ARIMA Forecast

Curtis Hammons

3/16/2022

Research Question

Our objective in this analysis is to identify trends in revenue and forecast future revenue. Our goal is to forecast the upcoming half year's revenue.

Method Justification

We'll be using seasonal Auto-Regressive Integrated Moving Average (ARIMA) to make our forecast. ARIMA models make predictions based on a given time series.

Data Preparation

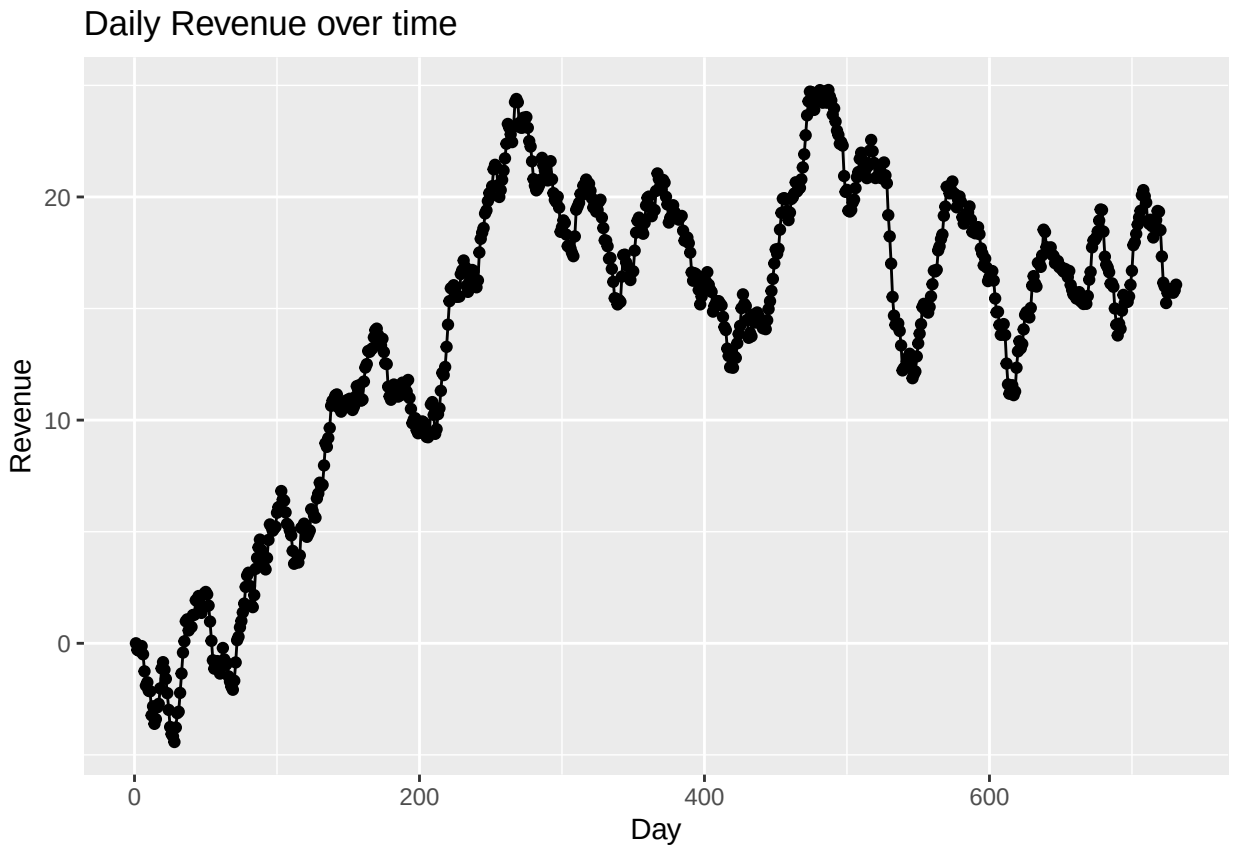
Let's import and inspect our data.

```
df = read.csv("data/medical_time_series.csv")
head(df)
```

```
##   Day    Revenue
## 1   1  0.0000000
## 2   2 -0.2923555
## 3   3 -0.3277718
## 4   4 -0.3399871
## 5   5 -0.1248875
## 6   6 -0.4915896
```

Our dataset is the daily revenue over the previous two years of operation, with each datapoint beint a single day. Let's look at the plot of the data.

```
df %>% ggplot(aes(x=Day, y=Revenue)) +
  geom_point() +
  geom_line() +
  ggtitle("Daily Revenue over time")
```



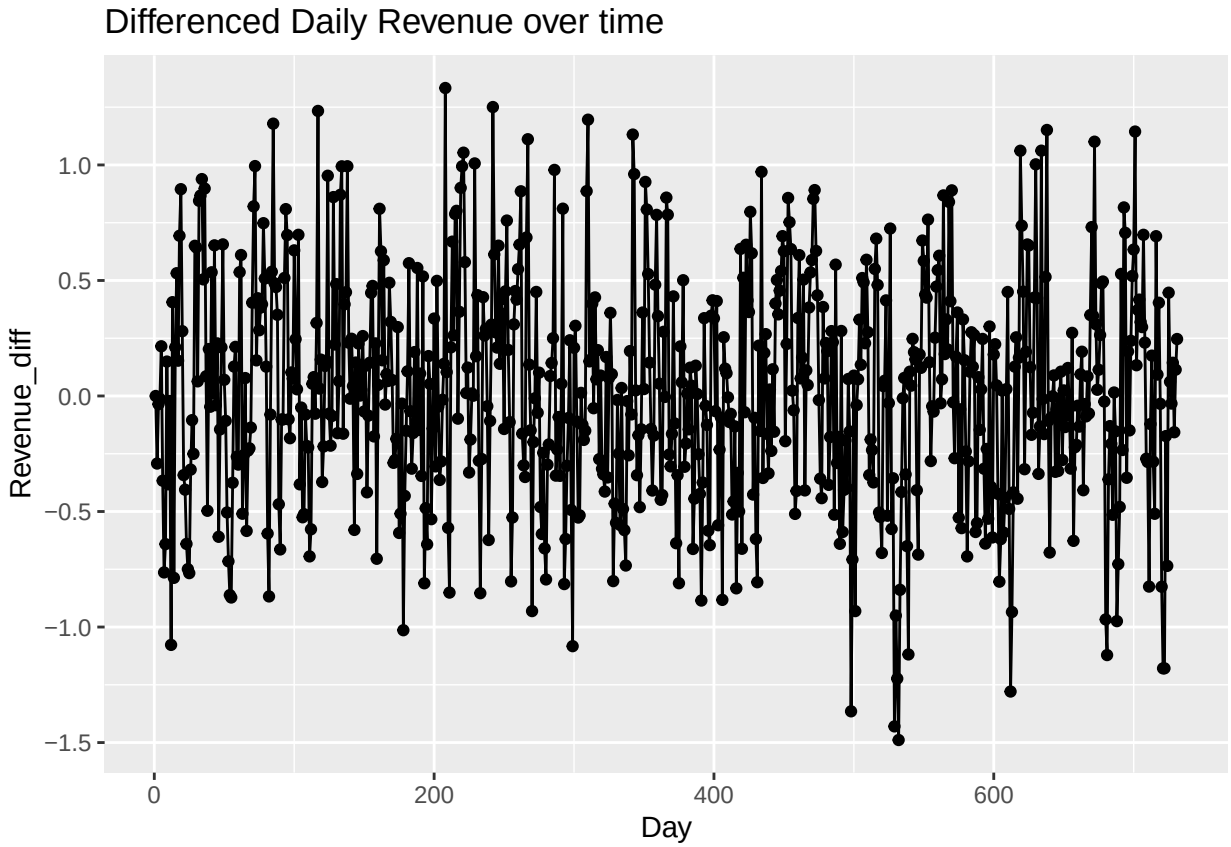
We can see from the graph that the data are not stationary and are. There is an obvious upward trend in the first year. This will by taking one difference of the raw data. We also see Seasonal flucuations in the data.

We'll first convert our revenue data into a TimeSeries object. Then We'll take the difference and plot it.

```
series = ts(df$Revenue, frequency = 365, start= c(2016,1), end = c(2018,1))

df$Revenue_diff = c(0, diff(series))

df %>% ggplot(aes(x=Day, y=Revenue_diff)) +
  geom_point() +
  geom_line() +
  ggtitle("Differenced Daily Revenue over time")
```




```
write_csv2(df, "data/time_series_diff.csv")
```

As we can see the data is now stationary. The Seasonality and early trend has been removed. Now we'll split the data into training and test sets.

```
# we're using an 80/20 split
train = ts(df$Revenue, frequency = 365,start = c(2016,1), end=c(2017,220))
test = ts(df$Revenue, frequency = 365,start = c(2017,221), end=c(2018,1))
```

Model and Analysis

Finding a model

We'll use `auto.arima` to select the best p, q, d, P, and Q parameters for our model. We'll specify a D of 1.

```
#the auto.arima function automatically applies the Difference so we pass the raw data
train_model = auto.arima(train, D=1)
summary(train_model)
```

```
## Series: train
## ARIMA(1,1,2)(0,1,0)[365]
##
## Coefficients:
##          ar1     ma1     ma2
##       0.1571  0.2597  0.2599
## s.e.  0.1952  0.1879  0.0866
##
## sigma^2 = 0.3997:  log likelihood = -212.45
```

```
## AIC=432.91   AICc=433.1   BIC=446.47
##
## Training set error measures:
##                       ME      RMSE       MAE         MPE     MAPE       MASE
## Training set -0.01092138 0.3841846 0.1921876 -0.02796609 1.126499 0.01498081
##                      ACF1
## Training set -0.001395572
```

The model selected for the test set has the following parameters:

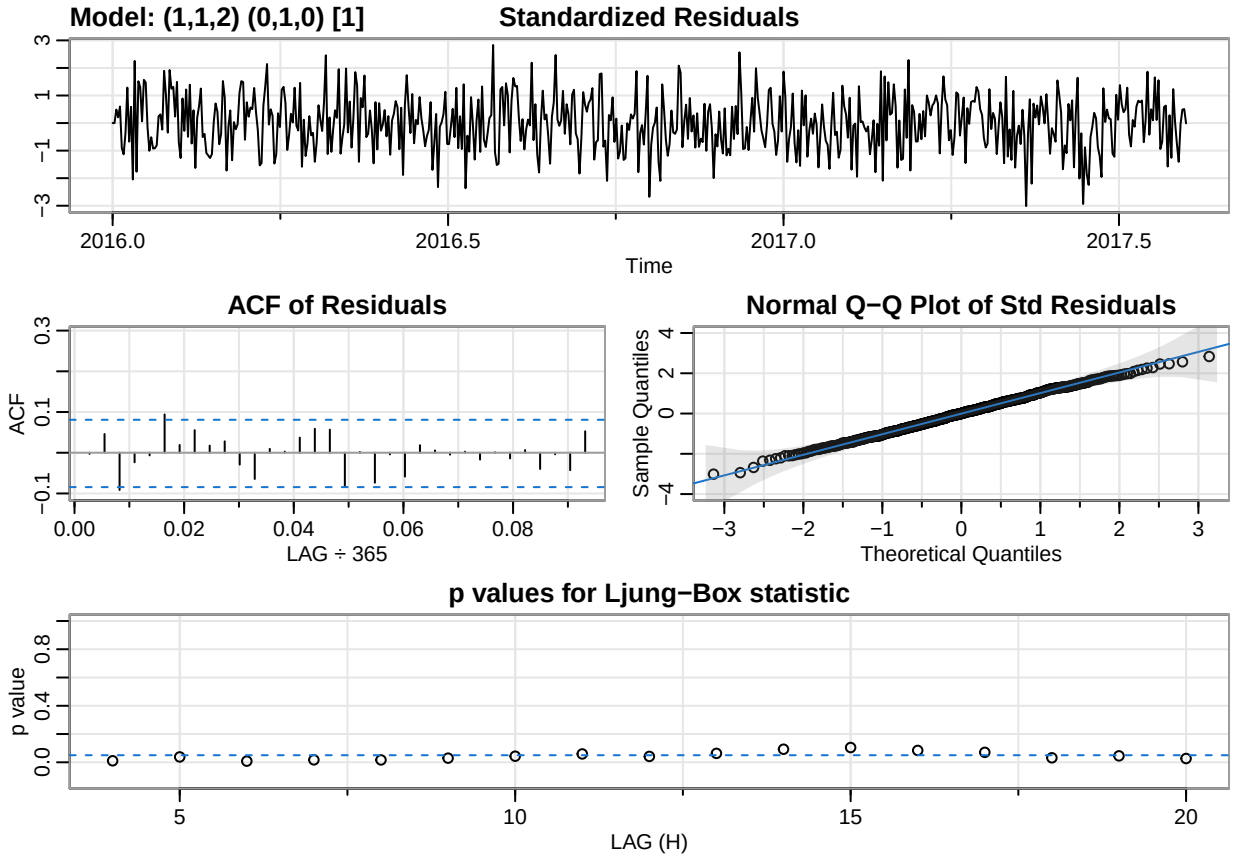
- p = 1
- d = 1
- q = 2
- P = 0
- D = 1
- Q = 0

Now we'll perform a forecast on the test set and compare it to the actual data

```
sarima(train, p=1, d=1, q=2, P=0, D=1, Q=0, S=1)
```

```
## initial  value -0.642535
## iter   2 value -0.695198
## iter   3 value -0.758288
## iter   4 value -0.770780
## iter   5 value -0.776111
## iter   6 value -0.783659
## iter   7 value -0.783929
## iter   8 value -0.785421
## iter   9 value -0.787462
## iter  10 value -0.790564
## iter  11 value -0.796848
## iter  12 value -0.798806
## iter  13 value -0.799611
## iter  14 value -0.800694
## iter  15 value -0.800923
## iter  16 value -0.801685
## iter  17 value -0.802725
## iter  18 value -0.804717
## iter  19 value -0.807106
## iter  20 value -0.809089
## iter  21 value -0.810162
## iter  22 value -0.810359
## iter  23 value -0.813718
## iter  24 value -0.815065
## iter  25 value -0.816040
## iter  26 value -0.816994
## iter  27 value -0.817175
## iter  28 value -0.818485
## iter  29 value -0.818642
## iter  29 value -0.818642
## iter  29 value -0.818642
## final  value -0.818642
## converged
## initial  value -0.808025
## iter   2 value -0.808757
```

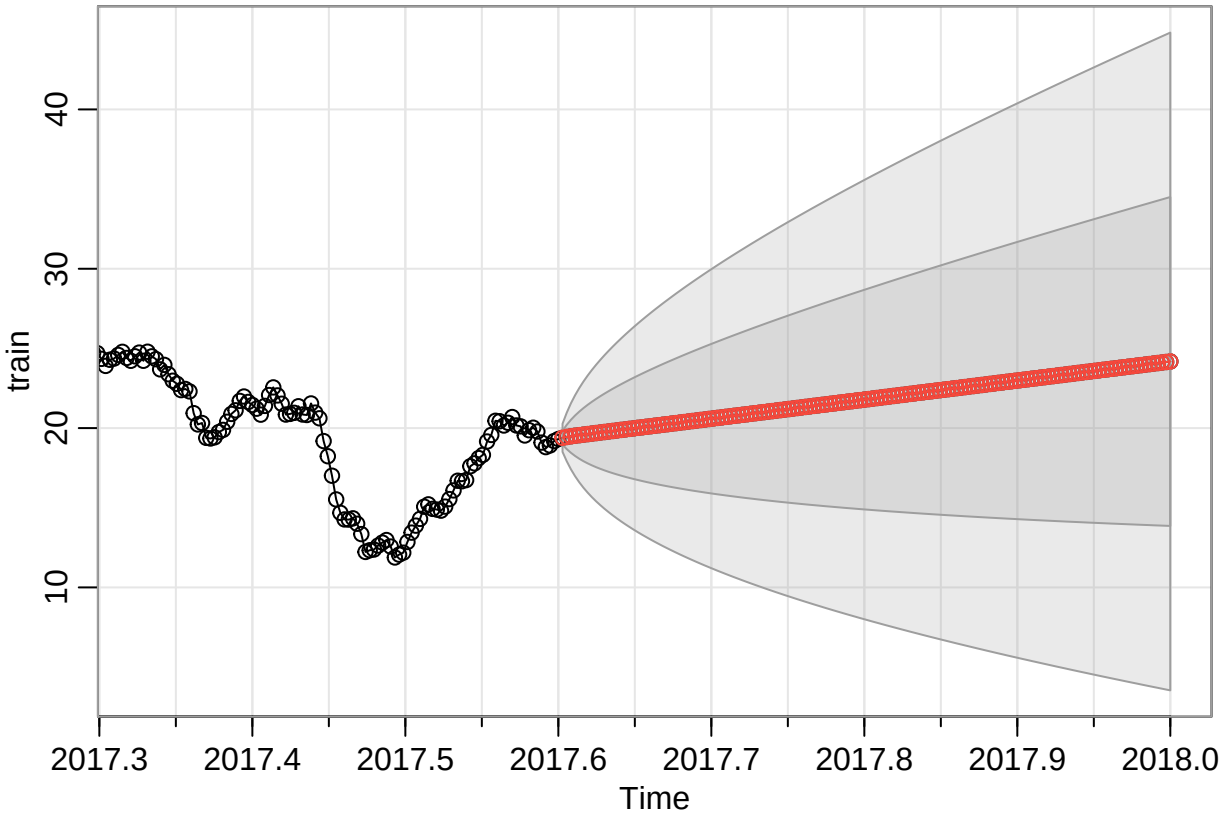
```
## iter   3 value -0.809684
## iter   4 value -0.810833
## iter   5 value -0.813840
## iter   6 value -0.814006
## iter   7 value -0.814021
## iter   8 value -0.814021
## iter   9 value -0.814021
## iter   9 value -0.814021
## iter   9 value -0.814021
## final  value -0.814021
## converged
```




```
## $fit
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     include.mean = !no.constant, transform.pars = trans, fixed = fixed, optim.control = list(trace =
##         REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1      ma1     ma2
##       0.4594  -1.0586  0.0586
## s.e.  0.0776   0.0844  0.0841
##
## sigma^2 estimated as 0.1945:  log likelihood = -352.67,  aic = 713.33
##
## $degrees_of_freedom
```

```
## [1] 580
##
## $ttable
##     Estimate     SE  t.value p.value
## ar1   0.4594 0.0776   5.9193  0.0000
## ma1  -1.0586 0.0844 -12.5410  0.0000
## ma2   0.0586 0.0841   0.6969  0.4862
##
## $AIC
## [1] 1.223556
##
## $AICc
## [1] 1.223628
##
## $BIC
## [1] 1.253527
```

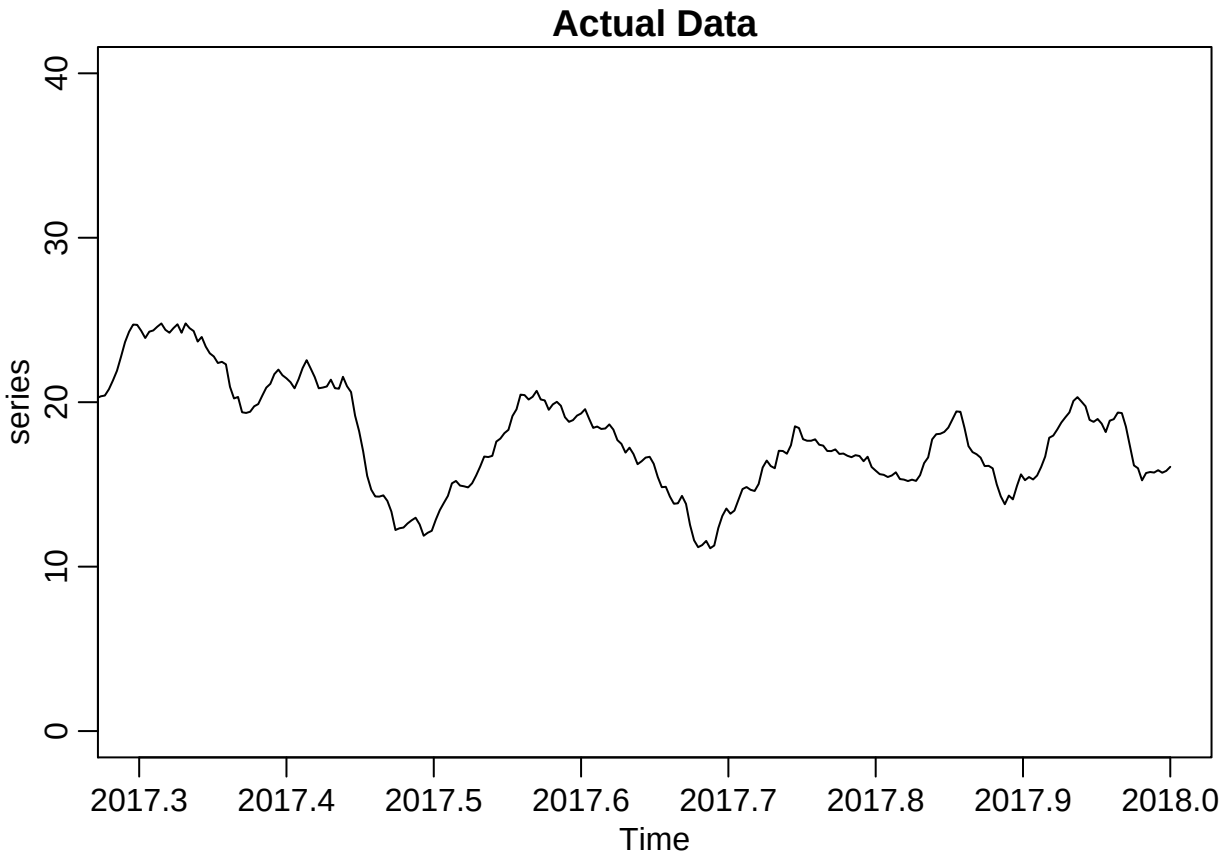
```
sarima.for(train, n.ahead=146, p=1, d=1, q=2, P=0, D=1, Q=0, S=1)
```




```
## $pred
## Time Series:
## Start = c(2017, 221)
## End = c(2018, 1)
## Frequency = 365
##   [1] 19.38891 19.44163 19.48358 19.52057 19.55530 19.58897 19.62217 19.65515
##   [9] 19.68802 19.72085 19.75366 19.78645 19.81925 19.85204 19.88483 19.91762
##  [17] 19.95041 19.98320 20.01599 20.04878 20.08157 20.11435 20.14714 20.17993
##  [25] 20.21272 20.24551 20.27830 20.31109 20.34388 20.37667 20.40946 20.44225
##  [33] 20.47504 20.50783 20.54062 20.57340 20.60619 20.63898 20.67177 20.70456
```

```
##  [41] 20.73735 20.77014 20.80293 20.83572 20.86851 20.90130 20.93409 20.96688
##  [49] 20.99967 21.03245 21.06524 21.09803 21.13082 21.16361 21.19640 21.22919
##  [57] 21.26198 21.29477 21.32756 21.36035 21.39314 21.42593 21.45872 21.49150
##  [65] 21.52429 21.55708 21.58987 21.62266 21.65545 21.68824 21.72103 21.75382
##  [73] 21.78661 21.81940 21.85219 21.88498 21.91777 21.95055 21.98334 22.01613
##  [81] 22.04892 22.08171 22.11450 22.14729 22.18008 22.21287 22.24566 22.27845
##  [89] 22.31124 22.34403 22.37682 22.40960 22.44239 22.47518 22.50797 22.54076
##  [97] 22.57355 22.60634 22.63913 22.67192 22.70471 22.73750 22.77029 22.80308
## [105] 22.83587 22.86865 22.90144 22.93423 22.96702 22.99981 23.03260 23.06539
## [113] 23.09818 23.13097 23.16376 23.19655 23.22934 23.26213 23.29492 23.32770
## [121] 23.36049 23.39328 23.42607 23.45886 23.49165 23.52444 23.55723 23.59002
## [129] 23.62281 23.65560 23.68839 23.72118 23.75397 23.78675 23.81954 23.85233
## [137] 23.88512 23.91791 23.95070 23.98349 24.01628 24.04907 24.08186 24.11465
## [145] 24.14744 24.18023
##
## $se
## Time Series:
## Start = c(2017, 221)
## End = c(2018, 1)
## Frequency = 365
##   [1]  0.4413833  0.7602561  1.0343208  1.2716707  1.4805496  1.6675942
##   [7]  1.8377341  1.9945511  2.1406527  2.2779620  2.4079226  2.5316398
##  [13]  2.6499766  2.7636190  2.8731219  2.9789407  3.0814545  3.1809827
##  [19]  3.2777973  3.3721318  3.4641889  3.5541455  3.6421573  3.7283620
##  [25]  3.8128822  3.8958275  3.9772964  4.0573779  4.1361526  4.2136938
##  [31]  4.2900683  4.3653373  4.4395572  4.5127797  4.5850525  4.6564199
##  [37]  4.7269229  4.7965995  4.8654854  4.9336137  5.0010152  5.0677190
##  [43]  5.1337523  5.1991406  5.2639081  5.3280772  5.3916694  5.4547048
##  [49]  5.5172026  5.5791808  5.6406566  5.7016461  5.7621648  5.8222275
##  [55]  5.8818480  5.9410397  5.9998153  6.0581869  6.1161661  6.1737638
##  [61]  6.2309908  6.2878571  6.3443724  6.4005460  6.4563868  6.5119034
##  [67]  6.5671040  6.6219966  6.6765887  6.7308876  6.7849003  6.8386337
##  [73]  6.8920943  6.9452884  6.9982220  7.0509010  7.1033311  7.1555178
##  [79]  7.2074663  7.2591818  7.3106692  7.3619333  7.4129787  7.4638099
##  [85]  7.5144313  7.5648470  7.6150612  7.6650778  7.7149007  7.7645335
##  [91]  7.8139800  7.8632435  7.9123276  7.9612356  8.0099707  8.0585360
##  [97]  8.1069345  8.1551693  8.2032432  8.2511591  8.2989196  8.3465275
## [103]  8.3939854  8.4412957  8.4884609  8.5354835  8.5823658  8.6291101
## [109]  8.6757186  8.7221935  8.7685368  8.8147508  8.8608374  8.9067985
## [115]  8.9526362  8.9983523  9.0439486  9.0894270  9.1347891  9.1800369
## [121]  9.2251718  9.2701956  9.3151099  9.3599163  9.4046162  9.4492113
## [127]  9.4937029  9.5380926  9.5823817  9.6265716  9.6706638  9.7146595
## [133]  9.7585600  9.8023666  9.8460806  9.8897032  9.9332356  9.9766790
## [139] 10.0200345 10.0633033 10.1064864 10.1495851 10.1926002 10.2355330
## [145] 10.2783845 10.3211556
```

```
x_lim = c(2017.3, 2018.0)
y_lim = c(0,40)
plot(series, xlim=x_lim, ylim=y_lim, main='Actual Data')
```



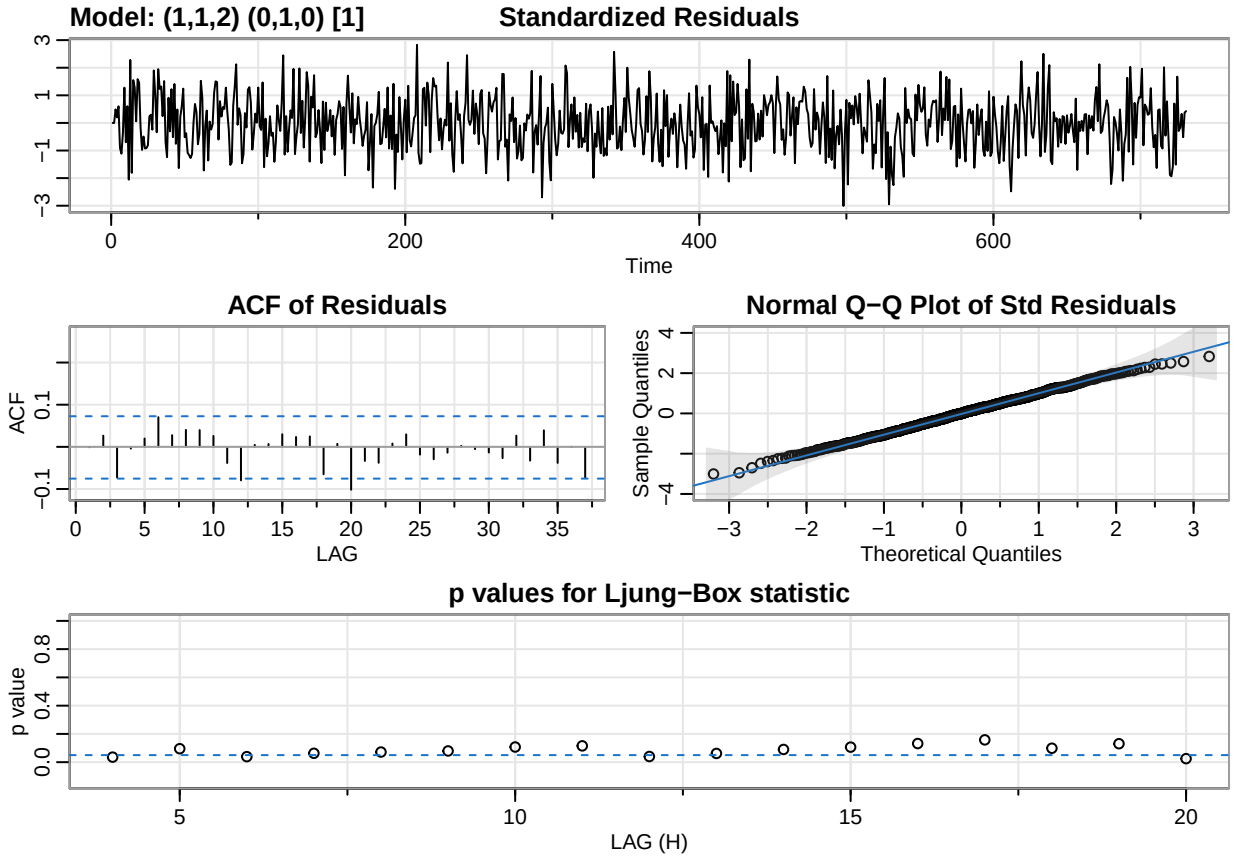
Model

As we can see our forecast is more or less in line with our actual data. We'll use this model for a main forecast. We'll use the parameters from `auto.arima` to build a seasonal ARIMA model with `sarima` and predict an interval of 132 days, or roughly six months.

```
sarima(df$Revenue, p=1, d=1, q=2, P=0, D=1, Q=0, S=1)
```

```
## initial  value -0.644936
## iter   2 value -0.710090
## iter   3 value -0.770739
## iter   4 value -0.776089
## iter   5 value -0.783746
## iter   6 value -0.787520
## iter   7 value -0.789102
## iter   8 value -0.789439
## iter   9 value -0.790611
## iter  10 value -0.794665
## iter  11 value -0.813288
## iter  12 value -0.815243
## iter  13 value -0.815742
## iter  14 value -0.815810
## iter  14 value -0.815810
## final  value -0.815810
## converged
## initial  value -0.814229
## iter   2 value -0.814428
```

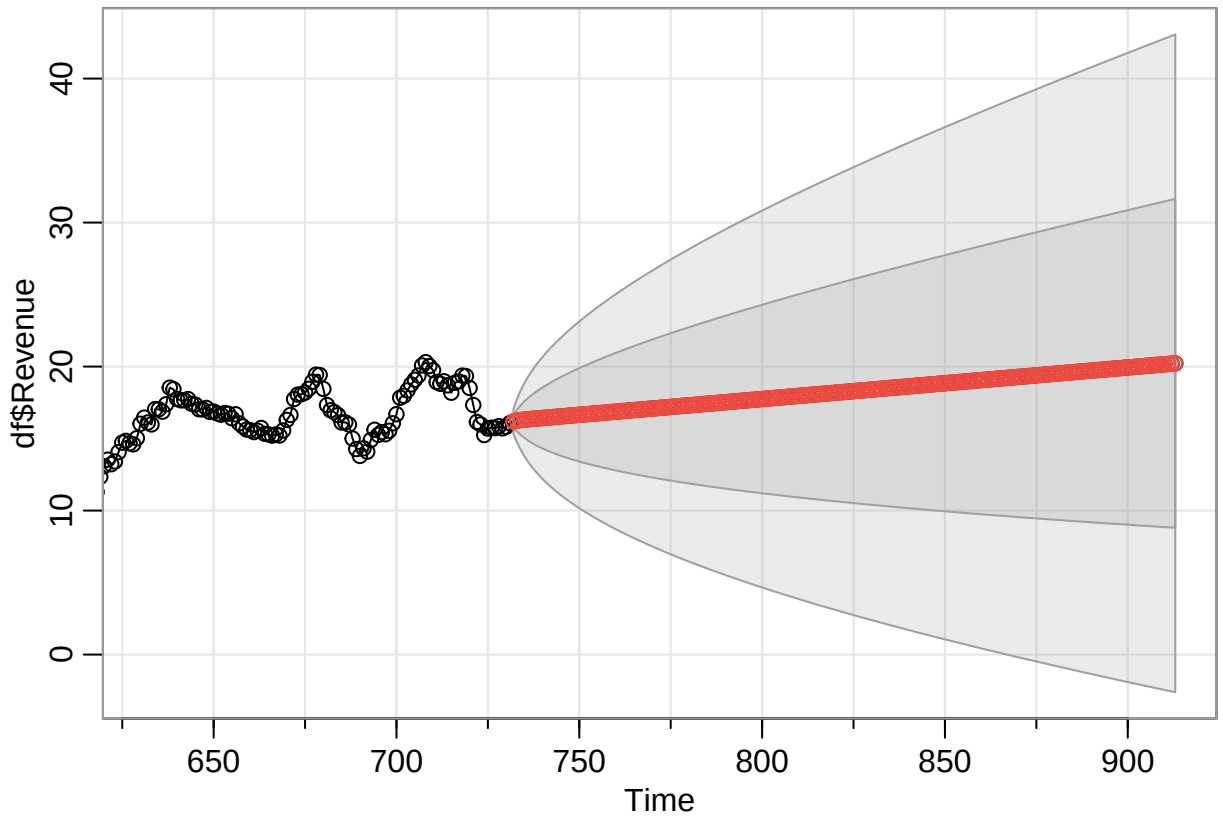
```
## iter   3 value -0.814544
## iter   4 value -0.814637
## iter   5 value -0.814833
## iter   6 value -0.814900
## iter   7 value -0.814940
## iter   8 value -0.814943
## iter   9 value -0.814943
## iter  10 value -0.814943
## iter  10 value -0.814943
## iter  10 value -0.814943
## final  value -0.814943
## converged
```




```
## $fit
##
## Call:
## arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D, Q), period = S),
##     include.mean = !no.constant, transform.pars = trans, fixed = fixed, optim.control = list(trace =
##         REPORT = 1, reltol = tol))
##
## Coefficients:
##          ar1      ma1     ma2
##       0.4316  -1.0201  0.0201
## s.e.  0.0727   0.0790  0.0786
##
## sigma^2 estimated as 0.1944:  log likelihood = -440.31,  aic = 888.63
##
```

```
## $degrees_of_freedom
## [1] 726
##
## $ttable
##     Estimate     SE  t.value p.value
## ar1   0.4316 0.0727   5.9405  0.0000
## ma1  -1.0201 0.0790 -12.9177  0.0000
## ma2   0.0201 0.0786   0.2558  0.7982
##
## $AIC
## [1] 1.218965
##
## $AICc
## [1] 1.21901
##
## $BIC
## [1] 1.244159
```

```
sarima.for(df$Revenue, n.ahead=182, p=1, d=1, q=2, P=0, D=1, Q=0, S=1)
```




```
## $pred
## Time Series:
## Start = 732
## End = 913
## Frequency = 1
##   [1] 16.18453 16.24667 16.28595 16.31537 16.34053 16.36385 16.38638 16.40856
##   [9] 16.43060 16.45257 16.47452 16.49645 16.51838 16.54030 16.56223 16.58415
##  [17] 16.60608 16.62800 16.64993 16.67185 16.69378 16.71570 16.73763 16.75955
##  [25] 16.78148 16.80340 16.82533 16.84725 16.86917 16.89110 16.91302 16.93495
```

```
##  [33] 16.95687 16.97880 17.00072 17.02265 17.04457 17.06650 17.08842 17.11034
##  [41] 17.13227 17.15419 17.17612 17.19804 17.21997 17.24189 17.26382 17.28574
##  [49] 17.30767 17.32959 17.35151 17.37344 17.39536 17.41729 17.43921 17.46114
##  [57] 17.48306 17.50499 17.52691 17.54884 17.57076 17.59268 17.61461 17.63653
##  [65] 17.65846 17.68038 17.70231 17.72423 17.74616 17.76808 17.79001 17.81193
##  [73] 17.83385 17.85578 17.87770 17.89963 17.92155 17.94348 17.96540 17.98733
##  [81] 18.00925 18.03118 18.05310 18.07502 18.09695 18.11887 18.14080 18.16272
##  [89] 18.18465 18.20657 18.22850 18.25042 18.27235 18.29427 18.31619 18.33812
##  [97] 18.36004 18.38197 18.40389 18.42582 18.44774 18.46967 18.49159 18.51352
## [105] 18.53544 18.55736 18.57929 18.60121 18.62314 18.64506 18.66699 18.68891
## [113] 18.71084 18.73276 18.75469 18.77661 18.79853 18.82046 18.84238 18.86431
## [121] 18.88623 18.90816 18.93008 18.95201 18.97393 18.99586 19.01778 19.03970
## [129] 19.06163 19.08355 19.10548 19.12740 19.14933 19.17125 19.19318 19.21510
## [137] 19.23703 19.25895 19.28087 19.30280 19.32472 19.34665 19.36857 19.39050
## [145] 19.41242 19.43435 19.45627 19.47820 19.50012 19.52205 19.54397 19.56589
## [153] 19.58782 19.60974 19.63167 19.65359 19.67552 19.69744 19.71937 19.74129
## [161] 19.76322 19.78514 19.80706 19.82899 19.85091 19.87284 19.89476 19.91669
## [169] 19.93861 19.96054 19.98246 20.00439 20.02631 20.04823 20.07016 20.09208
## [177] 20.11401 20.13593 20.15786 20.17978 20.20171 20.22363
##
## $se
## Time Series:
## Start = 732
## End = 913
## Frequency = 1
##   [1]  0.4412402  0.7637736  1.0378170  1.2731336  1.4791642  1.6631485
##   [7]  1.8302757  1.9842190  2.1276032  2.2623397  2.3898516  2.5112235
##  [13]  2.6272996  2.7387506  2.8461187  2.9498493  3.0503133  3.1478233
##  [19]  3.2426459  3.3350102  3.4251153  3.5131347  3.5992217  3.6835117
##  [25]  3.7661253  3.8471706  3.9267446  4.0049351  4.0818215  4.1574763
##  [31]  4.2319654  4.3053494  4.3776838  4.4490198  4.5194048  4.5888825
##  [37]  4.6574934  4.7252754  4.7922636  4.8584909  4.9239878  4.9887834
##  [43]  5.0529044  5.1163762  5.1792228  5.2414666  5.3031289  5.3642296
##  [49]  5.4247878  5.4848215  5.5443476  5.6033824  5.6619412  5.7200387
##  [55]  5.7776887  5.8349046  5.8916989  5.9480837  6.0040707  6.0596707
##  [61]  6.1148944  6.1697519  6.2242529  6.2784065  6.3322219  6.3857074
##  [67]  6.4388714  6.4917216  6.5442658  6.5965113  6.6484650  6.7001337
##  [73]  6.7515241  6.8026424  6.8534947  6.9040869  6.9544246  7.0045134
##  [79]  7.0543586  7.1039652  7.1533382  7.2024825  7.2514027  7.3001033
##  [85]  7.3485886  7.3968630  7.4449304  7.4927950  7.5404604  7.5879306
##  [91]  7.6352091  7.6822995  7.7292052  7.7759296  7.8224759  7.8688473
##  [97]  7.9150468  7.9610774  8.0069421  8.0526437  8.0981849  8.1435684
## [103]  8.1887969  8.2338730  8.2787990  8.3235774  8.3682107  8.4127011
## [109]  8.4570508  8.5012621  8.5453372  8.5892780  8.6330867  8.6767653
## [115]  8.7203157  8.7637398  8.8070395  8.8502167  8.8932730  8.9362103
## [121]  8.9790303  9.0217346  9.0643249  9.1068028  9.1491697  9.1914274
## [127]  9.2335772  9.2756207  9.3175592  9.3593943  9.4011272  9.4427593
## [133]  9.4842920  9.5257266  9.5670642  9.6083063  9.6494539  9.6905084
## [139]  9.7314709  9.7723425  9.8131243  9.8538176  9.8944234  9.9349427
## [145]  9.9753766 10.0157262 10.0559924 10.0961763 10.1362789 10.1763010
## [151] 10.2162437 10.2561078 10.2958944 10.3356042 10.3752382 10.4147972
## [157] 10.4542821 10.4936938 10.5330329 10.5723004 10.6114971 10.6506237
## [163] 10.6896809 10.7286697 10.7675906 10.8064444 10.8452319 10.8839538
## [169] 10.9226107 10.9612034 10.9997325 11.0381987 11.0766026 11.1149449
```

```
## [175] 11.1532262 11.1914472 11.2296085 11.2677106 11.3057542 11.3437398
## [181] 11.3816681 11.4195396
```

Our forecast predicts a general increase of revenue over the upcoming year.

Summary

We selected our ARIMA model by first using `auto.arima` to find the best parameters. We then tested the parameters against real data using a train/test split and concluded the parameters could be used. We then applied the parameters to a seasonal ARIMA model using the full data to predict the next six months of revenue. We chose six months because it is a quarter the size of the given data, a reasonable interval to test over.

Since the company is projected to make a profit it is recommended that the budget is adjusted accordingly. This may be a good time to shore up areas of the company that need investment since we know there will be income to spare.