STAT 443 – Term Project Report

Akbar Ali Qazi, Ash Sandhu, and Mark Kopani

Summary

Airbnb is a technology company that is part of the fast growing 'sharing economy', where it acts as an ebroker, allowing anyone to rent out their property or a spare room in their house to guests, using their web or phone apps. They collect a small commision from each booking and they operate globally with over $2.6 billion in annual revenue. The goal of our term project was to analyze Airbnb listing data to better understand the trends in pricing. The data is available for many cities, however we chose to focus on the pricing of properties in Metro Vancouver. A better understanding of the trends is of great importance to users who have to decide how to price their listing, thus understanding the weekly, monthly and seasonal trends would allow users to determine their earning potential during certain periods. It would also allow guests to book at a time that is priced ideally for them.

Due to the large amount of data available for a given listing, such as the amenities present, the location etc, we explored any leading indicators that impact the price of the listing. We will also look at how the price of similar of listings change overtime, to see if there is any underlying seasonality and what the general pricing trends are. Finally, with a thorough understanding of the underlying trends, we provided a forecast for what the future prices will be for the current year and up to 2021.

The Data & Initial Analysis

The raw data that used in our analysis comes from insideairbnb.com, where it is available as several different .csv files. After obtaining the data we inspected, cleaned and amalgamated it to produce a dataframe with the pertinent variables needed for the analysis. This was followed by visualizing the data via plots. Before performing any significant analysis, the raw price data was plotted through time in order to:

- Notice any monthly or annual swings in the series,
- Notice any trend in the change of mean, and to
- Notice any potential outliers due to confounding variables.

Variables

Response variable: Listing price

Explanatory variables for ARIMAX (as chosen by LASSO):
- Cleaning fee
- Review score at check in
- Review score post-stay and its accuracy
- The number of listings the host has
- Year-round availability of the accommodation
- The amount of extra people staying
- The number of guests that can be accommodated
- The maximum number of nights it can be booked for

Programming Languages

The majority of the analysis will be conducted using Jupyter Notebooks which are able to run both R and Python. This will allow us to clean the data using python and producing data frames which can then be analysed using R and the appropriate methods we have been taught in class.

Project Structure and Feasibility

The project will combine knowledge from several different statistics courses to produce time series data that we will subsequently analyse. Due to the data being so readily available, the project is very feasible with the biggest hurdle being the data cleaning process. The analysis of the time series data will look at and will not be limited to checking for seasonality in the data, identifying the underlying trends and modeling them using appropriate methods discussed in class. The accuracy of the different models at predicting future prices will be determined using the classical RMSE methods discussed using hold out data.

Discussion

The time series of the Airbnb data was plotted to see how the price fluctuates with time as seen in **Fig. 1**. From the plot we can see that there is a substantial amount of noise in the time series thus the data was grouped by the weekly mean price, which would act as a filter to reduce noise as seen in **Fig. 2**.

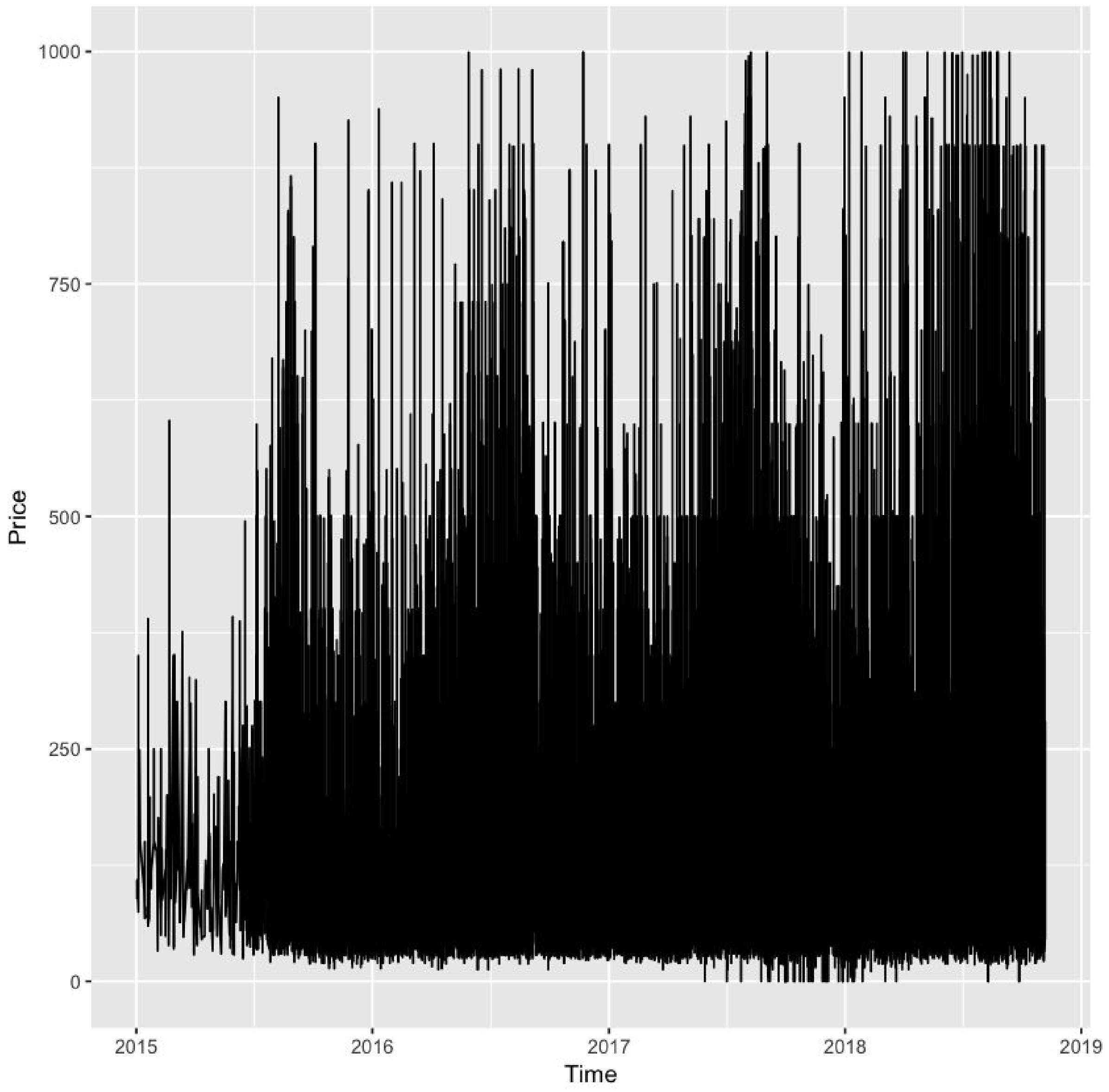

Fig. 1: *Time series plot for price for the Airbnb dataset.*

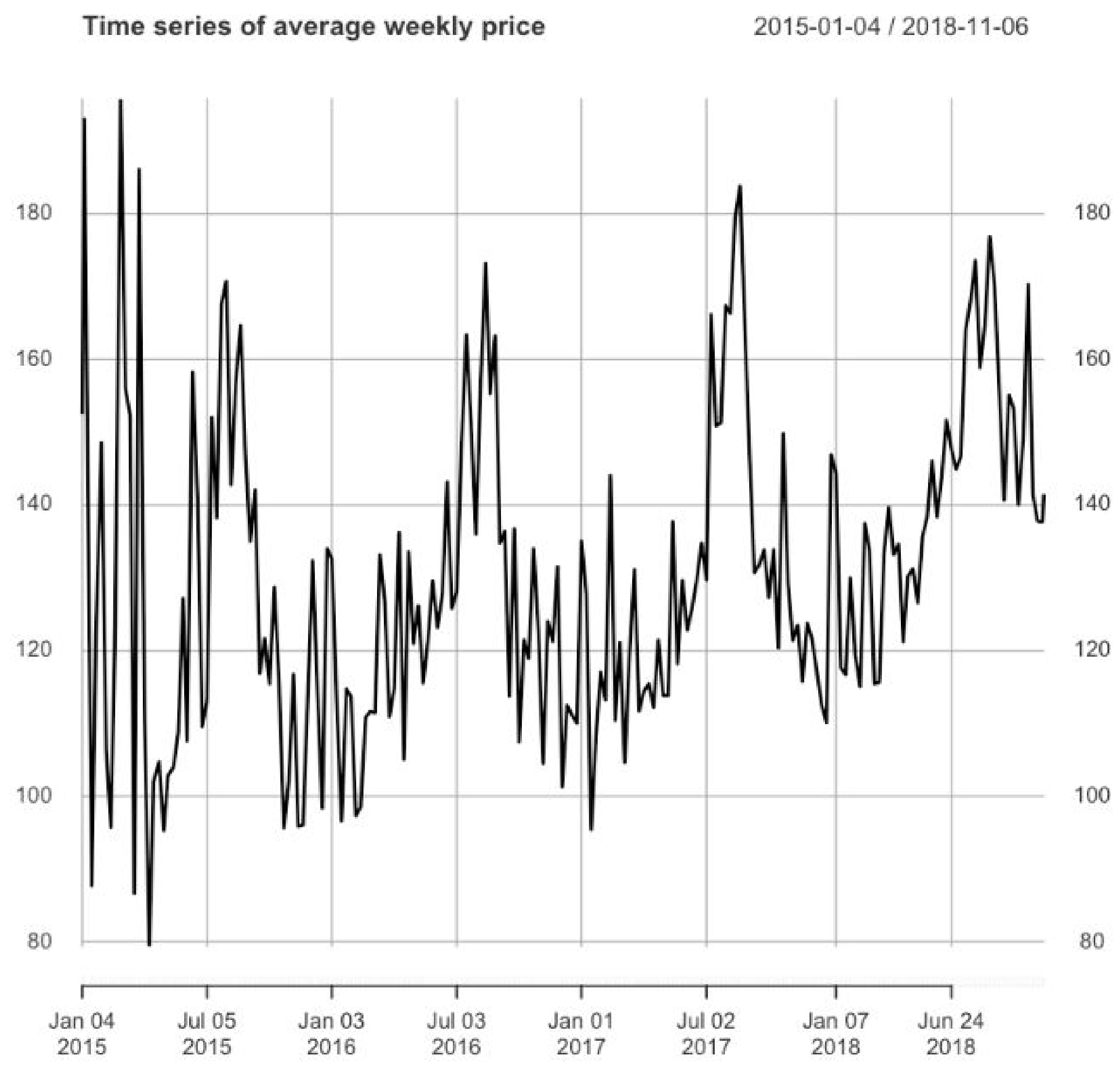

Fig. 2: *Time series plot for weekly average price for the Airbnb dataset.*

From **Fig. 2** we can see that there is a seasonal trend with surges in price in the summer months, which may be explained by increased number of tourists visiting the city during that period. The start of the time series also shows a greater variation in price than the following months, this may be explained by the fact that this was the time when Airbnb was launched in Vancouver and people did not know how to price their homes in the market initially, but this variation seems to stabilize as time progresses with the market adjusting to the demand of the houses.

Since there is a seasonal trend the in the time series, it was decomposed into its constituent elements: seasonality, trend and random noise.

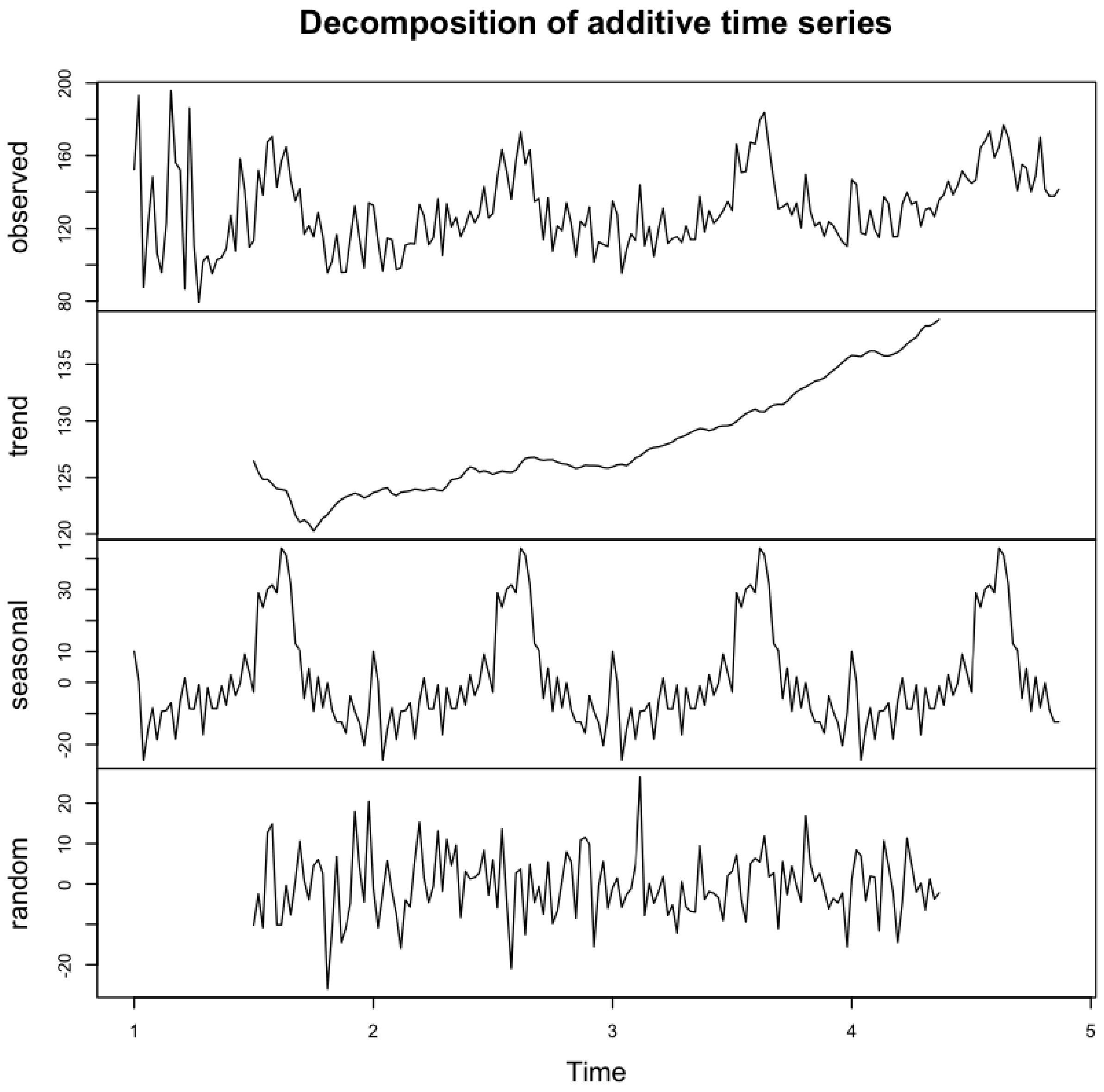

Fig 3: *Plots for time series decomposition.*

The plots from **Fig. 3** show the decomposed time series data. We can see that is a seasonality with price spikes in the summer. The trend also shows us that the prices initially start off quite high, followed by a drop and then a resurgence in price, possibly due to increased tourism and increased market activity. The remainder also shows that there is a lot of noise at the start, again due to uncertainty about to price a house. This slowly gets better with less variation, due to prices settling down.

Before trying to fit an ARIMA model, a Holt-Winter model was fit. This is a simple seasonal exponential smoothing model that was used as a baseline, and every other model built was compared to this.

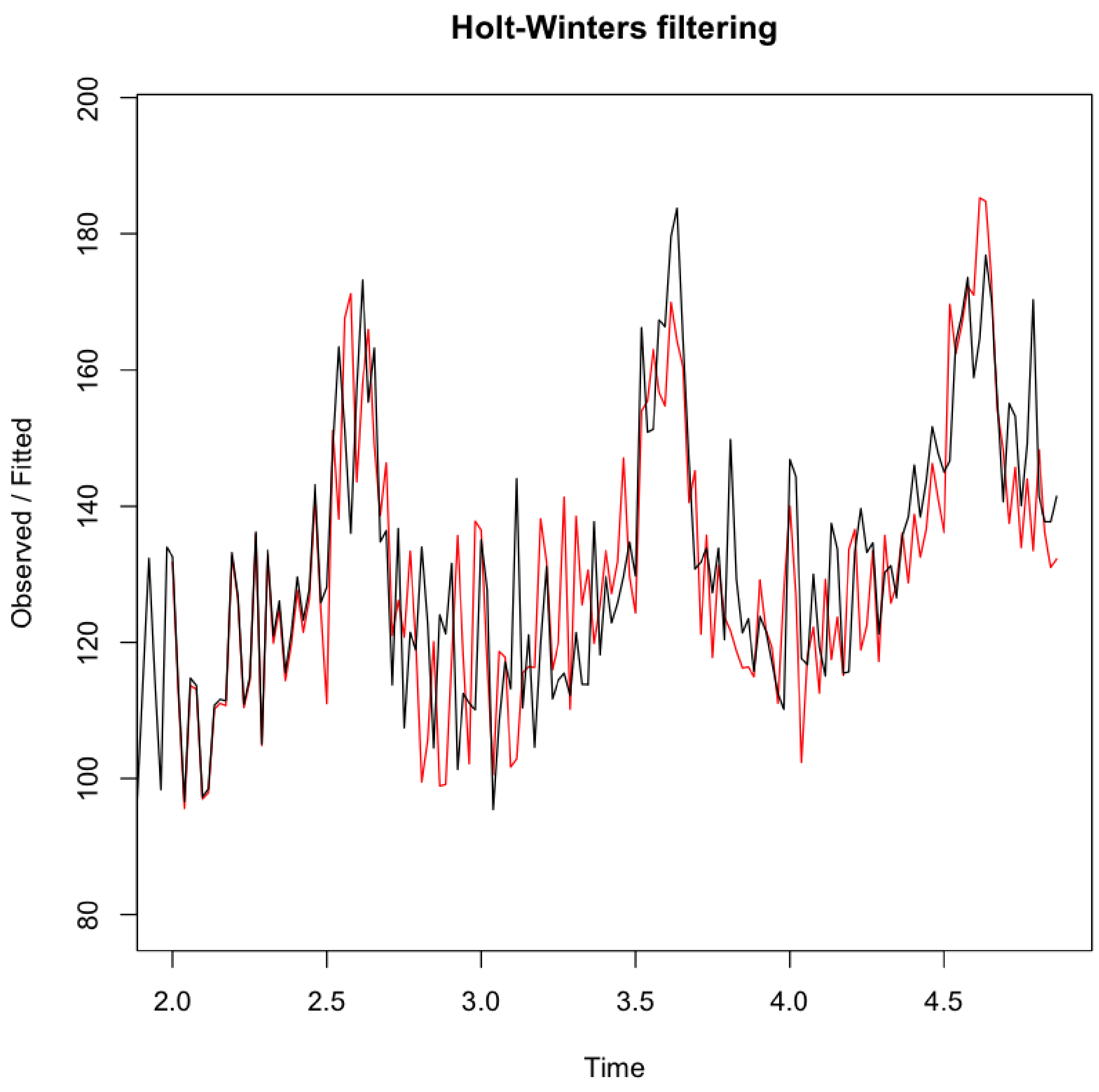

Fig. 4: *Holt- Winters seasonal exponential smoothing model (red) superimposed on weekly average price time series data.*

From **Fig. 4** we can see that the predictions made by the exponential smoothing model fit well with the observed data. The RMSE was also calculated for Holt-Winters model and from **Table**

1 we can see that the RMSE values for the training set and the test set are very similar, indicating that the model has a good fit.

	RMSE
Training Set	12.11623
Test Set	13.26653

Table 1: *Calculated RMSE values for the Holt-Winters model.*

An ARIMA model was then fit. To fit an ARIMA model we have to assume that the time series is stationary, but from visual inspection we can see that the time series may not be stationary. To confirm stationarity a variogram was constructed.

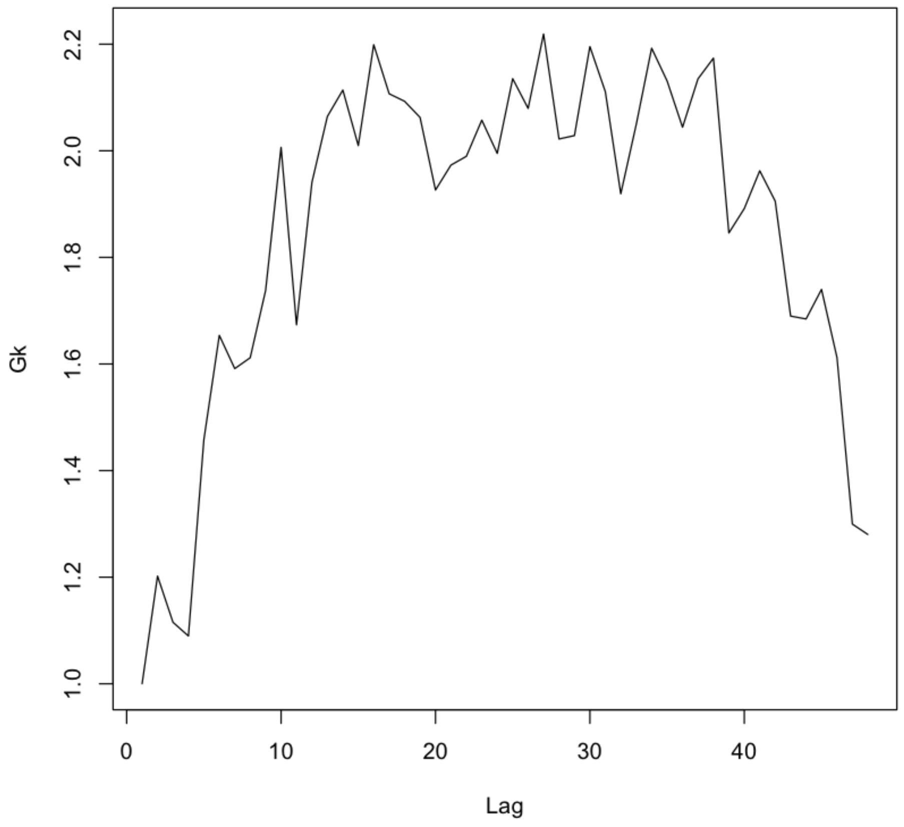

Fig. 5: *Variogram for the time series data.*

The Variogram (**Fig. 5**) suggests that our time series is not stationary. A variogram for a stationary time series should level out, but the above Variogram shows a quadratic trend, suggesting that a simple differencing by 1 or 2 steps is needed.

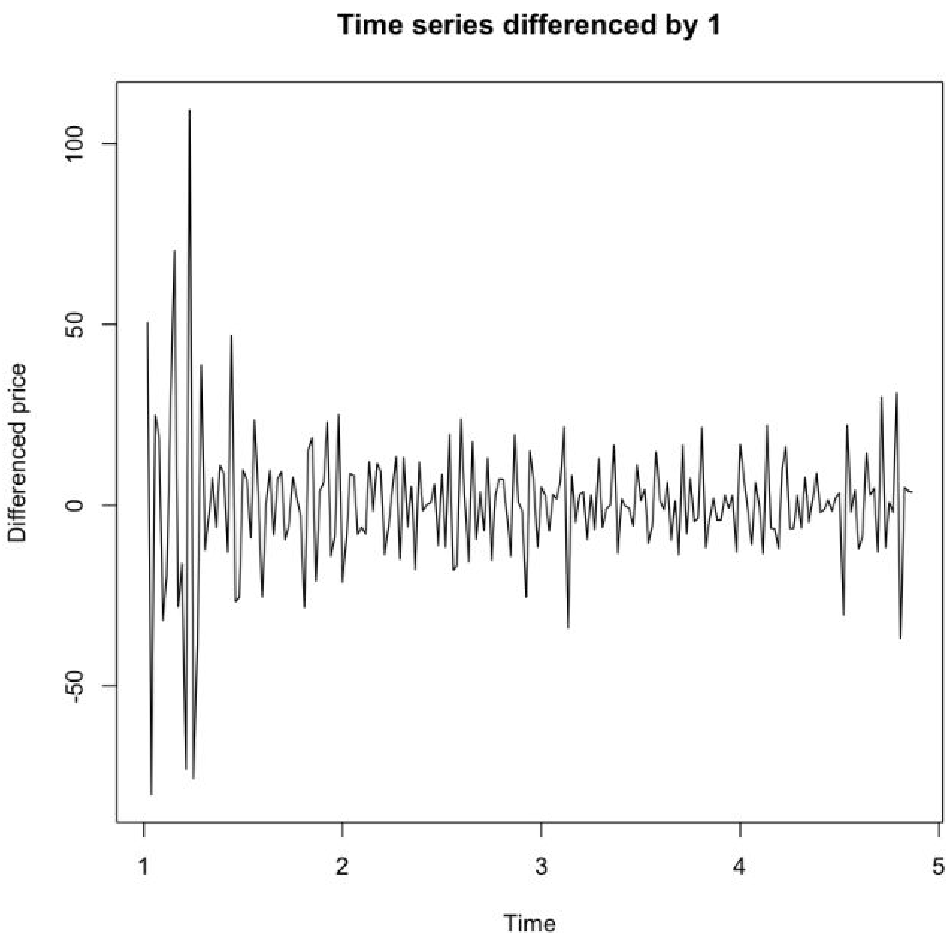

Fig. 6: *1-step differenced time series.*

The time series was differenced by 1-step as seen in **Fig. 6**. The new differenced time series appears to be stationary thus we assume d = 1 in our model. To determine the other parameters the ACF and PACF were plotted.

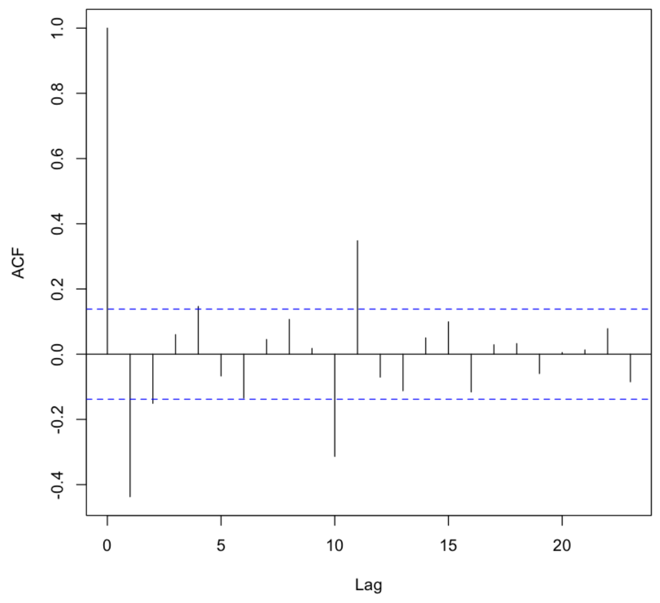

Fig. 7: *ACF plot for the 1-step differenced time series data.*

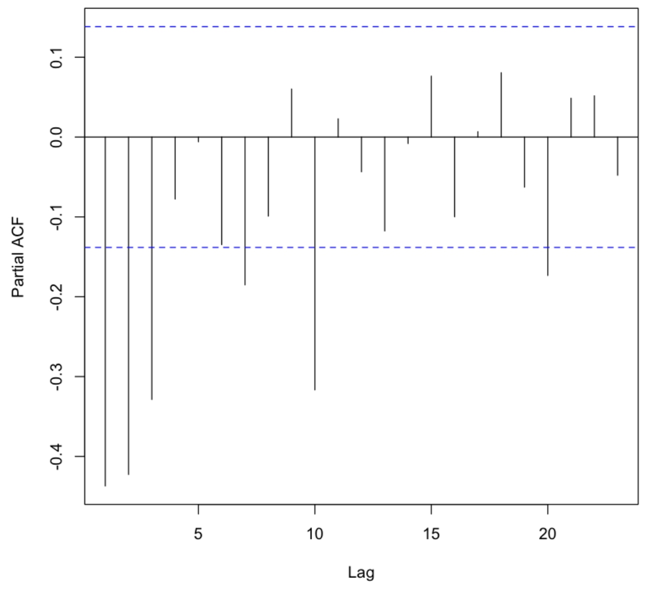

Fig. 8: P*ACF plot for the 1-step differenced time series data.*

The ACF (**Fig. 7**) plot shows that we have significant lags at 1 and 2, while the PACF (**Fig. 8**) plot shows that we have significant lags at 1, 2 and 3. Since we had a variable amount of significant lags, several ARIMA models were fit and the RMSEs were compared (Appendix). It was found that an ARIMA(2,1,2)(1,0,1) model produced the lowest RMSE value of 10.38445. We can see that this model produces a decent RMSE value which is very similar to the Holt-Winter model produced previously (**Table 1**). The ARIMA(2,1,2)(1,0,1) model was used to forecast the Airbnb prices into the year 2021 (**Fig. 9**), and we can see that the predicted prices also show a similar pattern and seasonality to the observed Airbnb dataset.

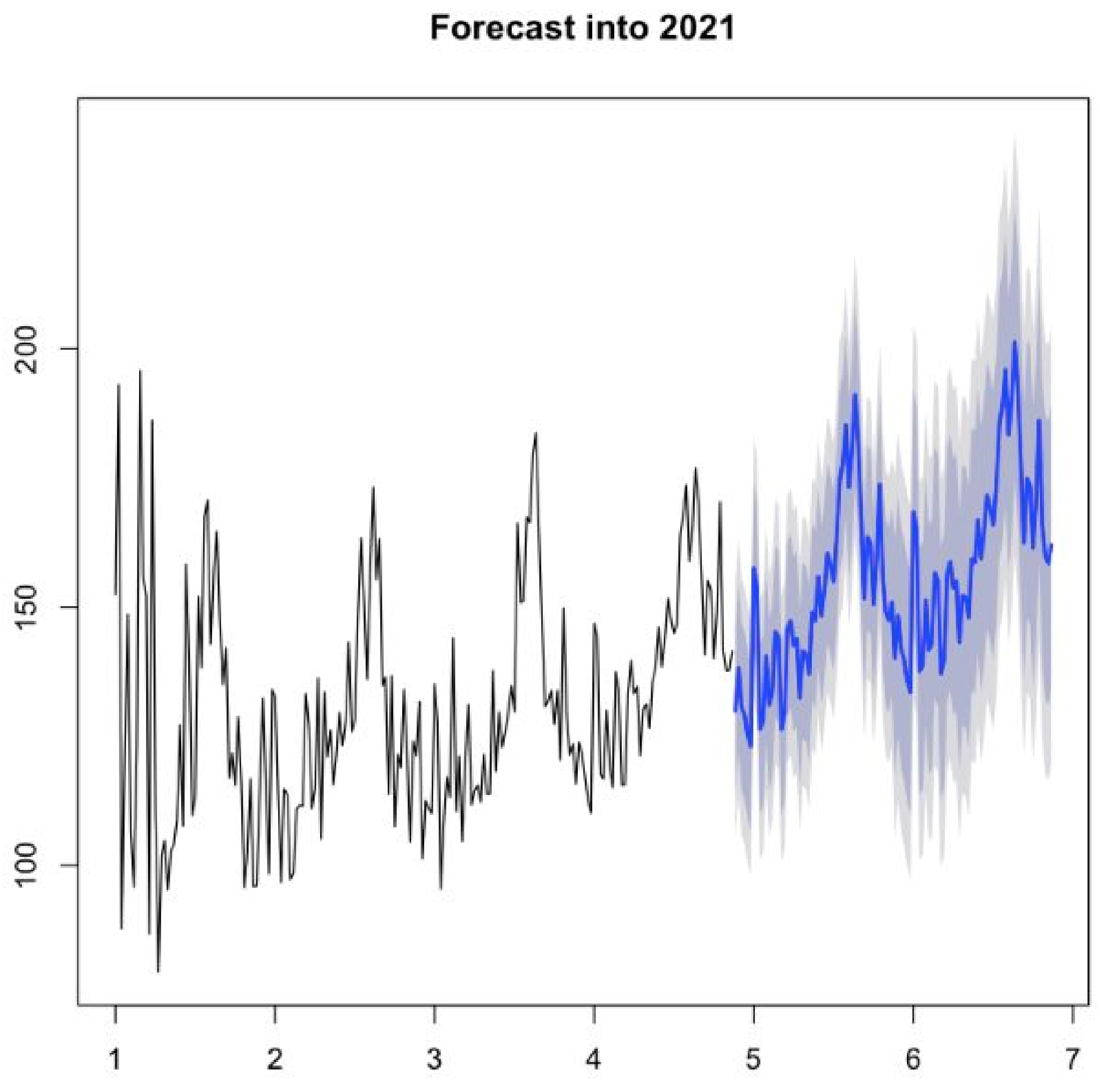

Fig. 9: *Forecasted weekly average price time series into 2021 based on ARIMA(2,1,2)(1,0,1) model.*

Next, an ARIMAX model was constructed for the Airbnb dataset. The pertinent variables for the ARIMAX model were selected via Lasso Regression. The significant lags from the ACF (**Fig. 7**) and PACF (**Fig. 8**) plots were used to fit several ARIMAX models and their respective RMSEs (Appendix), and the best model was deduced to be an ARIMAX(2,1,1)(1,0,1) model with an RMSE of 0.073661. The coefficients for this model are presented below in **Fig. 11** with their respective standard error.

```
Call:
arima(x = train.wk, order = c(2, 1, 1), seasonal = list(order = c(1, 0, 1),
    period = 52), xreg = vtrain[, c(1:5, 7:8, 11:14)], method = "CSS")

Coefficients:
          ar1      ar2      ma1      sar1    sma1  availability_365
      -0.2642  -0.0435  -0.2567  -0.0813  0.3579             3e-04
s.e.   0.3460   0.1506   0.3489   0.0734  0.1024             3e-04
      calculated_host_listings_count  cleaning_fee  extra_people
                             -0.0220         4e-04       -0.0008
s.e.                          0.0164         7e-04        0.0012
      guests_included  latitude  longitude  review_scores_accuracy
               0.3378    9.3376    -1.2183                  0.1125
s.e.           0.0515    2.9682     1.1888                  0.0837
      review_scores_checkin  review_scores_communication  reviews_per_month
                    -0.2441                       0.2217            -0.0505
s.e.                 0.1953                       0.1054             0.0130

sigma^2 estimated as 0.004377:  part log likelihood = 234.71
```

Fig. 11: *Coeffecients and standard errors for ARIMAX(2,1,1)(1,0,1) model.*

The ACF and PACF for the AIRIMAX(2,1,1)(1,0,1) model were consequently plotted (**Fig. 12**)

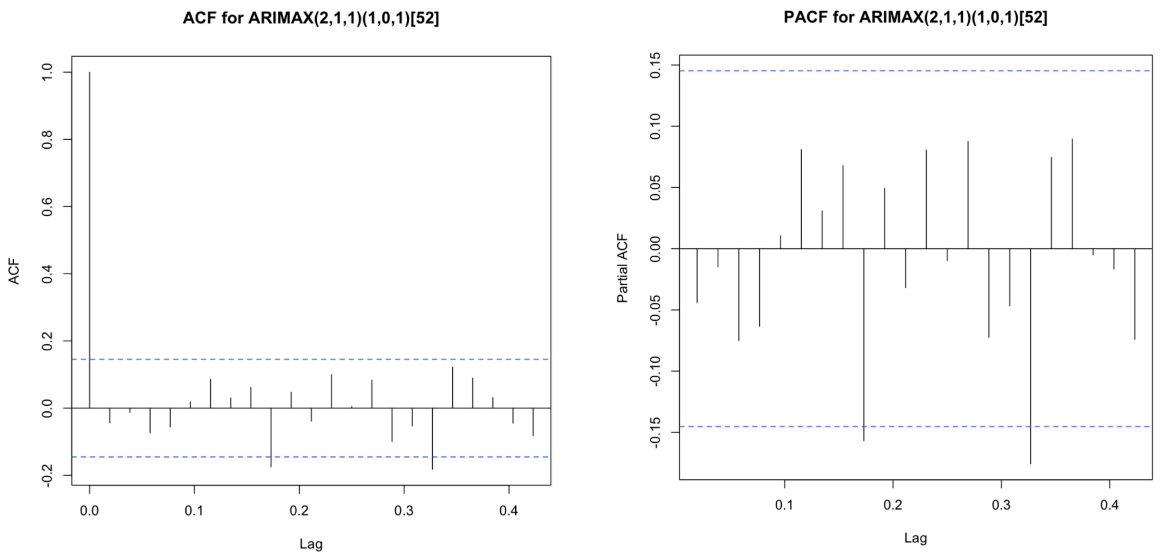

Fig. 12: *ACF and PACF plots for ARIMAX(2,1,1)(1,0,1) model.*

However, due to the lack of exogenous variables the AIRIMAX(2,1,1)(1,0,1) model could not be used to forecast the weekly average price for Airbnb listings into the year 2021.

Lastly, it is worth mentioning that approximately 15% of our data consisted of missing values, NaN's. We visualized this discrepancy using Amelia's `missmap` function in R.

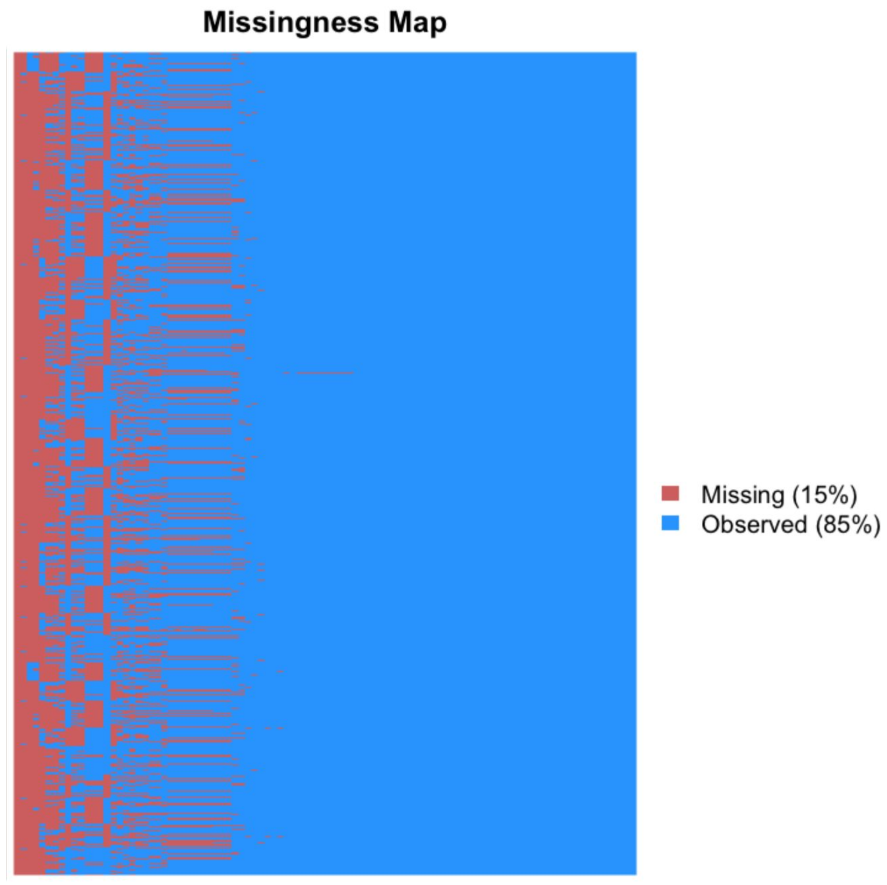

Fig. 13: *Missingness map for the Airbnb dataset.*

Contributions

Our team was formed on the basis of friendship. Having worked on previous projects together and spending a significant amount of time at The Nest's Gallery Lounge, our teamwork came as second nature. Following the consumption of several beverages, we decided to pull straws, which determined the ordering of the names according to length. Upon further deliberation, it was decided that the individual that could do the least amount of push-ups would be tasked with cleaning and pre-processing the data, a lengthy, tedious process. Unfortunately, that individual was Ash Sandhu.

Beyond that, however, Mark ran a multitude of LASSO iterations to select the pertinent variables for ARIMAX models and subsequently constructed those models. Akbar attained an astounding fifty push-ups during the pre-project planning session, and thus was tasked with comparing

RMSE’s for a variety of methods and with writing and compiling the final report. But alas, in the end, nobody stuck to what they were assigned and everyone ended up doing a bit of everything, true brothership.

Lastly, the amount and quality of collaboration eliminated the need for any kind of leadership role, especially under the guidance of Professor Harry Joe.

Appendix

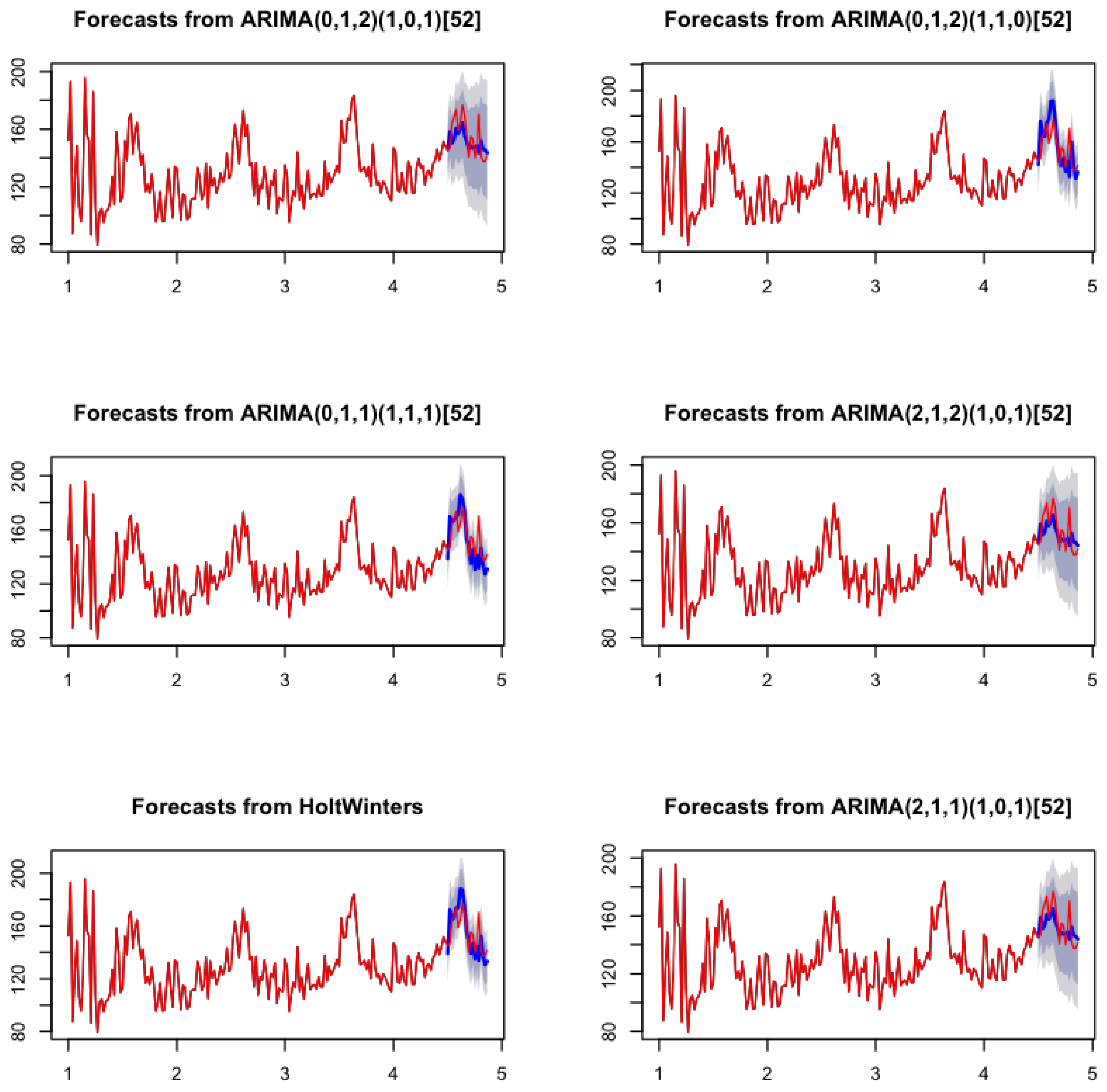

Fig. 14: *ARIMA models fit for the different lag values for ACF and PACF.*

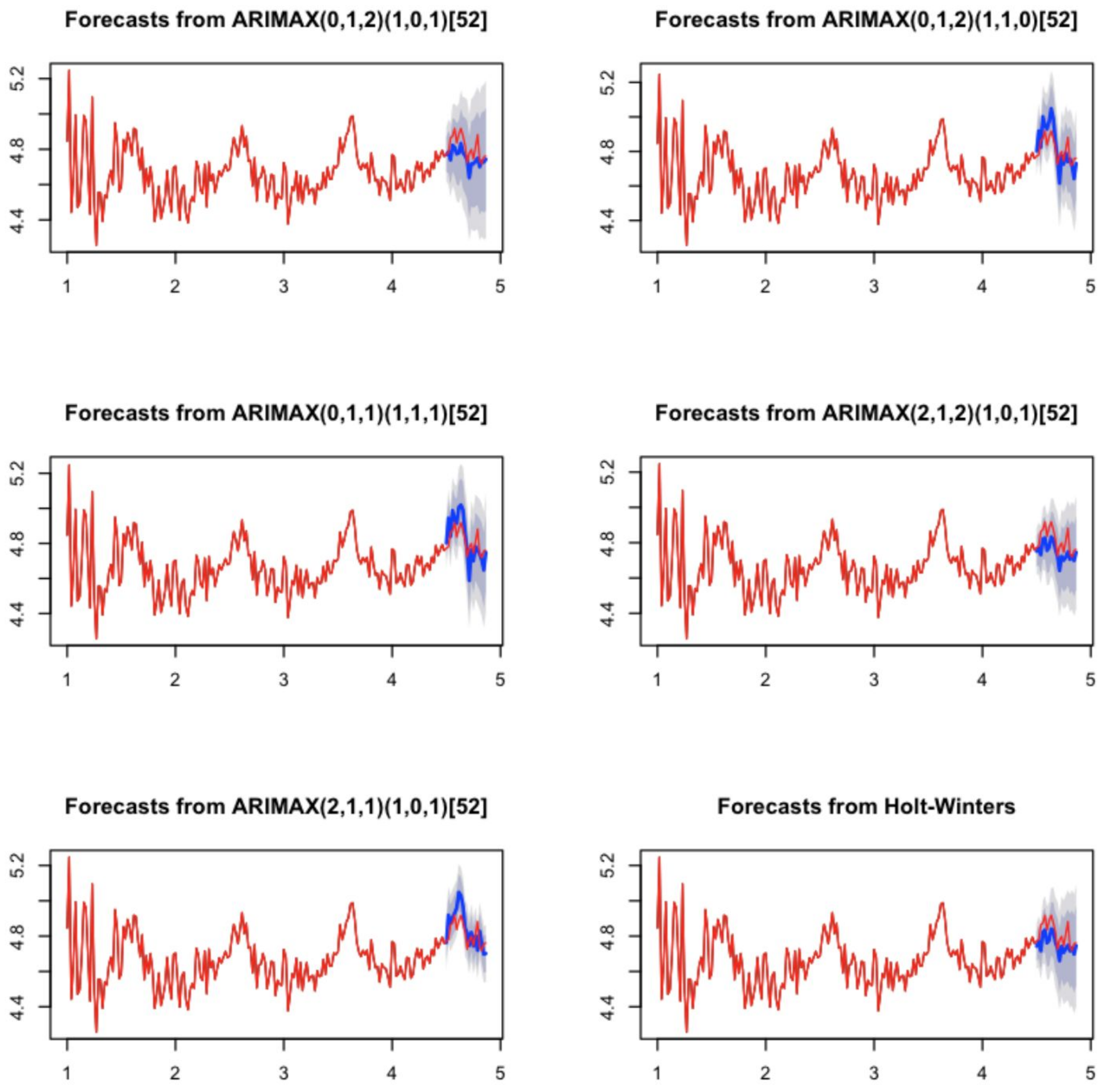

Fig. 15: *ARIMAX models fit for the different lag values for ACF and PACF.*

	ME	RMSE	MAE	MPE	MAPE	ACF1	Theil's U
ARIMA(0,1,2)(1,0,1)	**3.206987**	**10.58724**	**8.62564**	**1.616324**	**5.454843**	**0.083161**	**0.880402**

	ME	RMSE	MAE	MPE	MAPE	ACF1	Theil's U
ARIMA(0,1,2)(1,1,0)	**-2.08198**	**14.64793**	**10.46924**	**-1.37808**	**6.688597**	**0.118606**	**1.230631**

	ME	RMSE	MAE	MPE	MAPE	ACF1	Theil's U
ARIMA(0,1,1)(1,1,1)	**2.169485**	**13.49001**	**10.04688**	**1.423354**	**6.498787**	**0.159074**	**1.159702**

	ME	RMSE	MAE	MPE	MAPE	ACF1	Theil's U
ARIMA(2,1,2)(1,0,1)	**2.601856**	**10.38445**	**8.410879**	**1.227341**	**5.342311**	**0.069826**	**0.865678**

	ME	RMSE	MAE	MPE	MAPE	ACF1	Theil's U
Holt-Winters	**2.795508**	**10.45066**	**8.471474**	**1.352287**	**5.372979**	**0.071939**	**0.870645**

	ME	RMSE	MAE	MPE	MAPE	ACF1	Theil's U
ARIMA(2,1,1)(1,0,1)	**-0.31186**	**13.26653**	**9.387543**	**-0.20848**	**6.031749**	**0.104725**	**1.125607**

Fig. 16: *Test set RMSE values for fitted ARIMA models.*

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
ARIMAX(0,1,2)(1,0,1)[52]	0.060321	0.076789	0.063268	1.244078	1.306349	0.575391	0.006629	1.28143

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
ARIMAX(0,1,2)(1,1,0)[52]	-0.014298	0.08678	0.070279	-0.289586	1.454575	0.639148	0.277668	1.444587

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
ARIMAX(0,1,1)(1,1,1)[52]	-0.011748	0.088218	0.069986	-0.236295	1.450262	0.63648	0.303311	1.472373

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
ARIMAX(2,1,2)(1,0,1)[52]	0.064494	0.077481	0.065133	1.332445	1.345989	0.592349	-0.044156	1.292351

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
ARIMAX(2,1,1)(1,0,1)[52]	0.059467	0.073661	0.061418	1.227784	1.269098	0.558563	-0.117545	1.225994

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U

Holt-Winters	-0.031076	0.083305	0.064379	-0.643919	1.334484	0.585488	0.173383	1.383647

Fig. 17: *Test set RMSE values for fitted ARIMAX models.*

Code

```
#load relevant packages
#Make sure all these packages are installed before running this script
library(ggplot2)
library(forecast)
library(tseries)
library(IDPmisc)
library(zoo)
library(xts)
library(lubridate)
library(dplyr)
library(glmnet)
library(lars)
library(stats)
#####################

#load data
df <- read.csv('data/combined_listing.csv',header=TRUE,na.strings=c("","NA"))

#create a new df to explore price and time data
n <- nrow(df)
price.col <- df[1:n,'price']
price.col = as.numeric(gsub("\\$", "", price.col))
last.review <- as.Date(df[1:n,'last_review'])

#make new dataframe with only price and last review column
ts.df <- data.frame(price.col,last.review)


#order df by date
ts.df <- ts.df[order(as.Date(ts.df$last.review, format="%Y-%m-%d")),]

#better col names and clean up df
colnames(ts.df) <- c('price','date')
ts.df <- NaRV.omit(ts.df)

#get first and last date
start_date <- ts.df$date[1]
last_date <- ts.df$date[nrow(ts.df)]

#sequence of dates for date column
date <- data.frame(seq(from=start_date,to=last_date,by='days'))
colnames(date) <- c('date')

#remove any data from before 2015. Not needed/too little
ts.df <- subset(ts.df, ts.df$date> '2015-01-01')
```

```
#ts plot of raw prices. May be a little too noisy. Some filtering probably needed
plot(as.Date(ts.df$date),ts.df$price, type='l',ylab='price',xlab='Time',col='blue')


#initial plot too noisy. Group by day
meanprice <-  aggregate(ts.df$price,by=list(ts.df$date),mean)
colnames(meanprice) <- c('date','meanprice')
meanprice <- merge(x = date, y = meanprice, by = 'date', all = TRUE)
meanprice$date <- as.Date(meanprice$date)

#remove data from before 2015. Too little of it
meanprice <- na.omit(meanprice[meanprice[['date']] > as.Date('2015-01-01') ,])
meanprice <- data.frame(meanprice)
meanprice$meanprice <-meanprice$meanprice

#plot with aggregated data. ggplot because its prettier.
p3<- ggplot(meanprice,aes(date,meanprice)) + geom_line() +ylab('Daily Average Price')
+ xlab('Time')
print(p3)


#After consulting Harry Joe, he suggested it might make sense to aggregate data by
week and see if there are trends.
#group by week
daily_xts <- as.xts(meanprice$meanprice,order.by=as.Date(meanprice$date))
weekly_ts <- apply.weekly(daily_xts,mean)


#plot weekly time series
print(plot.xts(weekly_ts,main='Time series of average weekly price'))

#decompose time series
weekly <- ts(as.numeric(weekly_ts), frequency=52)
weekly_components <- decompose(weekly)

#plot decomposistion
plot(weekly_components)

#Holt-Winter exponential smoothing
exposmoothing_forecasts <- HoltWinters(weekly,seasonal = 'additive')
plot(exposmoothing_forecasts)

#testing stationarity
adf.test(weekly,alternative = "stationary")

#Variogram defintion:
#G(k) = Var(vt+k - vt)/Var(zt+1 - zt)
```

```
Var <- var(diff(weekly,lag=1))
Gk <- vector(mode='numeric',length=48)
for (k in 1:48){
  Gk[k] <- var(diff(weekly,lag=k))/Var
}

plot(1:48,Gk,ylab='Gk',xlab='Lag',type='l')

#deseason
deseasoned <- weekly - weekly_components$seasonal
plot(deseasoned)

#deaseasoned stil has a trend. Need to difference
diff_ds <- diff(deseasoned,differences = 1)
adf.test(diff_ds,alternative = 'stationary')
plot(diff_ds,type='l',main='Time series differenced by 1',ylab='Differenced price')

acf(as.numeric(diff_ds))
pacf(as.numeric(diff_ds))

#fit some different arimas
fit <- arima(weekly,order=c(3,1,2),seasonal =
list(order=c(1,0,0),period=52),method='CSS')
fit2 <- arima(weekly,order=c(2,1,2),seasonal =
list(order=c(1,0,0),period=52),method='CSS')
fit3 <- arima(weekly,order=c(3,1,4),seasonal =
list(order=c(1,1,0),period=52),method='CSS')
fit4 <- arima(weekly,order=c(0,1,2),seasonal =
list(order=c(1,1,0),period=52),method='CSS')

#summary of fit
summary(fit);summary(fit2);summary(fit3);summary(fit4)

#diagnostic plots
tsdisplay(residuals(fit), main='(3,1,2)(1,0,0) Model Residuals',lag.max = 20)
tsdisplay(residuals(fit2), main='(2,1,2)(1,0,0) Model Residuals',lag.max=20)
tsdisplay(residuals(fit3), main='(3,1,4)(1,1,0) Model Residuals',lag.max=20)
tsdisplay(residuals(fit4), main='(0,1,2)(1,1,0) Model Residuals',lag.max=20)

#create a test set and training set
test <- window(as.numeric(weekly), start=183)
train <-window(as.numeric(weekly[1:182]))
train <- ts(train,frequency = 52)

#train models on subset of data
train_fit3 <- arima(train,order=c(0,1,2),seasonal =
list(order=c(1,0,1),period=52),method='CSS')
```

```
train_fit4 <- arima(train,order=c(0,1,2),seasonal =
list(order=c(1,1,0),period=52),method='CSS')
train_fit5 <- arima(train,order=c(0,1,1),seasonal =
list(order=c(1,1,1),period=52),method='CSS')
train_fit6 <- arima(train,order=c(2,1,2),seasonal =
list(order=c(1,0,1),period=52),method='CSS')
train_fit7 <- arima(train,order=c(2,1,1),seasonal =
list(order=c(1,0,1),period=52),method='CSS')
holtWint <- HoltWinters(train,seasonal = 'additive')


#get forecats for holdout set
fcast_train3 <- forecast(train_fit3,h=20)
fcast_train4 <- forecast(train_fit4,h=20)
fcast_train5 <- forecast(train_fit5,h=20)
fcast_train6 <- forecast(train_fit6,h=20)
fcast_train7 <- forecast(train_fit7,h=20)
hwint_fcast <- forecast(holtWint,h=20)

par(mfrow=c(3,2))
#plot forecast and holdout data
plot(fcast_train3)
lines(weekly,col='red')


plot(fcast_train4)
lines(weekly,col='red')

plot(fcast_train5)
lines(weekly,col='red')

plot(fcast_train6)
lines(weekly,col='red')

plot(hwint_fcast)
lines(weekly,col='red')

plot(fcast_train7)
lines(weekly,col='red')

accuracy(fcast_train3,test);accuracy(fcast_train4,test);
accuracy(fcast_train5,test);accuracy(fcast_train6,test);
accuracy(fcast_train7,test);accuracy(hwint_fcast,test)


########### Feature Selection ###########

df$price = as.numeric(df$price)
```

```r
drops = c("weekly_price", "square_feet", "monthly_price", "is_business_travel_ready")
df = df[, !(names(df) %in% drops)]

data.complete = df[complete.cases(df),]
data.complete$price = as.numeric(data.complete$price)
data.complete1 = data.matrix(data.complete)

str(data.complete)

factorvars = colnames(data.complete[,sapply(data.complete, is.factor)])
factorvars


#LASSO for feature selection since orginal dataframe had 96 explanatory variables

removevars = c(factorvars, "price", "latitude", "longitude")

y <- as.vector(data.complete$price)
x = data.complete[, !(names(data.complete) %in% removevars)]
# x = data.frame(matrix(unlist(x), nrow = nrow(x), byrow = T))
# x = as.data.frame(x)
xm = as.matrix(x)
# alpha = 1 - LASSO
lambdas <- exp( seq(-3, 10, length=50))
a <- glmnet(x=xm, y=y, lambda=rev(lambdas),
            family='gaussian', alpha=1, intercept=TRUE)

plot(a, xvar='lambda', label=TRUE, lwd=6, cex.axis=1.5, cex.lab=1.2)
b = lars(x=xm, y=y, type="lasso", intercept=TRUE)
plot(b, lwd = 4)

##### ARIMAX model ########

#load data and clean it up
vars <-c('date',
         'price',
         'latitude',
         'longitude',
         'cleaning_fee',
         'review_scores_communication',
         'review_scores_accuracy',
         'calculated_host_listings_count',
         'availability_365',
         'review_scores_checkin',
         'reviews_per_month',
         'extra_people',
         'guests_included',
         'maximum_nights', 'last_review')
```

```r
df2 <- df[,names(df) %in% vars]

#change to date time object
df2$last_review <- as.Date(df2$last_review, format="%Y-%m-%d")

#subset dataframe
df2 <- df2[df2$last_review > as.Date("2015-01-01"),]

#clean it up
df2 <- NaRV.omit(df2)
df2$cleaning_fee <- as.numeric(df2$cleaning_fee)
df2$extra_people <- as.numeric(df2$extra_people)
df2$price <- as.numeric(df2$price)
df3 <-  NaRV.omit(df2)
df3 <- as.xts(df3,order.by=as.Date(df3$last_review))
weeklydf <- apply.weekly(df3,mean)

n = length(weekly)
ntest = 20
ntrain = n-ntest

train.wk = weekly[1:ntrain]
test.wk  = weekly[(ntrain+1):n]

train.wk = ts(train.wk, frequency = 52)
test.wk = ts(test.wk, frequency = 52)

vtrain = weeklydf[1:ntrain,]
vtest  = weeklydf[(ntrain+1):n,]

fitarimax = arima(train.wk, order=c(2,1,2),
                  seasonal=list(order=c(1,0,1), period=52),
                  xreg=vtrain[,c(1:5, 7:8, 11:14)], method="CSS")

# acf(fitarimax$residuals)

p1 = predict(fitarimax, n.ahead=20, newxreg=vtest[,c(1:5, 7:8, 11:14)])

plot(p1$pred)

#train models on subset of data
train_fit3 <- arima(train.wk,order=c(0,1,2),seasonal =
list(order=c(1,0,1),period=52), xreg=vtrain[,c(1:5, 7:8, 11:14)], method='CSS')
train_fit4 <- arima(train.wk,order=c(0,1,2),seasonal =
list(order=c(1,1,0),period=52), xreg=vtrain[,c(1:5, 7:8, 11:14)], method='CSS')
```

```
train_fit5 <- arima(train.wk,order=c(0,1,1),seasonal =
list(order=c(1,1,1),period=52), xreg=vtrain[,c(1:5, 7:8, 11:14)], method='CSS')
train_fit6 <- arima(train.wk,order=c(2,1,2),seasonal =
list(order=c(1,0,1),period=52), xreg=vtrain[,c(1:5, 7:8, 11:14)], method='CSS')
train_fit7 <- arima(train.wk,order=c(2,1,1),seasonal =
list(order=c(1,0,1),period=52), xreg=vtrain[,c(1:5, 7:8, 11:14)], method='CSS')


#get forecats for holdout set
fcast_train3 <- forecast(train_fit3, h=20, xreg=vtest[,c(1:5, 7:8, 11:14)])
fcast_train4 <- forecast(train_fit4, h=20, xreg=vtest[,c(1:5, 7:8, 11:14)])
fcast_train5 <- forecast(train_fit5, h=20, xreg=vtest[,c(1:5, 7:8, 11:14)])
fcast_train6 <- forecast(train_fit6, h=20, xreg=vtest[,c(1:5, 7:8, 11:14)])
fcast_train7 <- forecast(train_fit7, h=20, xreg=vtest[,c(1:5, 7:8, 11:14)])


par(mfrow=c(3,2))
#plot forecast and holdout data
plot(fcast_train3, main="Forecasts from ARIMAX(0,1,2)(1,0,1)[52]")
lines(weekly,col='red')


plot(fcast_train4, main="Forecasts from ARIMAX(0,1,2)(1,1,0)[52]")
lines(weekly,col='red')

plot(fcast_train5, main="Forecasts from ARIMAX(0,1,1)(1,1,1)[52]")
lines(weekly,col='red')

plot(fcast_train6, main="Forecasts from ARIMAX(2,1,2)(1,0,1)[52]")
lines(weekly,col='red')

plot(fcast_train7, main="Forecasts from ARIMAX(2,1,1)(1,0,1)[52]")
lines(weekly,col='red')

test.wk <- window(as.numeric(weekly), start=183)

accuracy(fcast_train3,test.wk);
accuracy(fcast_train4,test.wk);
accuracy(fcast_train5,test.wk);
accuracy(fcast_train6,test.wk);
accuracy(fcast_train7,test.wk)

##### DONE ########
```