Forecasting Annual Tourist Count at the Grand Canyon

Allison Yih

5/29/2020

1. Abstract

As one of the world's Seven Natural Wonders, the Grand Canyon not only attracts millions of worldwide visitors annually, but also serves as a major contributor to the tourism sector of the economy. In this time series analysis project, seasonal autoregressive integrated moving average (SARIMA) models will be developed and used to predict the number of visitors in the upcoming 10 years. By doing so, this can help the park's administrators understand visitor trends, and hopefully this can encourage them to more carefully plan their spending and better regulate park resources. Our time series model was successful in accurately forecasting future visitor counts for the Grand Canyon.

2. Introduction

Problem:

Grand Canyon National Park is one of America's biggest tourist attractions, where it draws over 6 million visitors per year, according to the National Park Service (NPS). However, national parks do not receive enough government funding, which makes it difficult in affording enough park maintenance and visitor services. Consequently, tourists are required to pay entrance fees in order to have these parks continue operating. Although more national park tourism benefits the economy, park overcrowding often causes environmental issues like pollution and litter, leading to extra, out-of-pocket costs for park improvement and conservation. Through developing an accurate forecast model, this can help park administrators alter the different parts of their yearly budget based on the number of tourists. Adjusting the annual budget's areas of spending can prevent park workers from experiencing the financial challenges that come with having an increased amount of visitors.

Software Used

To approach this question on predicting the number of annual visitors, time series forecasting will be used to identify and develop the model that follows all time series assumptions and fits the data well enough, in order to accurately predict the number of visitors for the following year. The RStudio software, as well as its readr, lubridate, dplyr, and forecast libraries, was used for performing the various statistical analysis tasks to build the final model and estimate future visitor counts.

Dataset Description

The dataset for this project is from the United States NPS, which is the agency that manages national parks such as the Grand Canyon. There are 98 observations in the dataset, with the response variable “Total Recreation Visitors” and predictor variable “Year”. The visitor count data was collected every year from 1919 to 2016. This data is found in the NPS section of the data.world website.

Techniques Used

To find the time series model that could accurately predict the number of visitors, the first step was to load in the dataset and plot a time series describing the visitor counts throughout the years. We then split the dataset into two parts; the first 88 observations would be used to identify and construct our model, and the remaining 10 would be used as a test set to validate that the chosen model gave highly accurate predictions.

After plotting the series, we noticed that there is trend and nonconstant variance. We used Box-Cox transformation to normalize the data and make the variance constant, and we differenced the series at lag 1 to remove the trend component, which resulted in a now-stationary series. Afterward, we plotted the ACF and PACF to identify possible p and q parameters for two possible models. Residual diagnostic checks were used to pick the final model. By passing the diagnostic tests, which is an assumption for forecasting, the chosen model was used to predict 10 observations. The forecasted observations and test set values were plotted to determine how well the model was at forecasting.

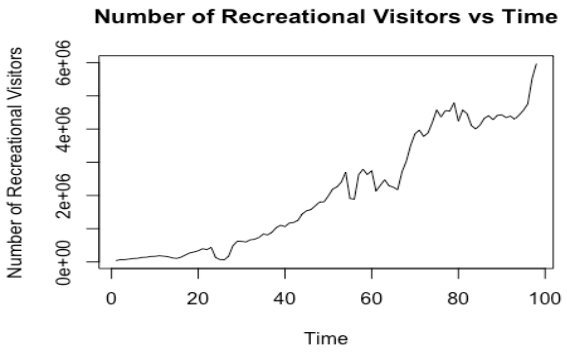
Results

Through developing a time series model that accurately performed predictions, the results from this analysis could be used to help park administrators more appropriately plan their services around tourist trends and ultimately enhance visitor experiences.

3. Analysis

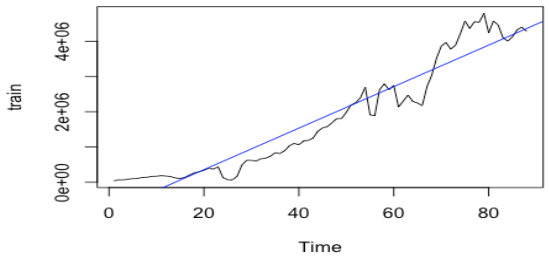
Exploratory Data Analysis

We load the dataset into R and use its response column, “Recreational Visitors”, to create and plot a time series object.



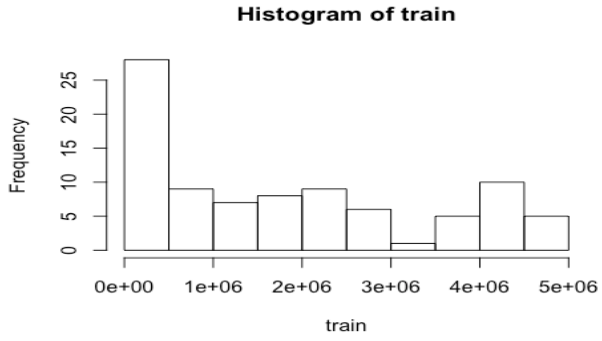
Then, we split the dataset into a training and test set. The test set contains the last 10 values of the original dataset and will be used during forecasting to determine how accurate the chosen model’s forecasts are to these 10 true values. From the plot, the series appears to have a positive trend and nonconstant mean and variance.

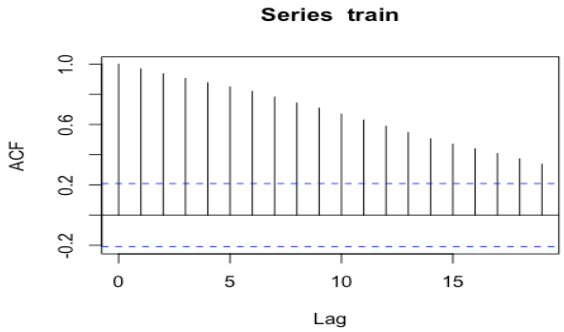
We now fit a regression line to our training data to check if our time series contains a trend component.



From the graph, it appears that there is an upward trend since the number of visitors gradually increases, and there is also nonconstant variance and mean. These are characteristics of a non-stationary series.

To confirm non-stationarity, the histogram and acf of the data are also plotted.

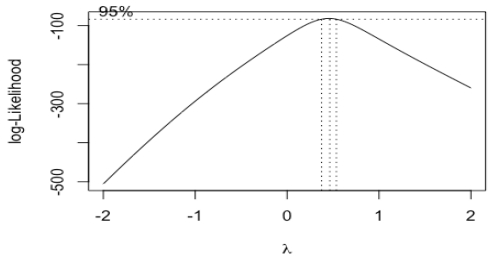




Because the time series has a nonconstant variance and trend, a skewed histogram, and large ACF values, these plots indicate that we need to transform the data to get a stationary series with constant variance and difference the series to remove trend.

Transforming the Data

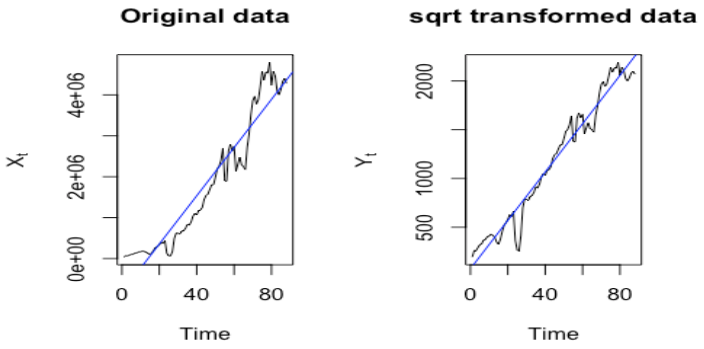
We use the Box-Cox method to find the optimal lambda that will stabilize variance and transform our data to more closely follow a Gaussian distribution.




```
## [1] 0.4646465
```

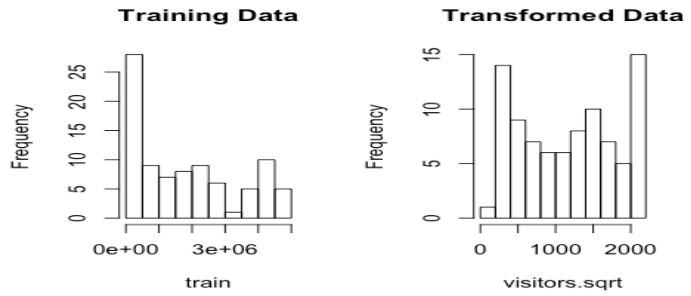
The box-cox transformation gives us a lambda value of 0.46, which rounds to 0.5. A lambda value of 0.5 indicates that we should take and plot a square root transformation of the original time series, and compare its line of best fit to the original series's.

We then compare the regression fits and variances for the original and transformed series.



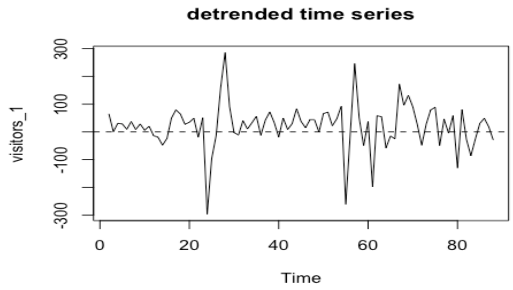
After Box-Cox and square root transformations on the data, the transformed series plot seems to have a more stable variance and more closely follows the regression line.

We can also check that the transformed series is a better fit by plotting its histogram.



Compared to the original data, the square root transformation appears to have a more constant variance and symmetry. Hence, the square root transformed series will be used to difference the series.

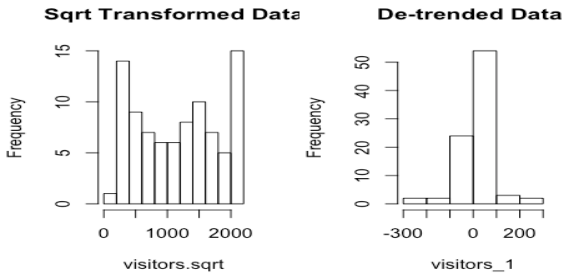
Because the time series has a trend component, we will difference the series at lag 1 to remove the trend.



From the plot, the differenced series seems to have a more constant variance and mean. We then compare the variances and histograms of square root transformation and differenced series to check if the series is stationary.

After differencing the series, the variance significantly decreased.

```
##                               variance
## sqrt transformed             418407.327
## sqrt trans. + differenced      6597.847
```



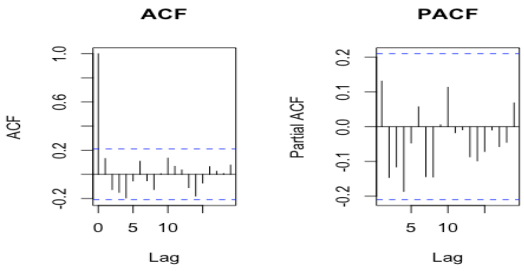
Histogram of the data after being differenced at lag 1 more closely follows the Gaussian distribution shape and looks more symmetric.

In addition, we use a Kwiatkowski–Phillips–Schmidt–Shin (KPSS) test, which is used to test whether a time series is stationary. If the KPSS test's p-value is less than the α significance level 0.05, then the null hypothesis is rejected and the series is not stationary.

```
##
##  KPSS Test for Level Stationarity
##
## data:  visitors_1
## KPSS Level = 0.046548, Truncation lag parameter = 3, p-value = 0.1
```

Since our p-value, which is 0.1, is greater than 0.05, we can confirm that the data is stationary.

Using ACF, PACF plots to identify preliminary model parameters



In the ACF plot, the ACF cuts off after lag 0, so we could have q = 0. The PACF is not significant at any of the lags, so we could have p = 0. Because we applied differencing once, then d = 1. Since our dataset had one observation per year, it is possible that there is an annual seasonal pattern, so an alternative model could also have D = 1 and s = 1.

```
##
## arima(x = visitors.sqrt, order = c(0, 1, 0), xreg =
1:length(visitors.sqrt),
##     method = "ML")
##
## Coefficients:
##       1:length(visitors.sqrt)
##                       21.5448
## s.e.                   8.6583
##
## sigma^2 estimated as 6522:  log likelihood = -505.51,  aic = 1015.01
```

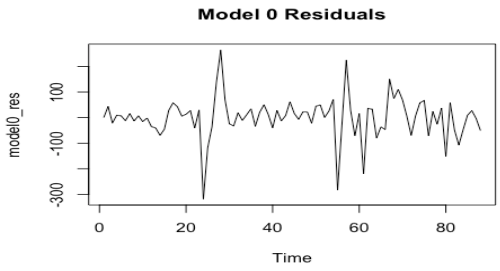
Since an ARIMA(0,1,0) model follows the form $X_t = X_{t-1} + \mu + \epsilon_t$, the equation for model 0 can be written as: $X_t = X_{t-1} + 21.5448 + \epsilon_t$.

```
##
## Call:
## arima(x = visitors.sqrt, order = c(0, 1, 0), seasonal = list(order = c(0,
1,
##     0), period = 1))
##
##
## sigma^2 estimated as 11411:  log likelihood = -523.75,  aic = 1049.5
```

Since an ARIMA(0,1,0) model follows the form $X_t = X_{t-1} + \mu + \epsilon_t$, the equation for model 1 can be written as: $X_t = X_{t-1} + \epsilon_t$.

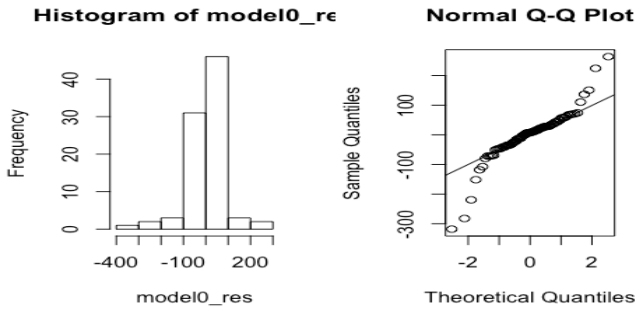
Diagnostic Checking

To determine how strong the fit is of our chosen models, we need to analyze each model's residuals. We start our diagnosis of model 0 by plotting its residuals and checking whether they are similar to White Noise. This is important because if the residuals do not have a mean of 0, which is a White Noise property, the forecasts would be more biased and less accurate.



From the graph, model 0's residuals appear to resemble White Noise. There are no trend and seasonality components, the mean is about 0, and the variance is constant.

We then plot the residuals on a histogram and normal-QQ plot to check whether they resemble a Gaussian distribution.



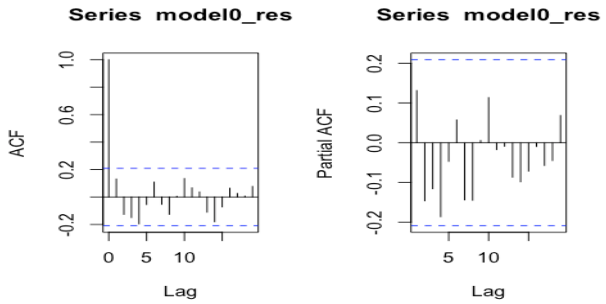
For the histogram, the residuals are in the shape of a bell curve, which is what Gaussian-distributed data looks like. On the normal QQ plot, the majority of the residuals are along the straight line, so the residuals appear approximately Gaussian.

We further test the residuals for normality using the Shapiro-Wilk test, check for model adequacy using the Box-Pierce and Ljung-Box tests, and determine nonlinear dependence of the residuals using the McLeod-Li test.

```
##
##  Shapiro-Wilk normality test
##
## data:  model0_res
## W = 0.87554, p-value = 4.839e-07

##
##  Box-Pierce test
##
## data:  model0_res
## X-squared = 1.5172, df = 1, p-value = 0.218

##
##  Box-Ljung test
##
## data:  model0_res
## X-squared = 1.5695, df = 1, p-value = 0.2103

##
##  Box-Ljung test
##
## data:  (model0_res)^2
## X-squared = 0.075811, df = 1, p-value = 0.7831
```

From the Shapiro test, our p-value is less than the α significance level 0.05, so we reject the normality assumption. However, our data passes the Box-Pierce, Ljung-Box, and McLeod-Li Portmanteau Statistic tests, as each p-value is greater than 0.05.

Next, to check if model 0's residuals follow White Noise, we plot the ACF and PACF of the residuals.



Since the residuals' ACF and PACF are inside the confidence intervals, they are all 0s and therefore represent White Noise.

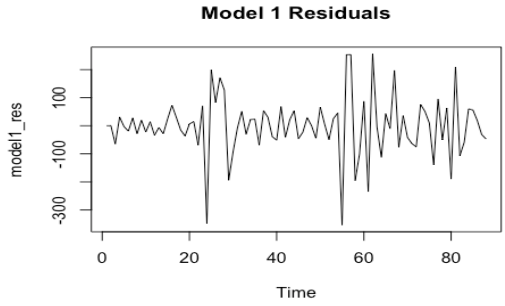
Finally, we use Yule-Walker estimation to determine whether the residuals fit an AR(0), or White Noise, model.

```
##
## Order selected 0  sigma^2 estimated as  6522
```

Since the ar() function selects order 0, this means our residuals were able to fit into an AR(0) model and can be considered as following White Noise. Overall, this model passes all but the Shapiro Wilk diagnostic tests.

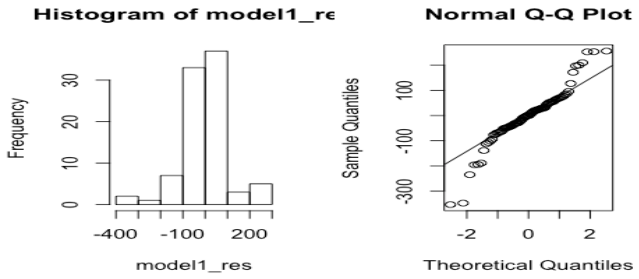
We then diagnose the residuals for model 1.

First, we plot the model's residuals and check whether they are similar to White Noise.



The graph indicates that model 1's residuals seem to resemble White Noise. There are no signs of trend, seasonality, and nonconstant variance, and the mean is approximately 0.

We then plot the residuals on a histogram and normal-QQ plot to check whether they resemble a Gaussian distribution.



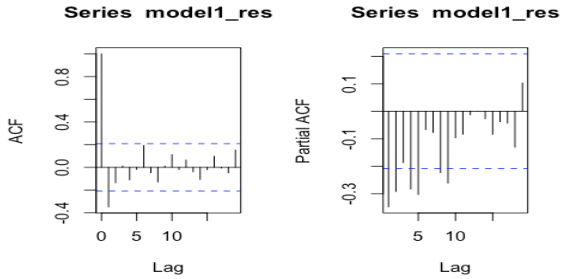
The histogram of the residuals has a bell curve, which is what Gaussian-distributed data looks like. Most of the residuals are on the straight line in the normal QQ plot. Based on these graphs, the residuals appear approximately Gaussian.

We further test the residuals for normality using the Shapiro-Wilk test, check for model adequacy using the Box-Pierce and Ljung-Box tests, and determine nonlinear dependence of the residuals using the McLeod-Li test.

```
shapiro.test(model1_res)

##
##  Shapiro-Wilk normality test
##
## data:  model1_res
## W = 0.92927, p-value = 0.0001294

Box.test(model1_res, lag = 10, type = c("Box-Pierce"), fitdf = 0)

##
##  Box-Pierce test
##
## data:  model1_res
## X-squared = 19.428, df = 10, p-value = 0.03516

Box.test(model1_res, lag = 10, type = c("Ljung-Box"), fitdf = 0)

##
##  Box-Ljung test
##
## data:  model1_res
## X-squared = 20.645, df = 10, p-value = 0.02371

Box.test((model1_res)^2, lag = 10, type = c("Ljung-Box"), fitdf = 0)

##
##  Box-Ljung test
##
## data:  (model1_res)^2
## X-squared = 27.118, df = 10, p-value = 0.002495
```

All p-values are smaller than $\alpha = 0.05$, so model 1 fails the Shapiro and all Portmanteau tests.

Next, to check if model 1's residuals follow White Noise, we plot the ACF and PACF of the residuals.



From the plots, the residual ACFs and PACFs are outside of the the confidence intervals. Therefore, they cannot be counted as 0s and do not represent White Noise.

Finally, we use Yule-Walker estimation to determine whether the residuals fit an AR(0), or White Noise, model.

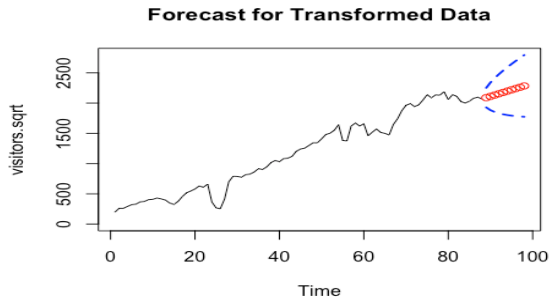
```
##
## Order selected 9  sigma^2 estimated as  7141
```

The ar() function selects the p parameter as order 9. This means that this model's residuals do not fit into an AR(0) model and cannot resemble White Noise.

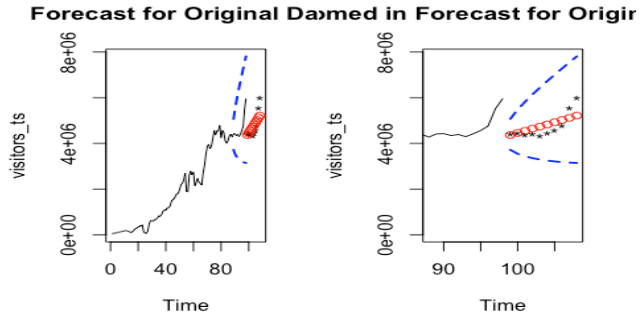
Model 1 fails the majority of the residual diagnostic tests, while model 0 passes all but one of the residual diagnostic tests. Clearly, model 0 would be the "best" model for forecasting future Grand Canyon visitor counts.

Forecasting

Model 0 satisfies the forecasting assumptions, which include passing the residual diagnostic checks and being developed through maximum likelihood estimation and AICC. We consider the model as appropriate for predicting future observations. We plot the next 10 observations on the transformed time series, and include a 95% prediction interval, which is represented by the blue dashed lines, for the forecasts, which are shown by the red circles.



From this graph, it shows that the estimated forecast values for our transformed data are within the 95% prediction intervals.



The graphs show that most of the forecasted values are very close to the actual observations, which are the original dataset's last 10 values that got put into a separate, testing set. We can conclude that our model is accurate in forecasting the number of annual visitors at the Grand Canyon.

4. Conclusion

Our time series model, $X_t = X_{t-1} + 21.5448 + \epsilon_t$, is able to accurately predict the Grand Canyon's visitor counts for the next 10 years. Using exploratory data analysis methods such as ACF and PACF preliminary parameter estimation, we were able to build a model that could successfully forecast future observations.

5. References

- Hughes, Trevor, and Karen Chávez. "Death on the Trail: More Americans Visit National Parks, but Fewer Rangers on Patrol." USA Today, Gannett Satellite Information Network, 9 July 2019, www.usatoday.com/story/news/nation/2019/06/29/national-parks-rangers-vanishing-putting-visitors-risk/1503627001/.
- Garder, John. "Background: The Economics of National Parks." National Parks Conservation Association, 15 Oct. 2015, www.npca.org/articles/832-background-the-economics-of-national-parks.
- Lock, S. "Grand Canyon National Park visitors 2019. " 18 Feb. 2020, https://www.statista.com/statistics/253878/number-of-visitors-to-grand-canyon-national-park/
- "Grand Canyon NP - Dataset by Nps." Data.world, 3 Nov. 2017, data.world/nps/grand-canyon-np.
- National Parks Service, U.S. Department of the Interior, irma.nps.gov/STATS/Reports/Park/GRCA.
- Brockwell, P. J., & Davis, R. A. "Introduction to time series and forecasting." New York: Springer.

6. Appendix

```
#loading libraries + dataset
library(readr)
library(lubridate)
library(dplyr)
library(forecast)
visitors_ds = read_csv("/Users/allisonyih/Desktop/grandcanyonannual.csv")

visitors_ts <- ts(visitors_ds$RecreationVisitors) #create time series object
plot.ts(visitors_ts, ylab = "Number of Recreational Visitors", main = "Number of Recreational Visito
rs vs Time")

#split data into training and test sets
train <- visitors_ts[c(1:88)]
test_obs = visitors_ts[c(89:98)] #leave 10 data points for model validation

lin_fit <- lm(train ~ as.numeric(1:length(train))) #fitting regression line to data
plot.ts(train)
abline(lin_fit, col = "Blue")

#exploratory data analysis on training set
hist(train)
acf(train)

#transforming our data
library(MASS)
t = 1:length(train)
fit = lm(train ~ t)
bcTransform = boxcox(train ~ t,plotit = TRUE) #box-cox transformation
lambda = bcTransform$x[which(bcTransform$y == max(bcTransform$y))] #get lambda value
lambda #0.46 so round to 0.5 = square root transform our data

op <- par(mfrow = c(1,2))
ts.plot(train,main = "Original data",ylab = expression(X[t]))
visitors.sqrt = as.ts(sqrt(train)) #convert into time series

transformed_fit <- lm(visitors.sqrt ~ as.numeric(1:length(visitors.sqrt)))
ts.plot(visitors.sqrt, main = "sqrt transformed data", ylab = expression(Y[t]))
abline(transformed_fit, col = "Blue") #regression line on transformed data

#compare histograms of original vs transformed data
op <- par(mfrow = c(1,2))
hist(train, main = "Training Data")
hist(visitors.sqrt, main = "Transformed Data")

#differencing our series at lag 1 to remove trend
library(ggplot2)
ndiffs(visitors.sqrt) #confirm number of differences needed
```

```
visitors_1 <- diff(visitors.sqrt, differences = 1)
var(visitors_1)
plot(visitors_1, main = "detrended time series")
abline(h = 0,lty = 2)

#compare sqrt transformed + differenced series - histograms
op <- par(mfrow = c(1,2))
hist(visitors.sqrt, main = "Sqrt Transformed Data")
hist(visitors_1, main = "De-trended Data ")

#compare sqrt transformed + differenced series - variance
variances <- c(var(visitors.sqrt), var(visitors_1))
matrix(variances, nrow = 2, ncol = 1, dimnames = list(c("sqrt transformed", "sqrt trans. + differenc
ed"), c("variance")))

#KPSS stationarity test
library(tseries)
kpss.test(visitors_1)

#ACF, PACF plots
op <- par(mfrow = c(1,2))
acf(visitors_1, na.action = na.pass, main = "ACF")
pacf(visitors_1, na.action = na.pass, main = "PACF")

#fitting 2 possible time series models
library(AICcmodavg)
model0 <- arima(visitors.sqrt, order = c(0, 1, 0), method = "ML", xreg = 1:length(visitors.sqrt))
model0

model1 <- arima(visitors.sqrt, order=c(0,1,0), seasonal = list(order = c(0, 1, 0), period = 1,  method
="ML"))
model1

#diagnostic checking model 0

model0_res <- model0$residuals
plot(model0_res, main = "Model 0 Residuals")

par(mfrow=c(1,2))
hist(model0_res) #check if residuals look normally-distributed
qqnorm(model0_res) #check if residuals follow the line
qqline(model0_res)

shapiro.test(model0_res) #portmanteau tests, p-vals should be greater than 0.05
Box.test(model0_res, lag = 1, type = c("Box-Pierce"), fitdf = 0)
Box.test(model0_res, lag = 1, type = c("Ljung-Box"), fitdf = 0)
Box.test((model0_res)^2, lag = 1, type = c("Ljung-Box"), fitdf = 0)

#acf, pacf - should be within the dashed confidence intervals
```

```
op <- par(mfrow = c(1,2))
acf(model0_res)
pacf(model0_res)

ar(model0_res, aic = TRUE, order.max = NULL, method = c("yule-walker"))

#diagnostic checking model 1
model1_res <- model1$residuals
plot(model1_res, main = "Model 1 Residuals")

par(mfrow=c(1,2))
hist(model1_res)
qqnorm(model1_res)
qqline(model1_res)

shapiro.test(model1_res)
Box.test(model1_res, lag = 10, type = c("Box-Pierce"), fitdf = 0)
Box.test(model1_res, lag = 10, type = c("Ljung-Box"), fitdf = 0)
Box.test((model1_res)^2, lag = 10, type = c("Ljung-Box"), fitdf = 0)

op <- par(mfrow = c(1,2))
acf(model1_res)
pacf(model1_res)

ar(model1_res, aic = TRUE, order.max = NULL, method = c("yule-walker"))

#forecast next 10 observations on transformed time series
pred.tr <- predict(model0, n.ahead = 10, newxreg=(length(visitors.sqrt)+1) : (length(visitors.sqrt)
+10))
U.tr= pred.tr$pred + 2*pred.tr$se #conf. interval upper bound
L.tr= pred.tr$pred - 2*pred.tr$se #conf. interval lower bound

ts.plot(visitors.sqrt, xlim = c(1,length(visitors.sqrt)+10), ylim = c(0, max(U.tr)))
lines(L.tr,lwd=2,col="blue",lty="dashed")
lines(U.tr,lwd=2,col="blue",lty="dashed")
points((length(visitors.sqrt)+1): (length(visitors.sqrt)+10),pred.tr$pred, col = "red")

#back-transform transformed series to get original series
pred.orig <- pred.tr$pred^2
U = U.tr^2
L = L.tr^2

op <- par(mfrow = c(1,2))

#plot test observations + forecasts using original data
pred.orig <- pred.tr$pred^2
U = U.tr^2
L = L.tr^2
```

```
op <- par(mfrow = c(1,2))
ts.plot(visitors_ts, xlim = c(1,length(visitors_ts)+10), ylim = c(0, max(U)), main = "Forecast for Ori
ginal Data")
lines(L,lwd=2,col="blue",lty="dashed")
lines(U,lwd=2,col="blue",lty="dashed")
points((length(visitors_ts)+1):(length(visitors_ts)+10),test_obs, pch = "*") #true values
points((length(visitors_ts)+1):(length(visitors_ts)+10), pred.orig, col = "red") #forecasted values

ts.plot(visitors_ts, xlim=c(length(visitors_ts)-10, length(visitors_ts) + 10), ylim = c(0,max(U)), mai
n = "Zoomed in Forecast for Original Data")
points((length(visitors_ts)+1):(length(visitors_ts)+10), pred.orig, col = "red") #forecasted values
points((length(visitors_ts)+1):(length(visitors_ts)+10),test_obs, pch = "*") #true values
lines((length(visitors_ts)+1): (length(visitors_ts)+10), U, lwd=2,col="blue",lty="dashed")
lines((length(visitors_ts)+1): (length(visitors_ts)+10), L,lwd=2,col="blue",lty="dashed"
```