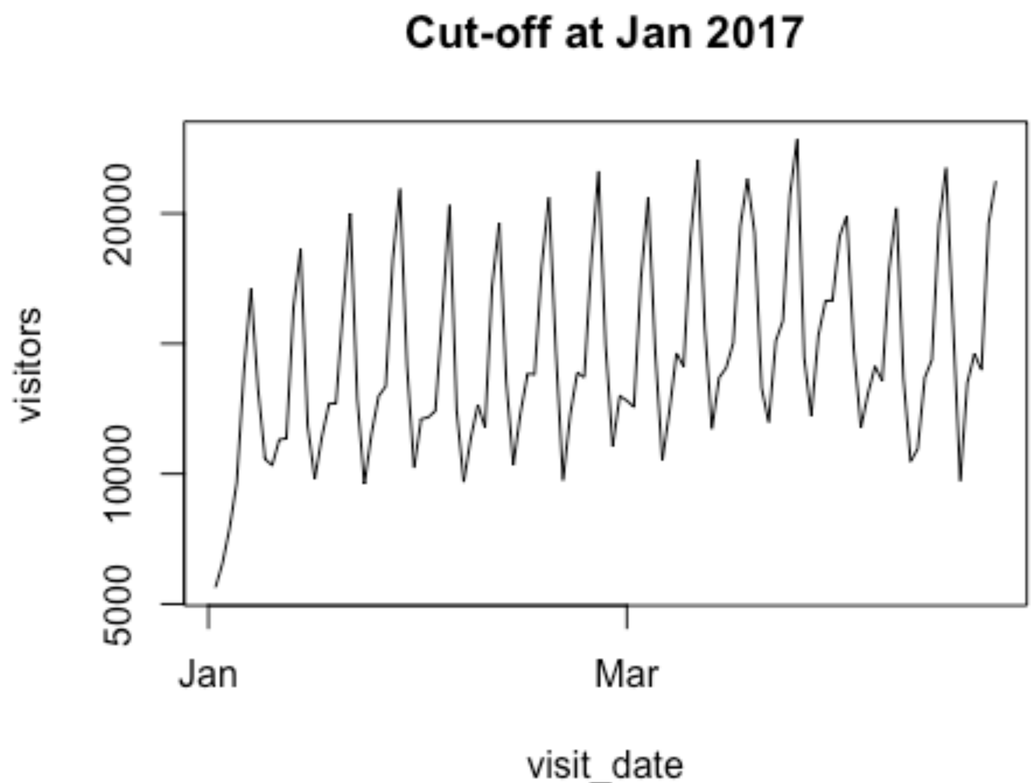
Plot Visitor Distribution After 2016-07-01

```
merged <- df_air %>% filter(visit_date > '2016-07-01') %>% dplyr::left_join(d
f_air_store, by='air_store_id', how='left')
merged_sum <- merged %>% group_by(visit_date) %>% summarize(visitors = sum(vi
sitors))
merged_sum %>% plot(type='l', main='Cut-off at July 2016')
```



Plot Visitor Distribution After 2017-01-01

```
merged1 <- df_air %>% filter(visit_date > '2017-01-01') %>% dplyr::left_join(
df_air_store, by='air_store_id', how='left')
merged_sum1 <- merged1 %>% group_by(visit_date) %>% summarize(visitors = sum(
visitors))
merged_sum1 %>% plot(type='l', main='Cut-off at Jan 2017')
```

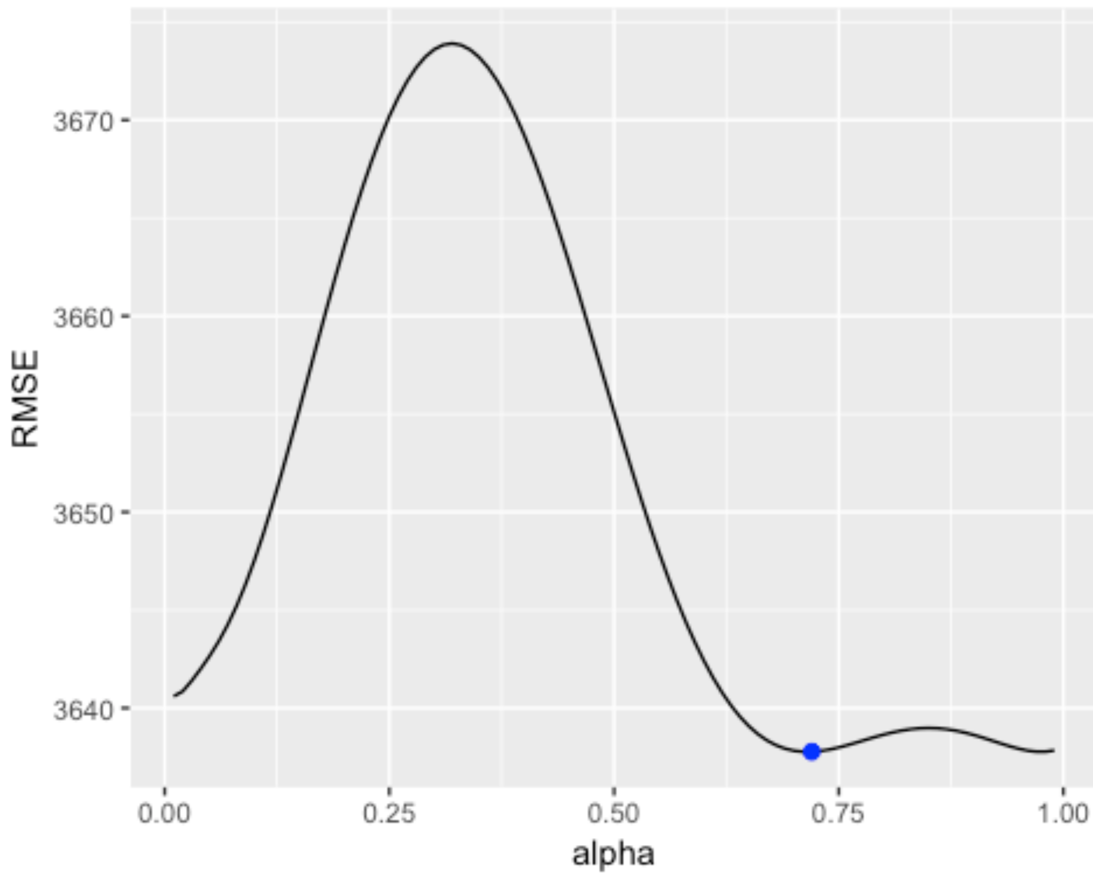


Simple Exponential Smoothing

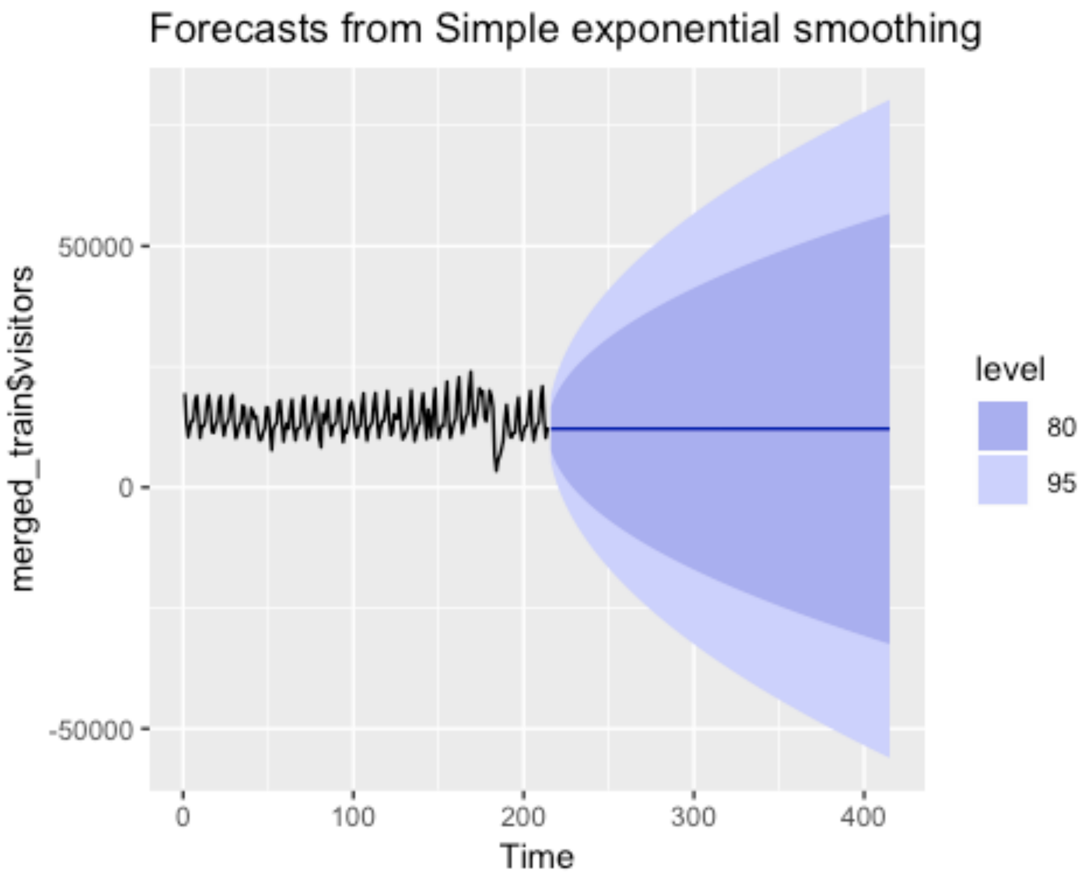
Cross-Validation

```
# train-test split
merged_train <- merged_sum %>% filter(visit_date <='2017-02-01')
merged_test <- merged_sum %>% filter(visit_date >'2017-02-01')
# the change in the number of visitors from the previous day
merged_dif_test <- diff(merged_test$visitors)
merged_dif <- diff(merged_train$visitors)
# identify optimal alpha parameter
alpha <- seq(.01, .99, by = .01)
RMSE <- NA
for(i in seq_along(alpha)) {
  fit <- ses(merged_dif, alpha = alpha[i], h = 100)
  RMSE[i] <- accuracy(fit, merged_dif_test)[2,2]
}

# convert to a data frame and idenitify min alpha value
alpha.fit <- data_frame(alpha, RMSE)
alpha.min <- filter(alpha.fit, RMSE == min(RMSE))

# plot RMSE vs. alpha
```

```r
ggplot(alpha.fit, aes(alpha, RMSE)) +
  geom_line() +
  geom_point(data = alpha.min, aes(alpha, RMSE), size = 2, color = "blue")
```



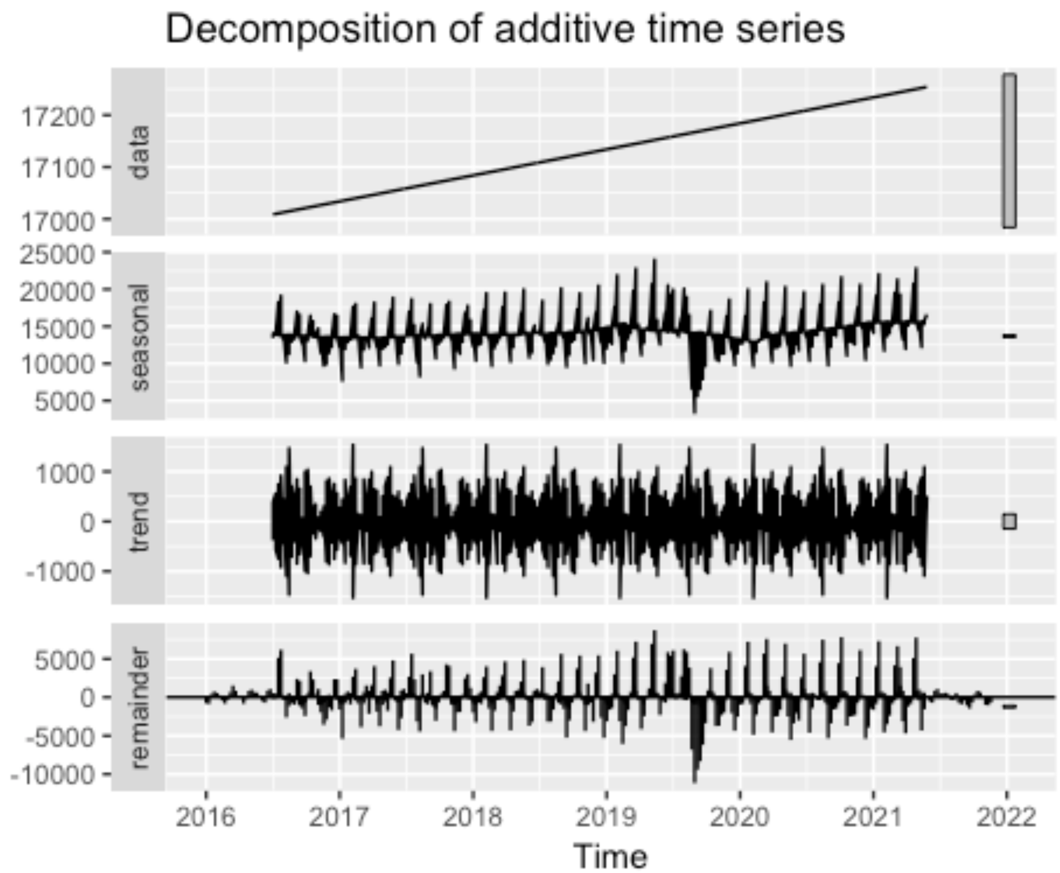

```r
# fit
ses_fit <- ses(merged_train$visitors, alpha = .7, h = 200)
autoplot(ses_fit)
```




```
# performance eval
accuracy(ses_fit, merged_dif_test)

##                       ME       RMSE        MAE          MPE       MAPE
## Training set    -35.14735   3505.141   2880.623    -4.909985   22.23778
## Test set     -12021.25878 12559.618 12021.259 -7921.567184 8862.51073
##                  MASE      ACF1
## Training set 1.045364 0.2773214
## Test set     4.362457        NA

# plotting results
p1 <- autoplot(ses_fit) +
  theme(legend.position = "bottom")
```

Check Seasonality

```
timeseries <- ts(merged_sum, frequency = 50, start = c(2016,1))
autoplot(decompose(timeseries))
```

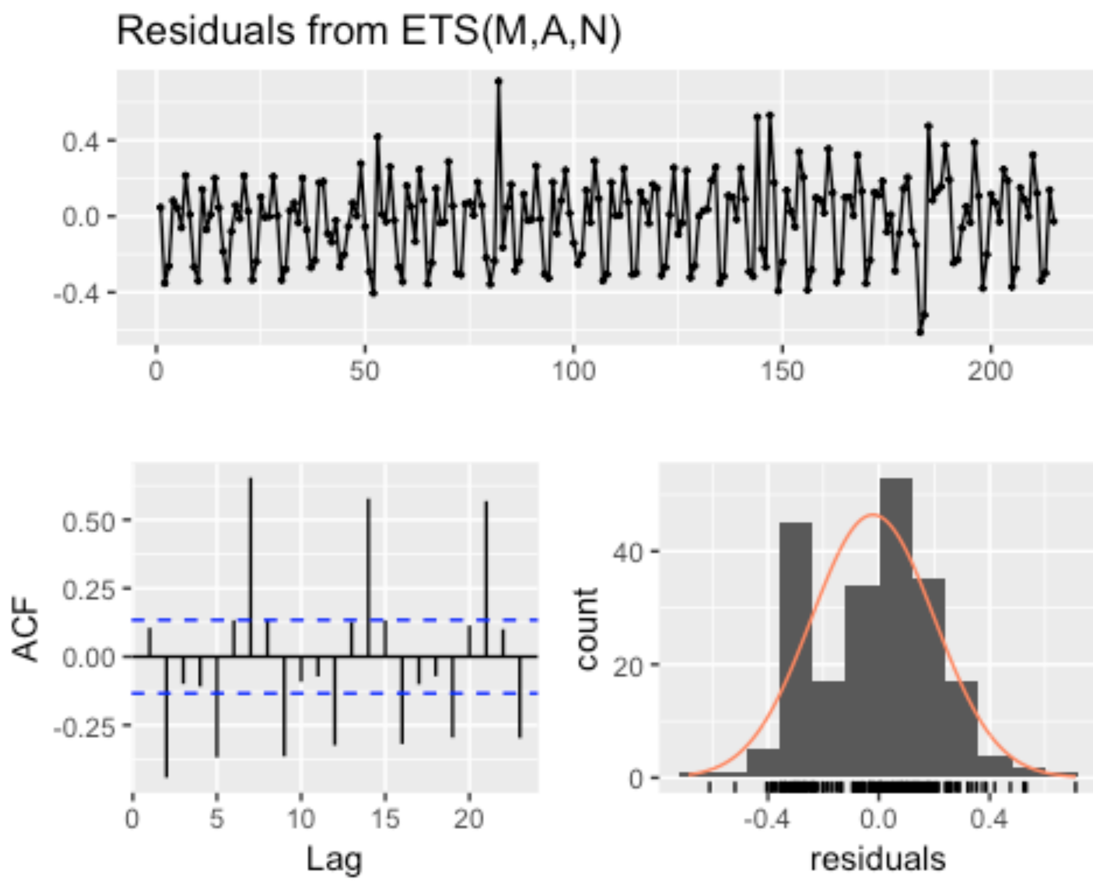



```r
# Detect Seasonaity & Which Model to Use (No Seasonality)
ets(merged_train$visitors)
```

```
## ETS(M,A,N)
##
## Call:
##  ets(y = merged_train$visitors)
##
##   Smoothing parameters:
##     alpha = 0.9999
##     beta  = 0.0038
##
##   Initial states:
##     l = 15761.0994
##     b = 721.3255
##
##   sigma:  0.2253
##
##      AIC     AICc      BIC
## 4631.119 4631.406 4647.972
```

Multiplicative Holt-Winters Non-Seasonal Model

```
# fit
merged_hw <- ets(merged_train$visitors, model='MAN',alpha = 0.9999, beta = 0.
0038)
merged_f <- forecast(merged_hw, h = 100)
# performance check
checkresiduals(merged_hw)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,A,N)
## Q* = 216.14, df = 6, p-value < 2.2e-16
##
## Model df: 4.   Total lags used: 10
```

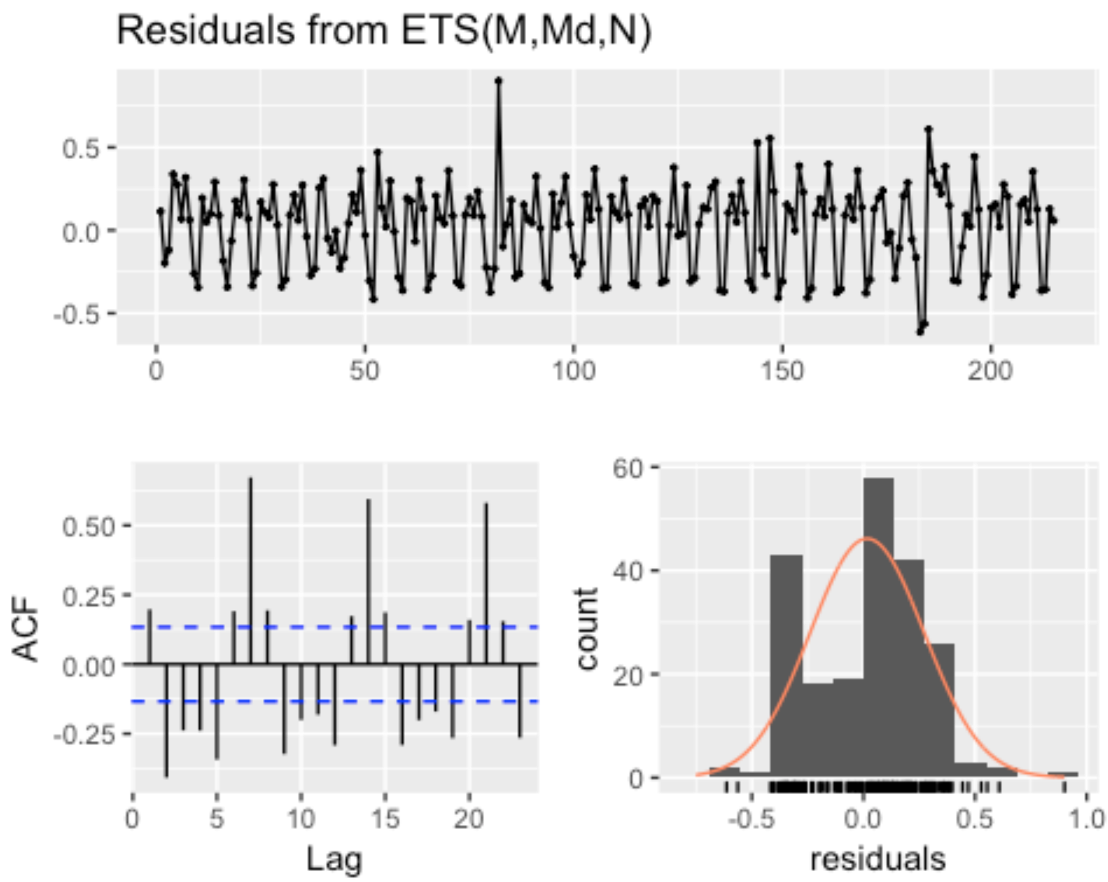
```
accuracy(merged_f, merged_dif_test)
```

```
##                       ME       RMSE       MAE           MPE        MAPE
## Training set   -664.2893  3443.075  2674.522     -8.152387    20.85765
## Test set     -28246.7956 29916.575 28246.796 -12351.149546 14261.14502
##                     MASE      ACF1
```

```
## Training set  0.9705712 0.1396123
## Test set     10.2506253        NA
```

Damping Method

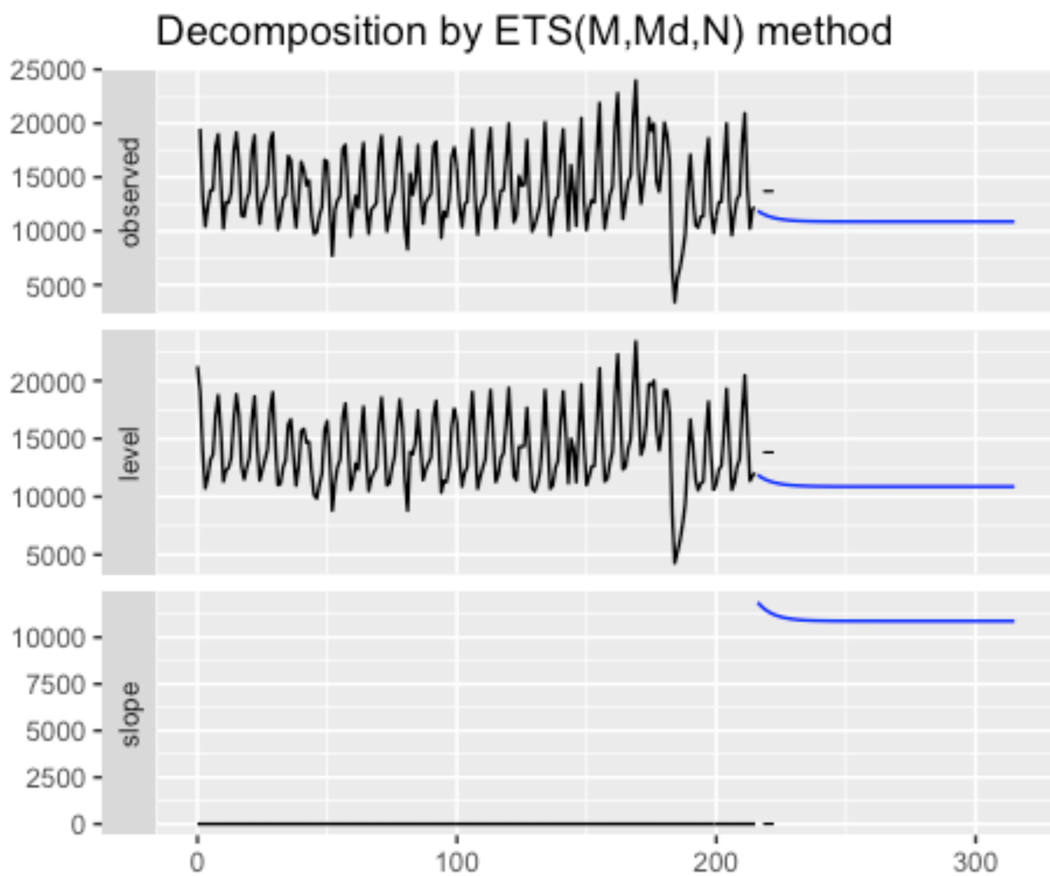
```
# fit
Damp_fit <- ets(merged_train$visitors, model = "ZMN", damped = TRUE, alpha =
0.8, beta = 0.2, phi = 0.85)
Damp_pred <- forecast(Damp_fit, h = 100)
# performance eval
checkresiduals(Damp_fit)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,Md,N)
## Q* = 247.39, df = 5, p-value < 2.2e-16
##
## Model df: 5.   Total lags used: 10

accuracy(Damp_pred, merged_test$visitors)

##                     ME     RMSE      MAE       MPE    MAPE     MASE
## Training set -310.9739 3805.471 3102.982 -5.876239 24.1762 1.126057
```

```
## Test set     4332.9675 5559.939 4469.172 24.662781 26.0386 1.621841
##                   ACF1
## Training set 0.2302951
## Test set            NA
```

```
# plot the result
autoplot(Damp_fit) +
  theme(legend.position = "bottom") + autolayer(Damp_pred$mean, color = "blue
")
```



####

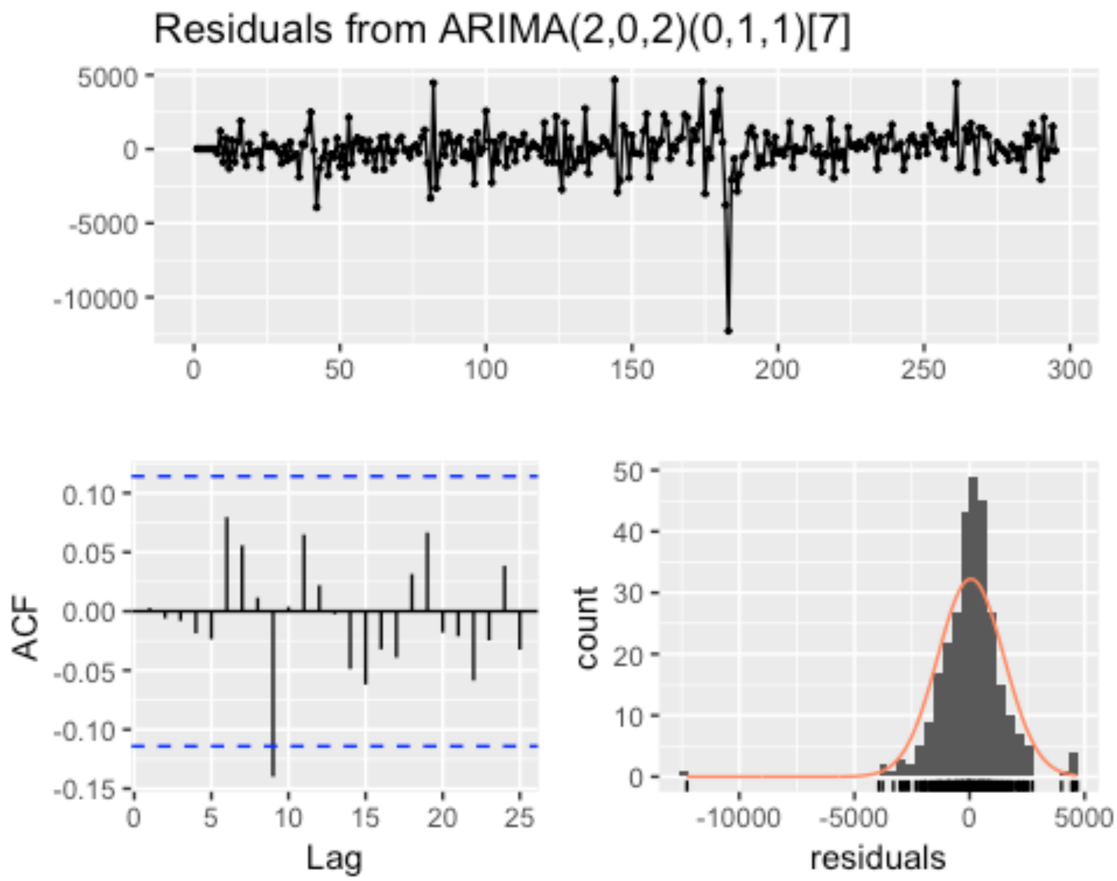
ARIMA Model

```
# p d q tuning
auto.arima(merged_train[,2])
```

```
## Series: merged_train[, 2]
## ARIMA(2,0,2) with non-zero mean
##
## Coefficients:
##          ar1      ar2     ma1      ma2        mean
##       0.7335  -0.3766  0.1782  -0.2820  13994.9983
## s.e.  0.1275   0.0741  0.1279   0.0978    227.5413
##
```

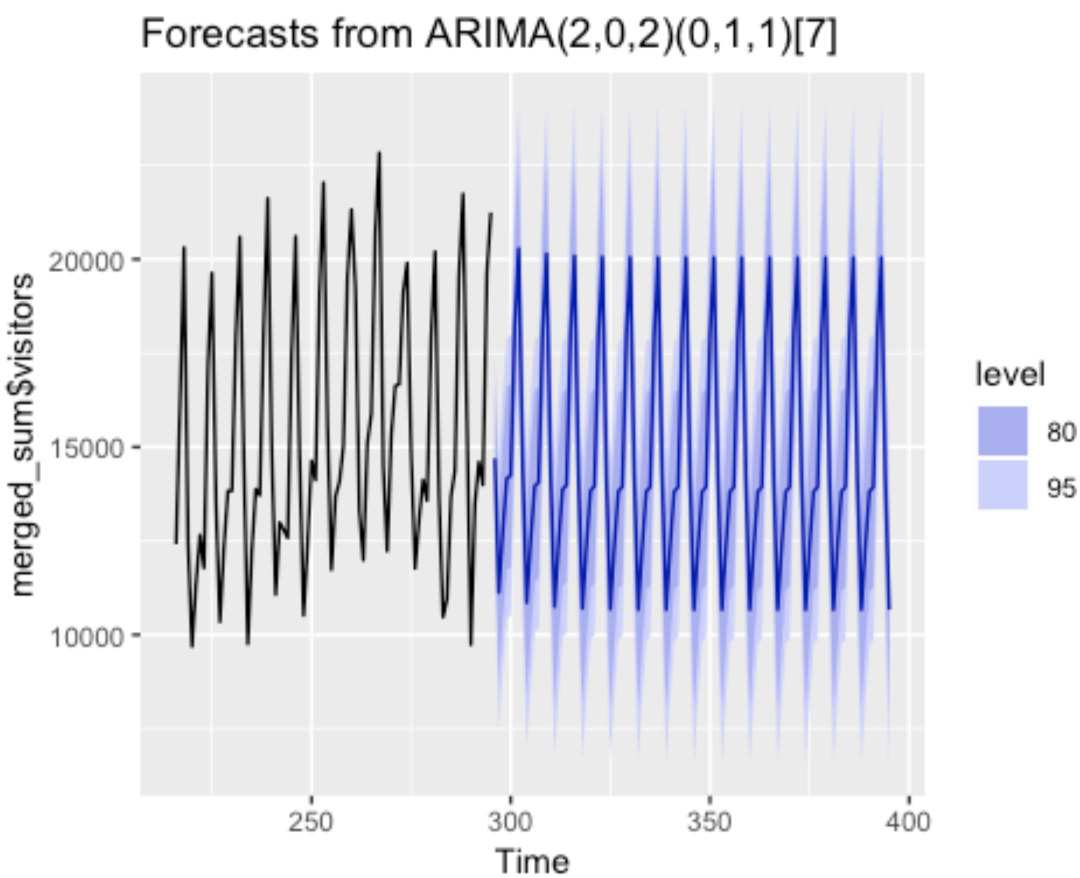
```
## sigma^2 estimated as 5844544:  log likelihood=-1978.12
## AIC=3968.24   AICc=3968.64   BIC=3988.46
```

```r
# fit model
m <- arima(merged_sum$visitors, order=c(2,0,2), seasonal= list(order=c(0,1,1)
, period=7))
checkresiduals(m)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,0,2)(0,1,1)[7]
## Q* = 9.2188, df = 5, p-value = 0.1006
##
## Model df: 5.   Total lags used: 10
```

```r
# plot the result
m %>% forecast(h=100) %>% autoplot(include=80)
```

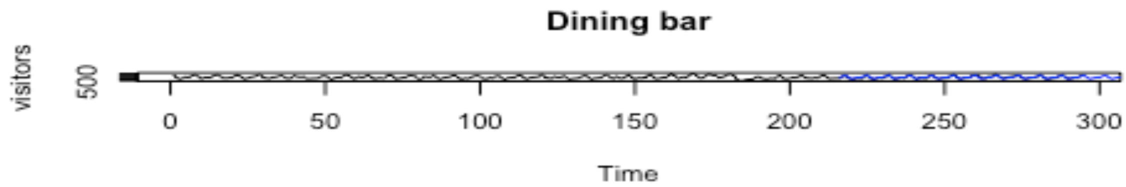


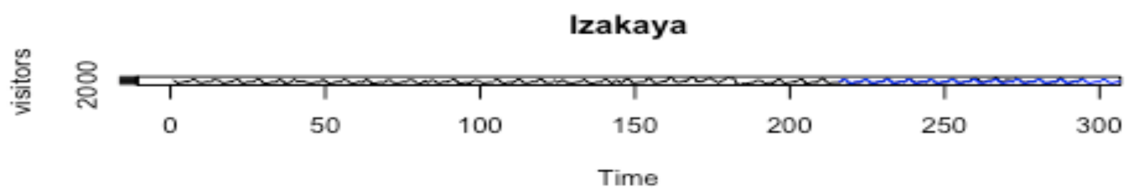
####

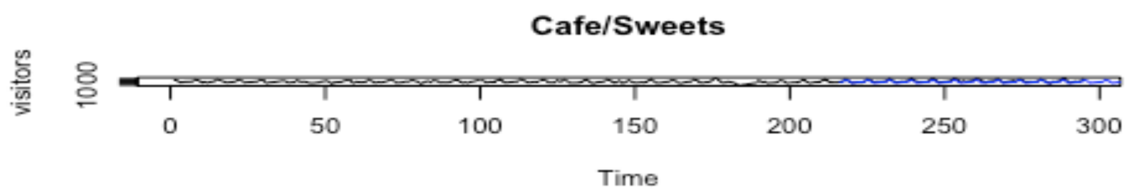
Plot Forcasting With Genres

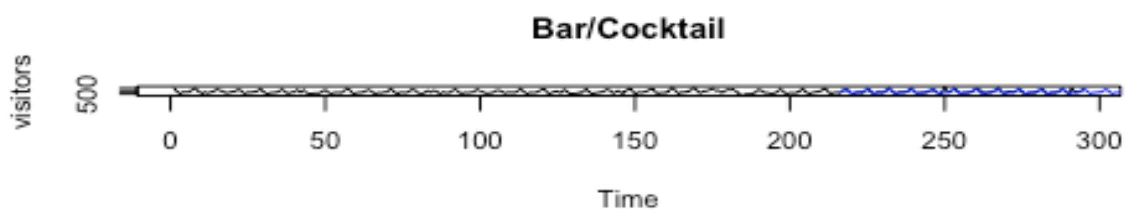
```
# transform
genre_sum <- merged %>%
  group_by(visit_date, air_genre_name) %>%
  summarize(visitors=sum(visitors))
genre_unique <- unique(merged$air_genre_name) %>% unlist
# plot the forcasting
plot_genre <- function(i){
  genre_specific_sum <- genre_sum %>% filter(air_genre_name==i)
  genre_train <- genre_specific_sum %>% filter(visit_date <='2017-02-01')
  genre_test <- genre_specific_sum %>% filter(visit_date >'2017-02-01')

  m <- arima(genre_train$visitors, order=c(2,0,2), seasonal= list(order=c(1,1
,1), period=7))
  y_pred <- forecast::forecast(m, h=100)

  plot(ts(genre_specific_sum$visitors), main=i,  ylab='visitors')
  lines(y_pred$mean, col='blue')
}

par(mfrow=c(3,1), cex=0.7)
```

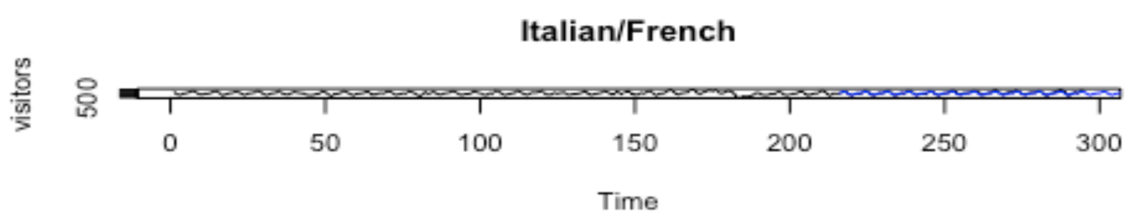
```r
for( i in genre_unique){
plot_genre(i)}
```

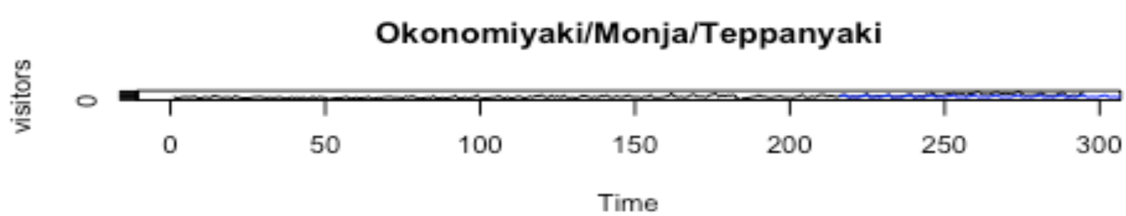


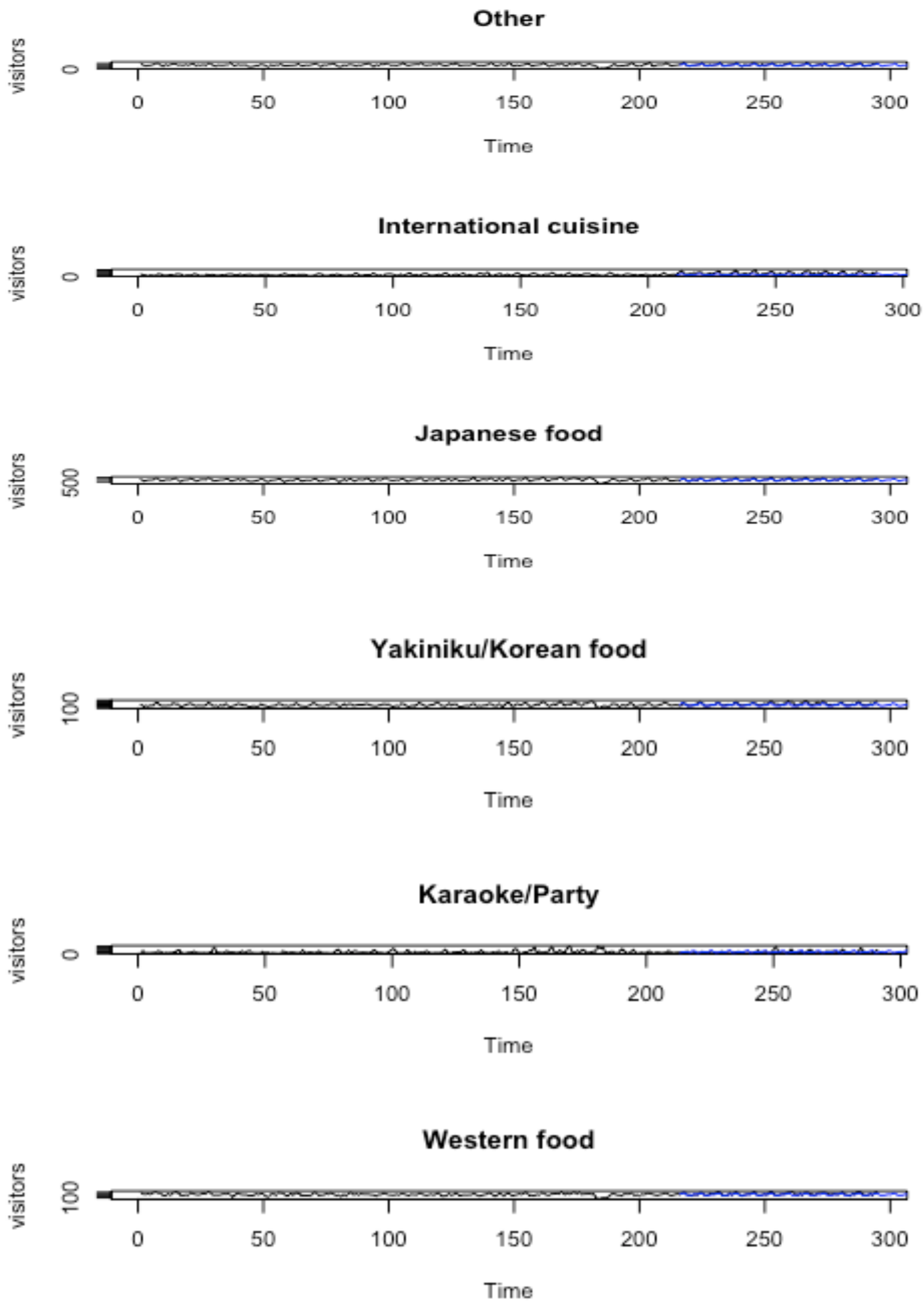


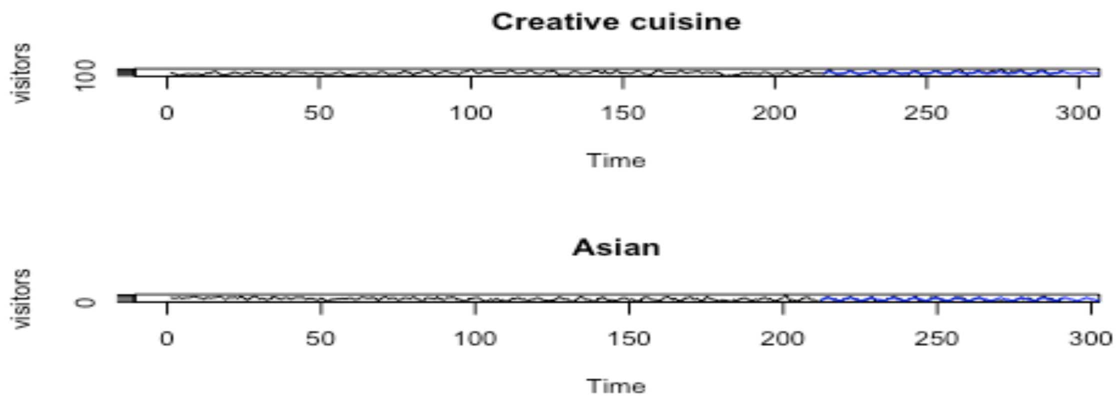






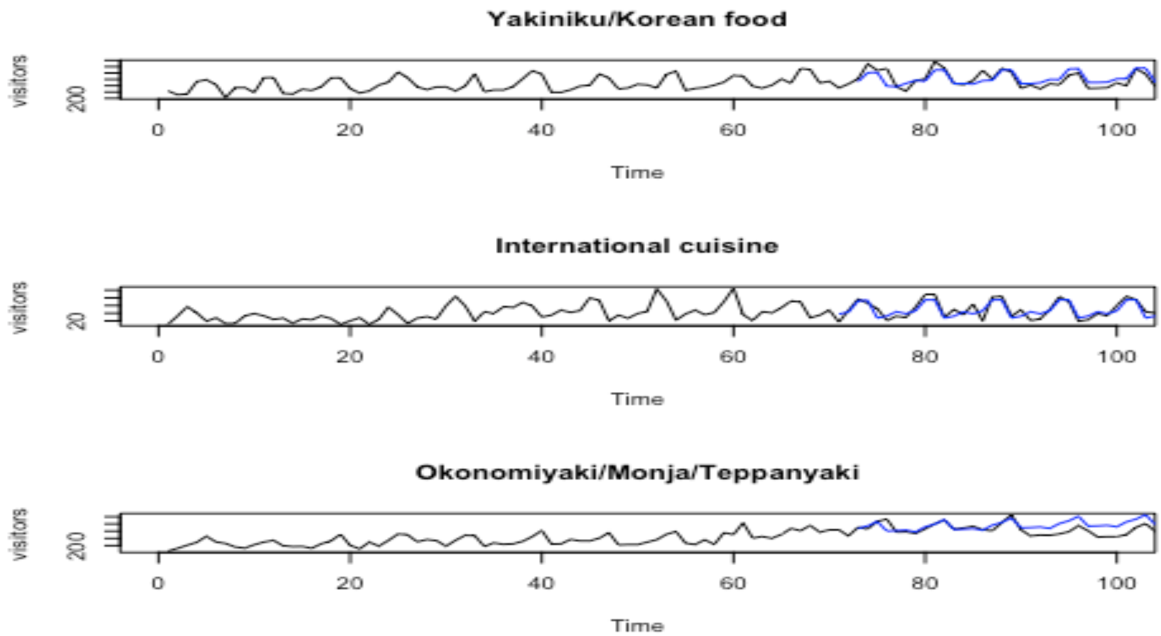



Other
visitors
0
0
50
100
150
200
250
300
Time
International cuisine
visitors
0
0
50
100
150
200
250
300
Time
Japanese food
visitors
500
0
50
100
150
200
250
300
Time
Yakiniku/Korean food
visitors
100
0
50
100
150
200
250
300
Time
Karaoke/Party
visitors
0
0
50
100
150
200
250
300
Time
Western food
visitors
100
0
50
100
150
200
250
300
Time



Adjustment for Three Genres

```
# adjustment
par(mfrow=c(3,1),cex=0.6)
for (i in c("Yakiniku/Korean food","International cuisine", "Okonomiyaki/Monj
a/Teppanyaki") ){
  genre_specific_sum <- genre_sum %>% filter(air_genre_name==i) %>% filter(vi
sit_date > '2017-01-02')
  split_date <- '2017-03-15'
  genre_train <- genre_specific_sum %>% filter(visit_date <= split_date)
  genre_test <- genre_specific_sum %>% filter(visit_date > split_date)

  m <- arima(genre_train$visitors, order=c(2,1,2), seasonal= list(order=c(1,1
,1), period=7))
  y_pred <- forecast::forecast(m, h=100)

  plot(ts(genre_specific_sum$visitors), main=i,ylab='visitors', xlim=c(0,100)
)
  lines(y_pred$mean, col='blue')
}
```



ARIMA Model has the best performance