TSA on Telecom Revenue Data

Alexander Vaillant

9/7/2021

Environment Setup

Import Necessary Libraries

```
library(astsa)
```

```
## Warning: package 'astsa' was built under R version 4.1.1
```

```
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
##
## Attaching package: 'forecast'
```

```
## The following object is masked from 'package:astsa':
##
##     gas
```

```
library(tseries)
library(stats)
library(TTR)
```

Data Gathering

Load Dataset into Dataframe using read.csv()

```
## Load dataset into dataframe
url <- "C:/Users/tedda/Desktop/Data Science Portfolio/Machine Learning/Supervised Learning/Time Series &
teleco_rev <- read.csv(url, header = TRUE, row.names = 'Day')
```

Data Preparation

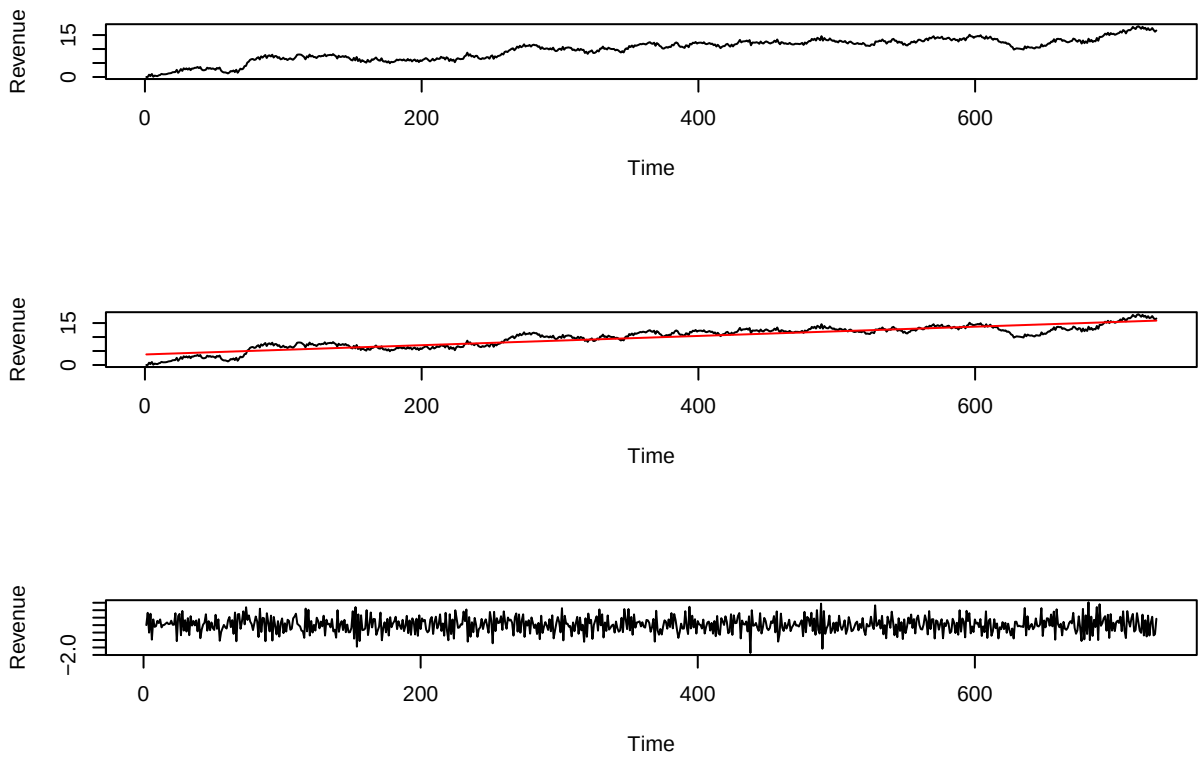
Convert the dataset to a time series object

```
ts_rev <- as.ts(teleco_rev)
```

Plot the dataset, dataset with trend line, and differenced data plots

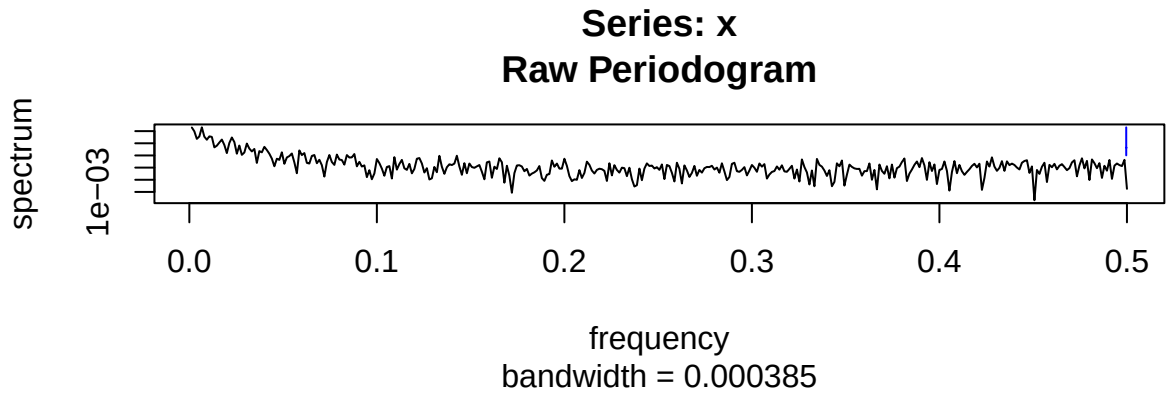
```
par(mfrow = c(3,1))
x <- (1:length(ts_rev))
plot(ts_rev)
```

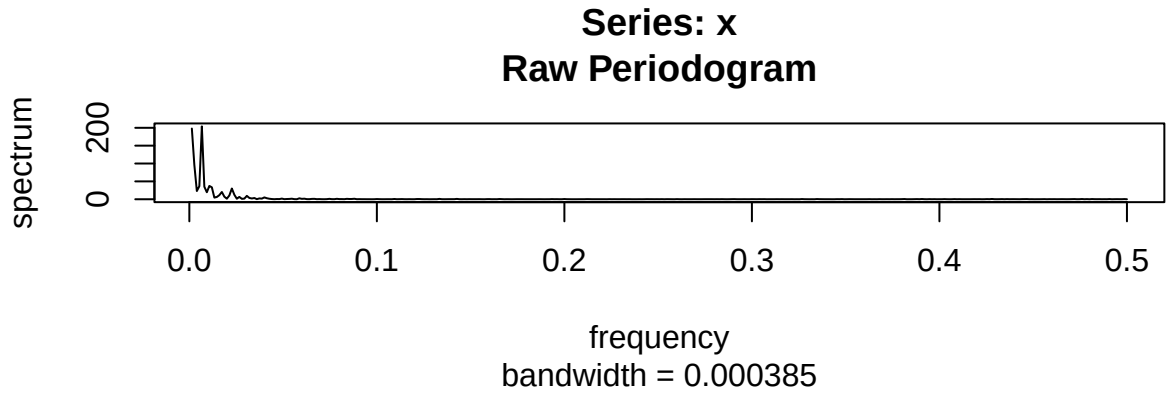
```
plot(ts_rev)
lines(predict(lm(ts_rev ~ x)), col = 'red')
plot(diff(ts_rev))
```



Plot the spectral density using both log = "yes" and "no"

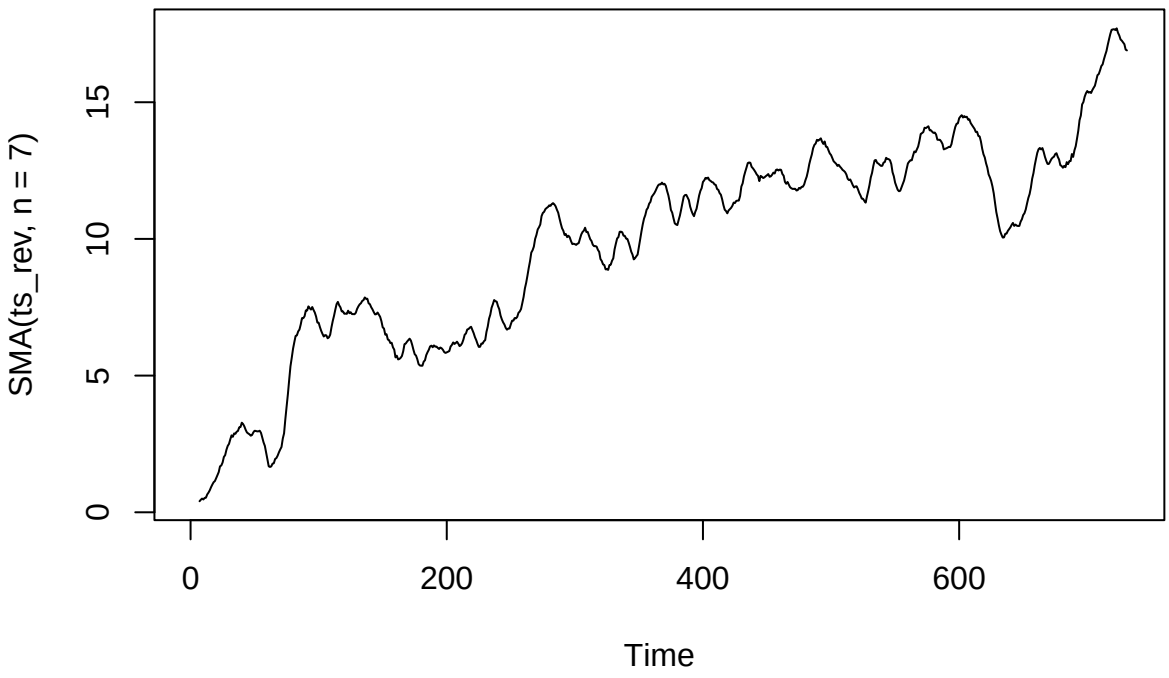
```
par(mfrow = c(2,1))
spectrum(ts_rev, log = "yes")
spectrum(ts_rev, log = "no")
```





Decompose our dataset by using a Simple Moving Average to smooth the data. n = 7 for 7 days per week cycle.

```
par(mfrow = c(1,1))
plot(SMA(ts_rev, n = 7))
```



There doesn't appear to be seasonality in our dataset, but test by using decompose()

```
decompose(ts_rev)
```

```
## Error in decompose(ts_rev): time series has no or less than 2 periods
```

```
# Results in an error that our ts has no or less than 2 periods/cycles of seasonality. (Lack of Seasona
```

Evaluate if the dataset is stationary by using the AD-Fuller test

```
adf.test(ts_rev)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ts_rev
## Dickey-Fuller = -3.6938, Lag order = 9, p-value = 0.02431
## alternative hypothesis: stationary
```

Split the data into train and test sets.

The training set will contain the data minus the last 60 days (2 months)

The test set will contain all of the data, including the last 60 days.

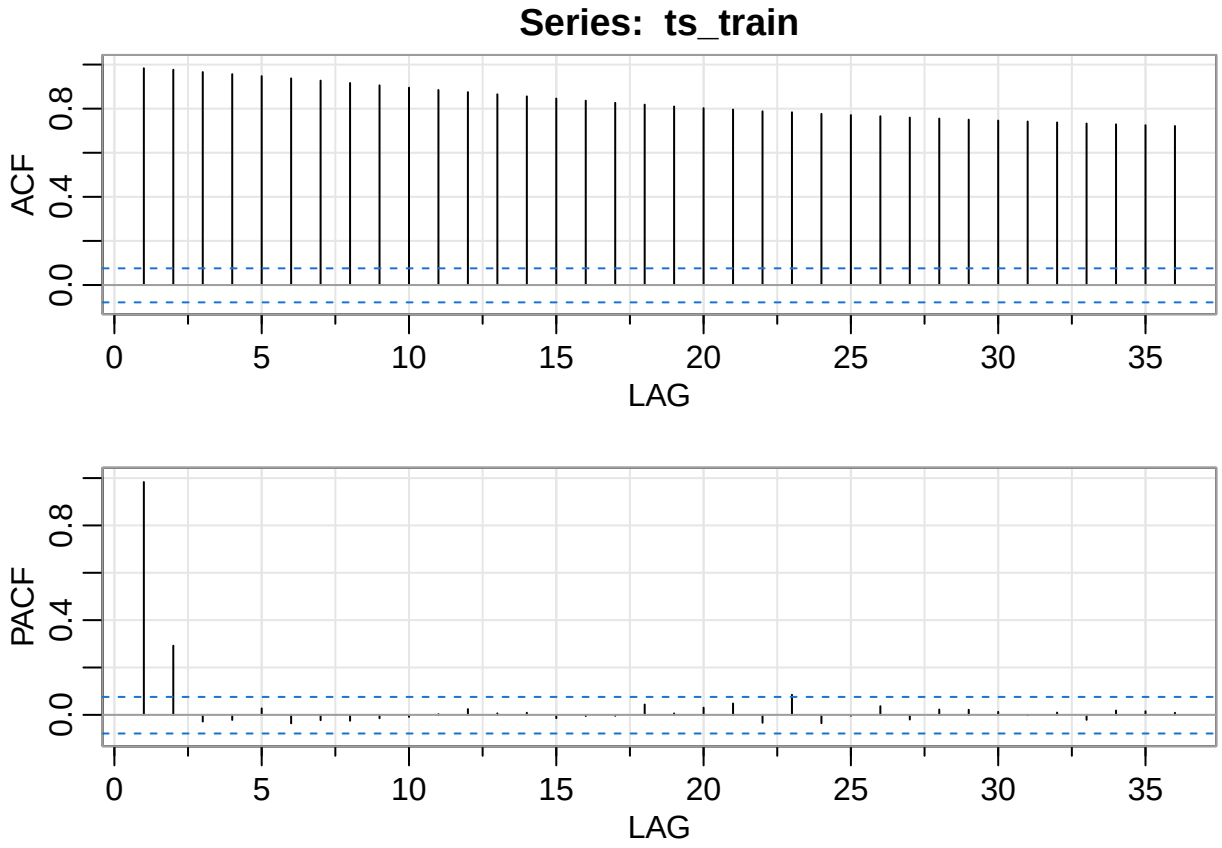
```
ts_train <- ts_rev[0:671]
ts_test <- ts_rev
```

Export the train and test datasets.

```
## Export the train and test datasets.
write.csv(ts_train, "C:/Users/tedda/Desktop/Data Science Portfolio/Machine Learning/Supervised Learning/
write.csv(ts_test,"C:/Users/tedda/Desktop/Data Science Portfolio/Machine Learning/Supervised Learning/T
```

Generate the ACF and PACF plots (Auto Correlation Function)

```
acf2(ts_train)
```




```
##       [,1] [,2]  [,3]  [,4] [,5]  [,6]  [,7]  [,8]  [,9] [,10] [,11] [,12] [,13]
## ACF  0.98 0.98  0.97  0.96 0.95  0.94  0.93  0.92  0.91  0.90  0.88  0.87  0.87
## PACF 0.98 0.29 -0.03 -0.02 0.03 -0.04 -0.02 -0.02 -0.01 -0.01  0.00  0.02  0.01
##       [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25]
## ACF   0.86  0.85  0.84  0.83  0.82  0.81  0.80  0.80  0.79  0.78  0.78  0.77
## PACF  0.01 -0.01  0.00  0.00  0.04  0.01  0.03  0.05 -0.03  0.08 -0.04  0.00
##       [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36]
## ACF   0.77  0.76  0.76  0.75  0.75  0.74  0.74  0.73  0.73  0.73  0.72
## PACF  0.04 -0.02  0.02  0.02  0.01  0.00  0.01 -0.02  0.02  0.02  0.01
```

Model Building

Use auto.arima from forecast library to find the best model

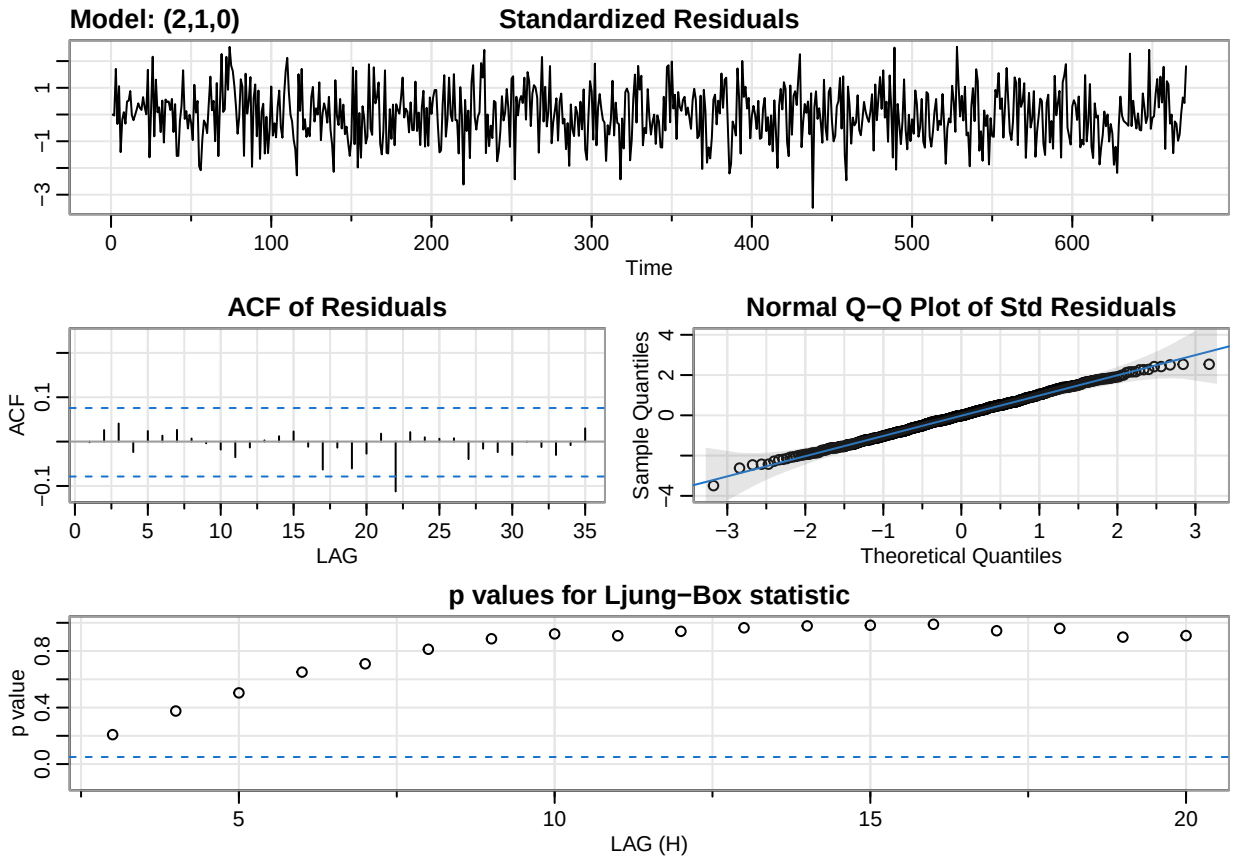
```
autoarima <- auto.arima(ts_train)
summary(autoarima)
```

```
## Series: ts_train
## ARIMA(2,1,0) with drift
##
## Coefficients:
##           ar1     ar2   drift
##       -0.4474  0.0144  0.0201
## s.e.   0.0387  0.0388  0.0126
##
## sigma^2 estimated as 0.218:  log likelihood=-439.08
## AIC=886.17   AICc=886.23   BIC=904.2
##
## Training set error measures:
##                          ME     RMSE       MAE  MPE MAPE      MASE       ACF1
## Training set -2.732736e-05 0.465563 0.3744196 -Inf  Inf 0.8821293 -0.0015183
```

View residual plots of best ARIMA model using sarima()

```
sarima(ts_train, p = 2, d = 1, q = 0)
```

```
## initial  value -0.648615
## iter   2 value -0.738161
## iter   3 value -0.762918
## iter   4 value -0.764425
## iter   5 value -0.764453
## iter   6 value -0.764453
## iter   7 value -0.764454
## iter   7 value -0.764454
## iter   7 value -0.764454
## final  value -0.764454
## converged
## initial  value -0.763586
## iter   2 value -0.763588
## iter   3 value -0.763590
## iter   4 value -0.763590
## iter   4 value -0.763590
## iter   4 value -0.763590
## final  value -0.763590
## converged
```




```
## $fit
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     xreg = constant, transform.pars = trans, fixed = fixed, optim.control = list(trace = trc,
##         REPORT = 1, reltol = tol))
##
## Coefficients:
##           ar1     ar2  constant
##       -0.4474  0.0144    0.0201
## s.e.   0.0387  0.0388    0.0126
##
## sigma^2 estimated as 0.2171:  log likelihood = -439.08,  aic = 886.17
##
## $degrees_of_freedom
## [1] 667
##
## $ttable
##          Estimate     SE  t.value p.value
## ar1       -0.4474 0.0387 -11.5545  0.0000
## ar2        0.0144 0.0388   0.3725  0.7096
## constant   0.0201 0.0126   1.6029  0.1094
##
## $AIC
## [1] 1.322638
##
```

```
## $AICc
## [1] 1.322692
##
## $BIC
## [1] 1.349547
```

Save and Load Model

```
# Save and Load Model
model_url <- "C:/Users/tedda/Desktop/Data Science Portfolio/Machine Learning/Supervised Learning/Time Se
saveRDS(autoarima, model_url)
TSA_model <- readRDS(model_url)
```

Model Evaluation

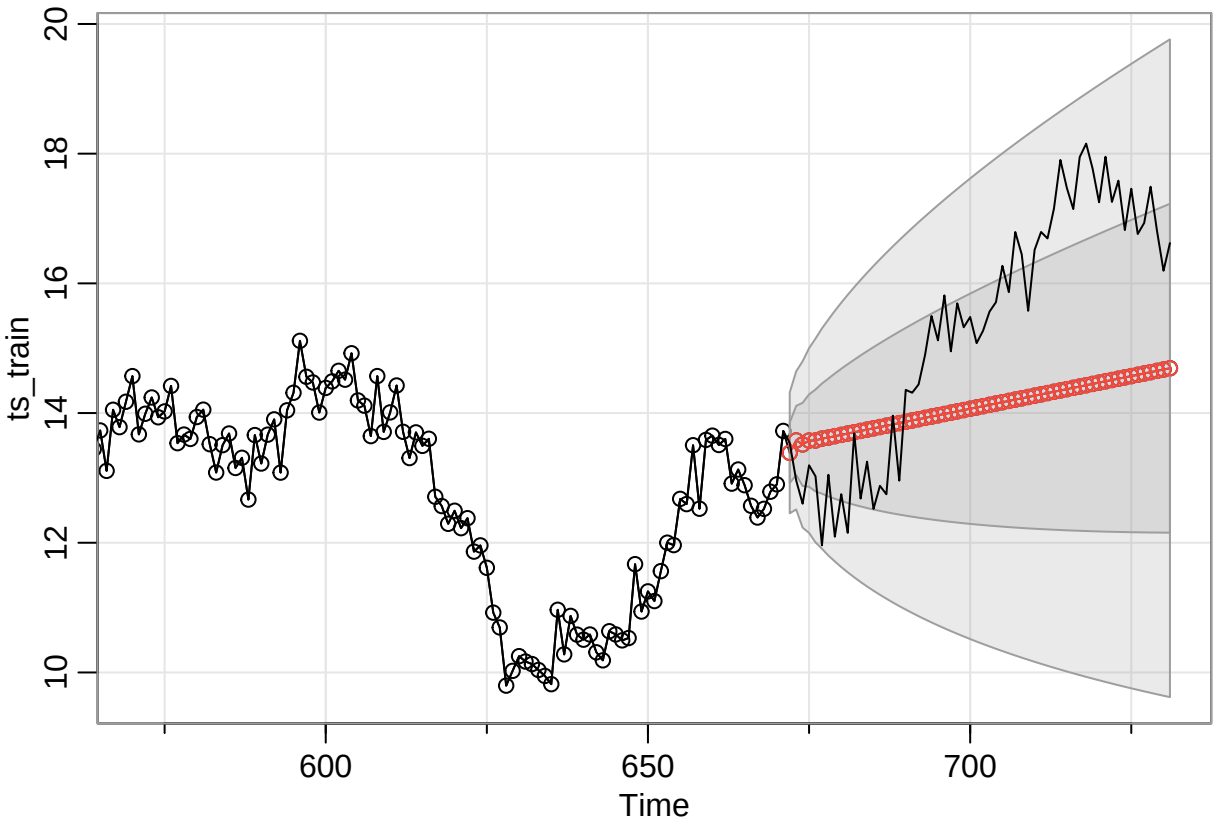
Forecast the next 60 days by using sarima.for() with the best ARIMA model found above

Plot the forecast's predictions against the actual revenue values

```
sarima.for(ts_train, n.ahead = 60, p = 2, d = 1, q = 0)
```

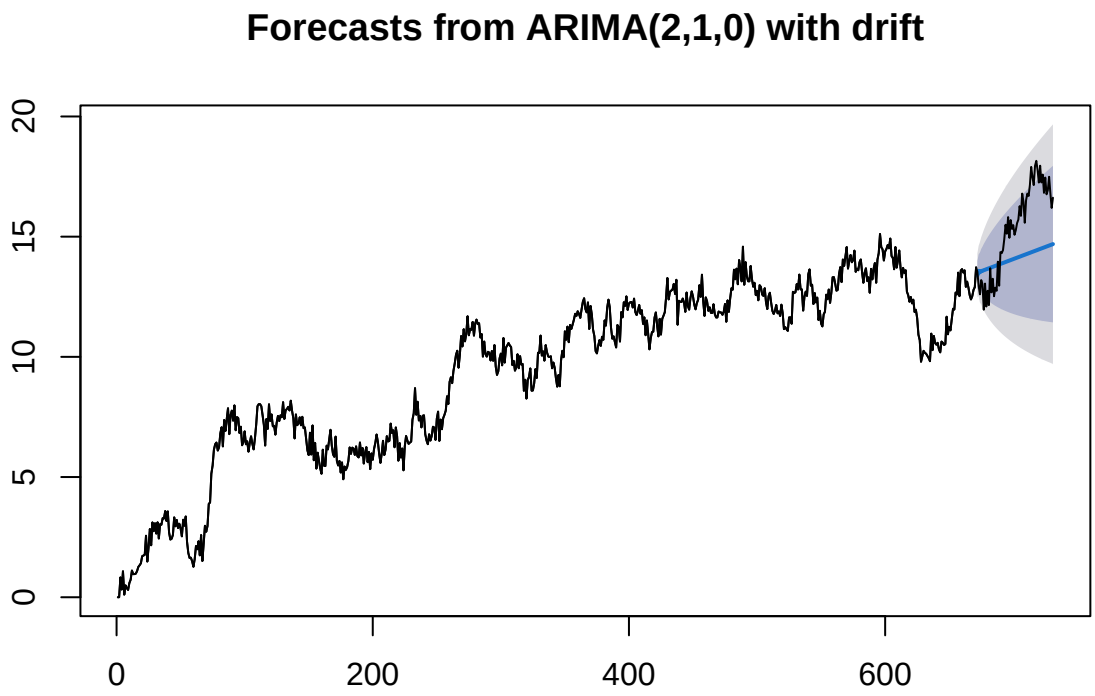
```
## $pred
## Time Series:
## Start = 672
## End = 731
## Frequency = 1
##  [1] 13.38621 13.57818 13.51628 13.57561 13.57704 13.60612 13.62200 13.64418
##  [9] 13.66335 13.68396 13.70388 13.72413 13.74423 13.76439 13.78453 13.80468
## [17] 13.82482 13.84496 13.86511 13.88525 13.90539 13.92554 13.94568 13.96583
## [25] 13.98597 14.00611 14.02626 14.04640 14.06655 14.08669 14.10683 14.12698
## [33] 14.14712 14.16727 14.18741 14.20755 14.22770 14.24784 14.26799 14.28813
## [41] 14.30827 14.32842 14.34856 14.36871 14.38885 14.40899 14.42914 14.44928
## [49] 14.46943 14.48957 14.50971 14.52986 14.55000 14.57015 14.59029 14.61043
## [57] 14.63058 14.65072 14.67087 14.69101
##
## $se
## Time Series:
## Start = 672
## End = 731
## Frequency = 1
##  [1] 0.4659103 0.5323182 0.6411960 0.7120763 0.7858857 0.8491504 0.9098732
##  [8] 0.9659526 1.0193285 1.0698730 1.1182137 1.1645134 1.2090579 1.2520110
## [15] 1.2935421 1.3337790 1.3728379 1.4108155 1.4477973 1.4838577 1.5190624
## [22] 1.5534694 1.5871307 1.6200928 1.6523974 1.6840825 1.7151824 1.7457283
## [29] 1.7757489 1.8052703 1.8343166 1.8629102 1.8910714 1.9188193 1.9461717
## [36] 1.9731450 1.9997544 2.0260144 2.0519384 2.0775389 2.1028278 2.1278161
## [43] 2.1525143 2.1769324 2.2010796 2.2249648 2.2485962 2.2719819 2.2951292
## [50] 2.3180455 2.3407374 2.3632115 2.3854738 2.4075302 2.4293864 2.4510478
## [57] 2.4725193 2.4938060 2.5149126 2.5358435
```

```
lines(ts_test)
```



View the full plot of the 60 day forecast using forecast() from Forecast library

```
## View the full plot of the 60 day forecast using forecast() from Forecast library
forecast60 <- forecast(TSA_model, h = 60)
plot(forecast60)
lines(ts_test)
```



Print forecast intervals at both 80% and 95% prediction intervals for the last 60 days.

```
forecast60
```

```
##     Point Forecast    Lo 80    Hi 80     Lo 95    Hi 95
## 672       13.38621 12.78778 13.98464 12.470993 14.30143
## 673       13.57818 12.89446 14.26191 12.532513 14.62385
## 674       13.51628 12.69271 14.33985 12.256739 14.77583
## 675       13.57561 12.66100 14.49022 12.176833 14.97439
## 676       13.57704 12.56763 14.58646 12.033273 15.12081
## 677       13.60612 12.51545 14.69680 11.938081 15.27417
## 678       13.62200 12.45333 14.79067 11.834674 15.40932
## 679       13.64418 12.40348 14.88488 11.746696 15.54167
## 680       13.66335 12.35409 14.97261 11.661016 15.66569
## 681       13.68396 12.30978 15.05814 11.582338 15.78558
## 682       13.70388 12.26761 15.14015 11.507301 15.90046
## 683       13.72413 12.22840 15.21987 11.436601 16.01166
## 684       13.74423 12.19127 15.29718 11.369193 16.11926
## 685       13.76439 12.15627 15.37251 11.304985 16.22380
## 686       13.78453 12.12306 15.44599 11.243535 16.32552
## 687       13.80468 12.09153 15.51782 11.184645 16.42471
## 688       13.82482 12.06150 15.58813 11.128060 16.52157
## 689       13.84496 12.03287 15.65706 11.073603 16.61632
## 690       13.86511 12.00551 15.72470 11.021101 16.70911
## 691       13.88525 11.97934 15.79116 10.970409 16.80009
## 692       13.90539 11.95426 15.85652 10.921398 16.88939
```

```
## 693       13.92554 11.93021 15.92086 10.873954 16.97712
## 694       13.94568 11.90712 15.98424 10.827975 17.06339
## 695       13.96583 11.88493 16.04672 10.783370 17.14828
## 696       13.98597 11.86358 16.10836 10.740055 17.23188
## 697       14.00611 11.84303 16.16920 10.697958 17.31427
## 698       14.02626 11.82323 16.22929 10.657011 17.39551
## 699       14.04640 11.80414 16.28867 10.617151 17.47565
## 700       14.06655 11.78572 16.34737 10.578324 17.55477
## 701       14.08669 11.76795 16.40543 10.540477 17.63290
## 702       14.10683 11.75078 16.46289 10.503563 17.71010
## 703       14.12698 11.73420 16.51976 10.467539 17.78642
## 704       14.14712 11.71817 16.57607 10.432364 17.86188
## 705       14.16727 11.70268 16.63186 10.398001 17.93653
## 706       14.18741 11.68769 16.68713 10.364415 18.01041
## 707       14.20755 11.67319 16.74192 10.331574 18.08353
## 708       14.22770 11.65915 16.79624 10.299447 18.15595
## 709       14.24784 11.64557 16.85012 10.268007 18.22768
## 710       14.26799 11.63241 16.90356 10.237226 18.29875
## 711       14.28813 11.61968 16.95658 10.207082 18.36918
## 712       14.30827 11.60734 17.00921 10.177549 18.43900
## 713       14.32842 11.59539 17.06145 10.148607 18.50823
## 714       14.34856 11.58381 17.11332 10.120234 18.57689
## 715       14.36871 11.57259 17.16482 10.092412 18.64500
## 716       14.38885 11.56172 17.21598 10.065122 18.71258
## 717       14.40899 11.55118 17.26681 10.038347 18.77964
## 718       14.42914 11.54097 17.31730 10.012070 18.84621
## 719       14.44928 11.53108 17.36748  9.986276 18.91229
## 720       14.46943 11.52149 17.41736  9.960950 18.97790
## 721       14.48957 11.51220 17.46694  9.936079 19.04306
## 722       14.50971 11.50320 17.51623  9.911647 19.10778
## 723       14.52986 11.49448 17.56524  9.887644 19.17207
## 724       14.55000 11.48603 17.61398  9.864057 19.23595
## 725       14.57015 11.47784 17.66245  9.840874 19.29942
## 726       14.59029 11.46991 17.71067  9.818084 19.36250
## 727       14.61043 11.46223 17.75863  9.795677 19.42519
## 728       14.63058 11.45480 17.80636  9.773643 19.48751
## 729       14.65072 11.44760 17.85384  9.751972 19.54947
## 730       14.67087 11.44064 17.90110  9.730655 19.61108
## 731       14.69101 11.43390 17.94812  9.709683 19.67234
```