The following data are monthly Australian sales of fortified Apple juice in thousands of liters from Jan 2005 to Jul 2020.

2585 3368 3210 3111 3756 4216 5225 4426 3932 3816 3661 3795 2285 2934 2985 3646 4198 4935 5618 5454 3624 2898 3802 3629 2369 2511 3079 3728 4151 4326 5054 5138 3310 3508 3790 3446 2127 2523 3017 3265 3822 4027 4420 5255 4009 3074 3465 3718 1954 2604 3626 2836 4042 3584 4225 4523 2892 2876 3420 3159 2101 2181 2724 2954 4092 3470 3990 4239 2855 2897 3433 3307 1914 2214 2320 2714 3633 3295 4377 4442 2774 2840 2828 3758 1610 1968 2248 3262 3164 2972 4041 3402 2898 2555 3056 3717 1755 2193 2198 2777 3076 3389 4231 3118 2524 2280 2862 3502 1558 1940 2226 2676 3145 3224 4117 3446 2482 2349 2986 3163 1651 1725 2622 2316 2976 3263 3951 2917 2380 2458 2883 2579 1330 1686 2457 2514 2834 2757 3425 3006 2369 2017 2507 3168 1545 1643 2112 2415 2862 2822 3260 2606 2264 2250 2545 2856 1208 1412 1964 2018 2329 2660 2923 2626 2132 1772 2526 2755 1154 1568 1965 2659 2354 2592 2714 2294 2416 2016 2799 2467 1153 1482 1818 2262 2612 2967 3179

Fit an ARIMA model to the data up to December 2019 and use it to make forecasts for 2020. You should follow the procedure below, but present each step as your own idea giving a reason from one step to the next.

1. Specify a tentative ARIMA model for this series by examining together the sample acf and pacf of the data suitably differenced.

2. Estimate the parameters of the specified model.

3. Examine the residuals of the fitted model, check in particular correlations and Gaussianity of the residuals.

4. Go back to step 1 if you find any evidence against model adequacy.

5. Reduce the number of parameters in the model if possible.

6. Make forecasts using the fitted model. Give prediction intervals.

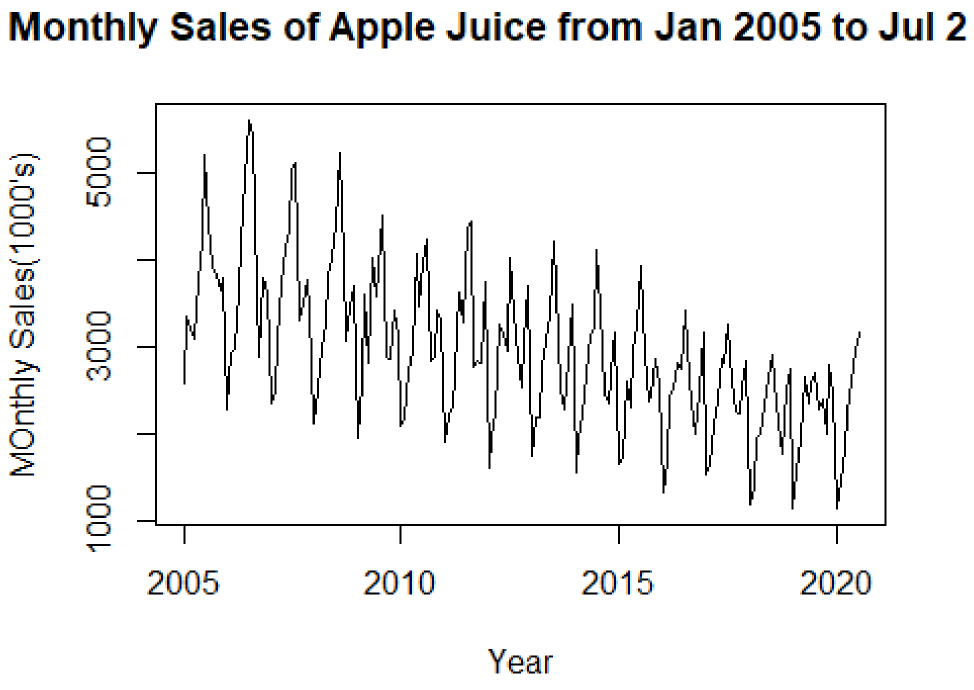
```
library(ggfortify)

## Loading required package: ggplot2

library(tseries)

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

library(forecast)

## Registered S3 methods overwritten by 'forecast':
##   method                 from
##   autoplot.Arima         ggfortify
##   autoplot.acf           ggfortify
##   autoplot.ar            ggfortify
##   autoplot.bats          ggfortify
##   autoplot.decomposed.ts ggfortify
##   autoplot.ets           ggfortify
##   autoplot.forecast      ggfortify
##   autoplot.stl           ggfortify
##   autoplot.ts            ggfortify
##   fitted.ar              ggfortify
##   fortify.ts             ggfortify
##   residuals.ar           ggfortify

p<-
c(2585,3368,3210,3111,3756,4216,5225,4426,3932,3816,3661,3795,2285,2934,2985,
3646,4198,4935,5618,5454,3624,2898,3802,3629,2369,2511,3079,3728,4151,4326,
5054,5138,3310,3508,3790,3446,2127,2523,3017,3265,3822,4027,4420,5255,4009,
3074,3465,3718,1954,2604,3626,2836,4042,3584,4225,4523,2892,2876,3420,3159,
2101,2181,2724,2954,4092,3470,3990,4239,2855,2897,3433,3307,1914,2214,2320,
2714,3633,3295,4377,4442,2774,2840,2828,3758,1610,1968,2248,3262,3164,2972
,4041,3402,2898,2555,3056,3717,1755,2193,2198,2777,3076,3389,4231,3118,2524,
2280,2862,3502,1558,1940,2226,2676,3145,3224,4117,3446,2482,2349,2986,3163,
1651,1725,2622,2316,2976,3263,3951,2917,2380,2458,2883,2579,1330,1686,2457,
2514,2834,2757,3425,3006,2369,2017,2507,3168,1545,1643,2112,2415,2862,2822,
3260,2606,2264,2250,2545,2856,1208,1412,1964,2018,2329,2660,2923,2626,2132,
1772,2526,2755,1154,1568,1965,2659,2354,2592,2714,2294,2416,2016,2799,2467,
1153,1482,1818, 2262,2612,2967,3179)
q<-ts(p,start=c(2005,1),end=c(2020,7),frequency=12)
class(q)

## [1] "ts"
```

The Monthly Sales of Apple Juice dataset in R provides monthly totals of a Monthly Sales of Apple Juice in Australia, from Jan 2005 to Jul 2020. This dataset is already of a time series class therefore no further class or date manipulation is required.

To perform exploratory analysis, let’s first review the data with summary statistics and plots in R.

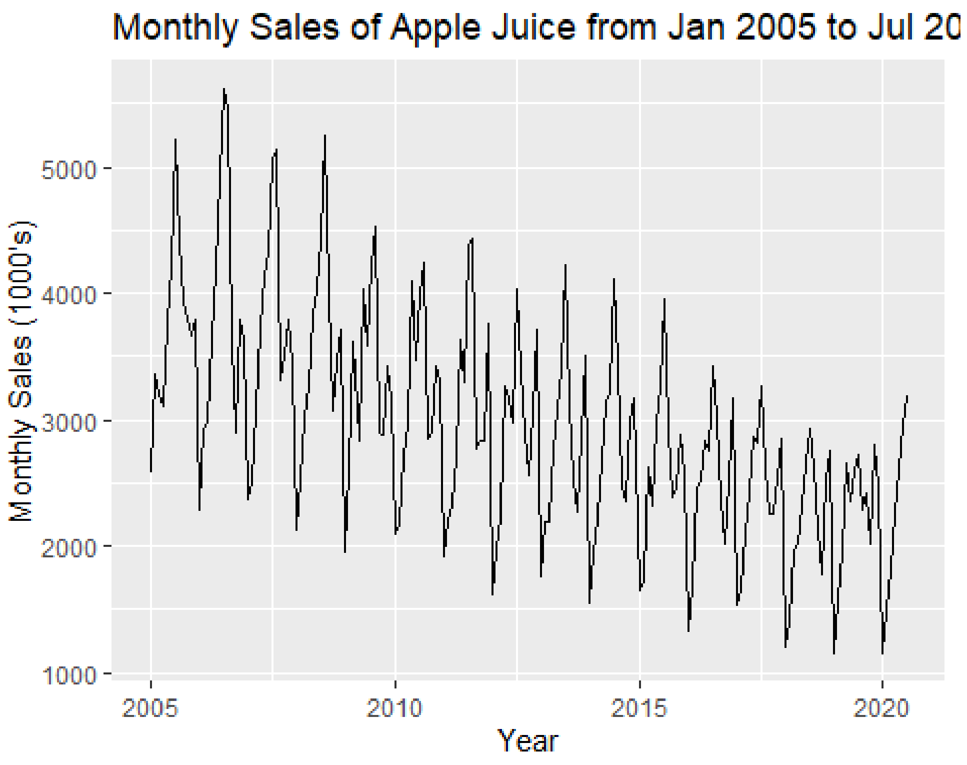
```
q

##       Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov  Dec
## 2005 2585 3368 3210 3111 3756 4216 5225 4426 3932 3816 3661 3795
## 2006 2285 2934 2985 3646 4198 4935 5618 5454 3624 2898 3802 3629
## 2007 2369 2511 3079 3728 4151 4326 5054 5138 3310 3508 3790 3446
## 2008 2127 2523 3017 3265 3822 4027 4420 5255 4009 3074 3465 3718
## 2009 1954 2604 3626 2836 4042 3584 4225 4523 2892 2876 3420 3159
## 2010 2101 2181 2724 2954 4092 3470 3990 4239 2855 2897 3433 3307
## 2011 1914 2214 2320 2714 3633 3295 4377 4442 2774 2840 2828 3758
## 2012 1610 1968 2248 3262 3164 2972 4041 3402 2898 2555 3056 3717
## 2013 1755 2193 2198 2777 3076 3389 4231 3118 2524 2280 2862 3502
## 2014 1558 1940 2226 2676 3145 3224 4117 3446 2482 2349 2986 3163
## 2015 1651 1725 2622 2316 2976 3263 3951 2917 2380 2458 2883 2579
## 2016 1330 1686 2457 2514 2834 2757 3425 3006 2369 2017 2507 3168
## 2017 1545 1643 2112 2415 2862 2822 3260 2606 2264 2250 2545 2856
## 2018 1208 1412 1964 2018 2329 2660 2923 2626 2132 1772 2526 2755
## 2019 1154 1568 1965 2659 2354 2592 2714 2294 2416 2016 2799 2467
## 2020 1153 1482 1818 2262 2612 2967 3179

# Check for missing values
sum(is.na(q))

## [1] 0

# Check the frequency of the time series
frequency(q)

## [1] 12

# Check the cycle of the time series
cycle(q)

##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 2005   1   2   3   4   5   6   7   8   9  10  11  12
## 2006   1   2   3   4   5   6   7   8   9  10  11  12
## 2007   1   2   3   4   5   6   7   8   9  10  11  12
## 2008   1   2   3   4   5   6   7   8   9  10  11  12
## 2009   1   2   3   4   5   6   7   8   9  10  11  12
## 2010   1   2   3   4   5   6   7   8   9  10  11  12
## 2011   1   2   3   4   5   6   7   8   9  10  11  12
## 2012   1   2   3   4   5   6   7   8   9  10  11  12
## 2013   1   2   3   4   5   6   7   8   9  10  11  12
## 2014   1   2   3   4   5   6   7   8   9  10  11  12
## 2015   1   2   3   4   5   6   7   8   9  10  11  12
## 2016   1   2   3   4   5   6   7   8   9  10  11  12
## 2017   1   2   3   4   5   6   7   8   9  10  11  12
## 2018   1   2   3   4   5   6   7   8   9  10  11  12
## 2019   1   2   3   4   5   6   7   8   9  10  11  12
## 2020   1   2   3   4   5   6   7
```

```r
# Review the table summary
summary(q)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1153    2362    2883    2969    3486    5618

# Plot the raw data using the base plot function
plot(q,xlab="Year", ylab = "MOnthly Sales(1000's)",main="Monthly Sales of
Apple Juice from Jan 2005 to Jul 2020")
```



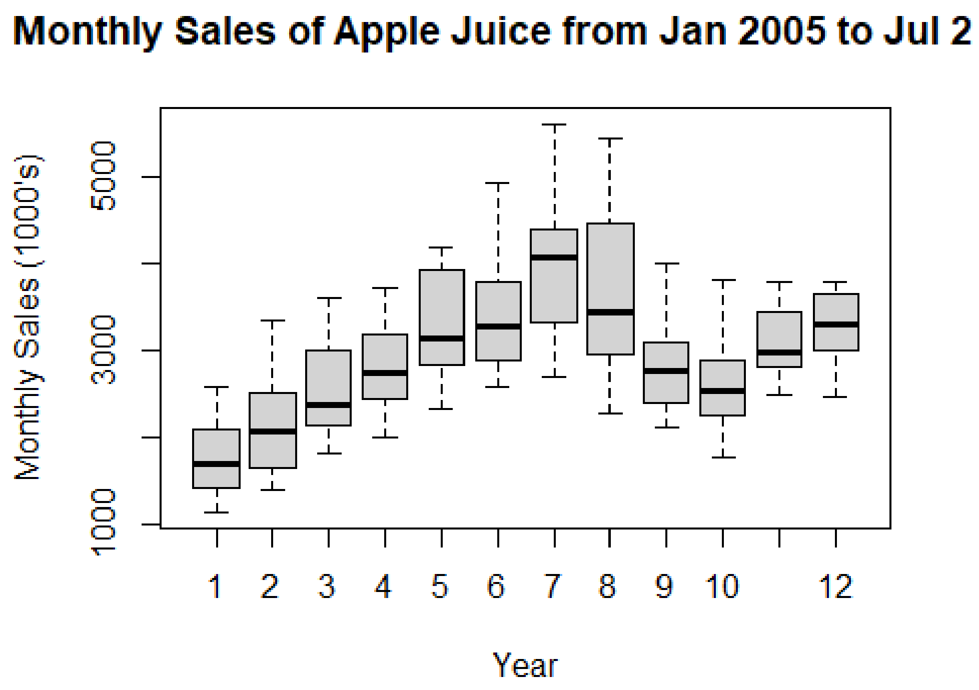
As an alternative to the base plot function, so we can also use the extension ggfortify R package from the ggplot2 package, to plot directly from a time series. The benefits are not having to convert to a dataframe as required with ggplot2, but still having access to the layering grammar of graphics.

```r
autoplot(q) + labs(x ="Year", y = "Monthly Sales (1000's)", title="Monthly
Sales of Apple Juice from Jan 2005 to Jul 2020")
```



Let’s use the boxplot function to see any seasonal effects.

```
boxplot(q~cycle(q),xlab="Year", ylab = "Monthly Sales (1000's)" ,main
="Monthly Sales of Apple Juice from Jan 2005 to Jul 2020")
```



From these exploratory plots, we can make some initial inferences:

- The mnonthly sales increase over time with each year which may be indicative of an increasing linear trend, perhaps due to increasing demand for flight travel and commercialisation of apples juicesin that time period.
- In the boxplot there are more sales in months 5 to 8 with higher means and higher variances than the other months, indicating seasonality with a apparent cycle of 12 months. The rationale for this could be more people taking holidays and fly over the summer months in the Australia.
- Monthly sales appears to be multiplicative time series as the passenger numbers increase, it appears so does the pattern of seasonality.
- There do not appear to be any outliers and there are no missing values. Therefore no data cleaning is required.

TIME SERIES DECOMPOSITION

We will decompose the time series for estimates of trend, seasonal, and random components using moving average method.

The multiplicative model is:

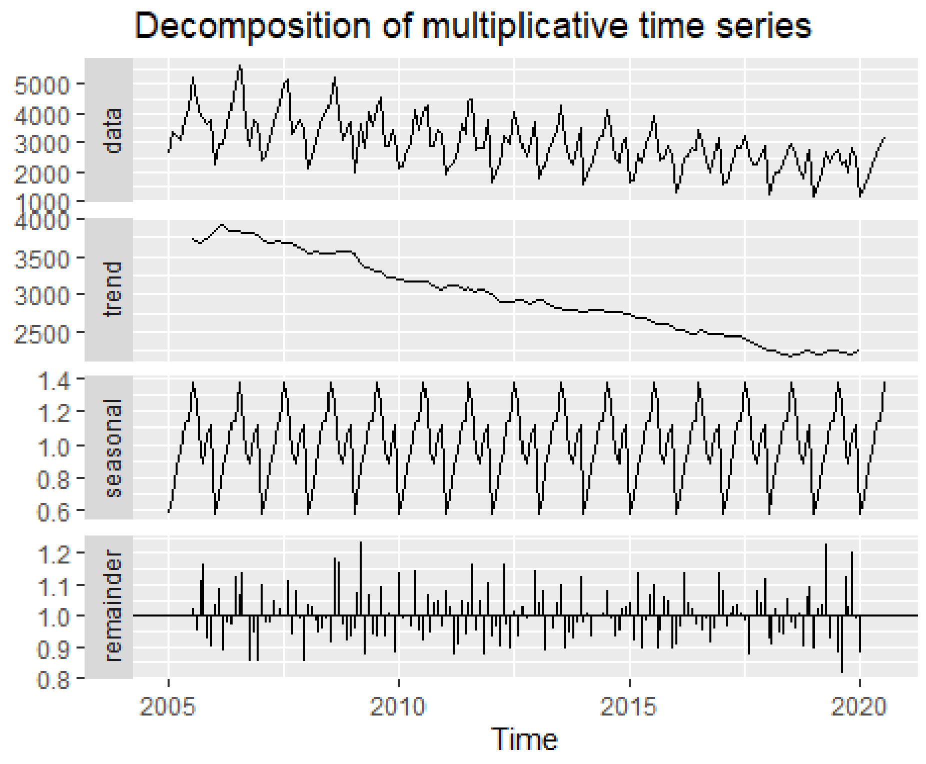
$$Y[t]=T[t]*S[t]*e[t]$$

where

- Y(t) is the number of passengers at time t,
- T(t) is the trend component at time t,
- S(t) is the seasonal component at time t,
- e(t) is the random error component at time t.

With this model, we will use the decompose function in R. Continuing to use ggfortify for plots, in one line, autoplot these decomposed components to further analyse the data.

```
decomposeq <- decompose(q,"multiplicative")
autoplot(decomposeq)
```



In these decomposed plots we can again see the trend and seasonality as inferred previously, but we can also observe the estimation of the random component depicted under the “remainder”.

TEST STATIONARITY OF THE TIME SERIES

A stationary time series has the conditions that the mean, variance and covariance are not functions of time. In order to fit arima models, the time series is required to be stationary. We will use two methods to test the stationarity.

1. Test stationarity of the time series (ADF)

In order to test the stationarity of the time series, let’s run the Augmented Dickey-Fuller Test using the adf.test function from the tseries R package.

First set the hypothesis test:

The null hypothesis H_o: that the time series is non stationary
The alternative hypothesis H_A : that the time series is stationary

```
adf.test(q)
```

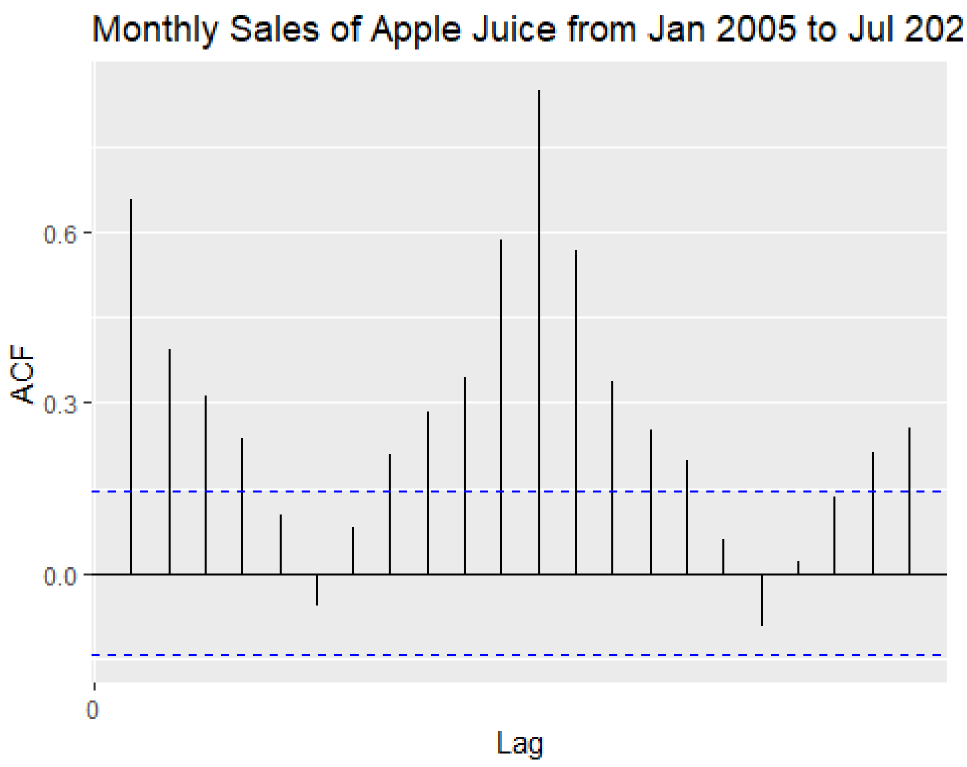
```
## Warning in adf.test(q): p-value smaller than printed p-value

##
##  Augmented Dickey-Fuller Test
##
## data:  q
## Dickey-Fuller = -12.681, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

As a rule of thumb, where the p-value is less than 5%, we strong evidence against the null hypothesis, so we reject the null hypothesis. As per the test results above, the p-value is 0.01 which is <0.05 therefore we reject the null in favour of the alternative hypothesis that the time series is stationary.

Test stationarity of the time series (Autocorrelation)

Another way to test for stationarity is to use autocorrelation. We will use autocorrelation function (acf) in from the base stats R package. This function plots the correlation between a series and its lags ie previous observations with a 95% confidence interval in blue. If the autocorrelation crosses the dashed blue line, it means that specific lag is significantly correlated with current series.

```
autoplot(acf(q,plot=FALSE))+ labs(title="Monthly Sales of Apple Juice from
Jan 2005 to Jul 2020")
```



The maximum at lag 1 or 12 months, indicates a positive relationship with the 12 month cycle.

Since we have already created the decomposeAP list object with a random component, we can plot the acf of the decomposeq$random.

```
# Review random time series for any missing values
decomposeq$random
```

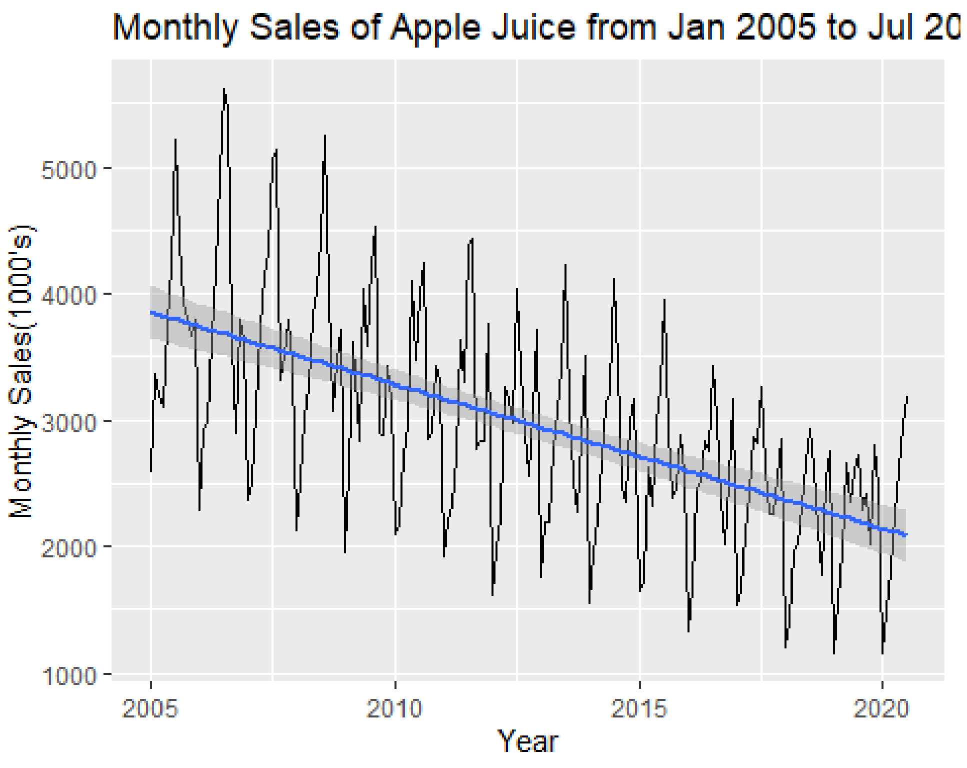
```
##            Jan       Feb       Mar       Apr       May       Jun       Jul
## 2005        NA        NA        NA        NA        NA        NA 1.0189660
## 2006 1.0332183 1.0874332 0.8874214 0.9763452 0.9709745 1.1219590 1.0694608
## 2007 1.0980961 0.9783326 0.9765891 1.0477004 0.9924917 1.0186332 1.0003503
## 2008 1.0366383 1.0296790 0.9857818 0.9454977 0.9564294 0.9908793 0.9087903
## 2009 0.9572652 1.0734963 1.2354430 0.8730057 1.0689272 0.9384075 0.9302468
## 2010 1.1359528 0.9881331 1.0017710 0.9659963 1.1456212 0.9526309 0.9168469
## 2011 1.0766554 1.0283768 0.8696167 0.9060767 1.0481582 0.9361151 1.0382276
## 2012 0.9289853 0.9635161 0.9012862 1.1653726 0.9689625 0.8919929 1.0131223
## 2013 1.0392413 1.0822991 0.8851169 1.0035354 0.9586138 1.0440925 1.0971236
## 2014 0.9759768 1.0082446 0.9311192 0.9947261 0.9984420 1.0090115 1.0819000
## 2015 1.0410016 0.9149080 1.1350395 0.8912145 0.9807779 1.0680625 1.0974277
## 2016 0.9082037 0.9651653 1.1354420 1.0406112 1.0185825 0.9702500 0.9949244
## 2017 1.0788913 0.9640745 1.0103500 1.0248469 1.0354868 1.0080036 0.9850543
## 2018 0.9243394 0.9045350 1.0190776 0.9415322 0.9394664 1.0567856 0.9743587
## 2019 0.8922691 1.0193192 1.0334582 1.2310518 0.9245787 1.0006890 0.8812498
## 2020 0.8813149        NA        NA        NA        NA        NA        NA
##            Aug       Sep       Oct       Nov       Dec
## 2005 0.9519076 1.1123286 1.1630101 0.9268886 0.8953102
## 2006 1.1398545 0.9924234 0.8563395 0.9431866 0.8558654
## 2007 1.1152951 0.9386078 1.0818814 0.9906030 0.8563192
## 2008 1.1831304 1.1689790 0.9670280 0.9176950 0.9318526
## 2009 1.0930893 0.9280147 1.0078714 1.0043906 0.8763920
## 2010 1.0676144 0.9433936 1.0437971 1.0485877 0.9615976
## 2011 1.1611490 0.9508611 1.0456396 0.8735354 1.1075659
## 2012 0.9280368 1.0295897 0.9888720 1.0015647 1.1443960
## 2013 0.8903042 0.9441123 0.9230325 0.9735617 1.1260232
## 2014 0.9923527 0.9306544 0.9517207 1.0241710 1.0260004
## 2015 0.8913309 0.9525945 1.0630604 1.0462516 0.8926301
## 2016 0.9524282 0.9863156 0.9146374 0.9559123 1.1383742
## 2017 0.8698392 0.9931845 1.0773997 1.0406148 1.1161852
## 2018 0.9556371 1.0099693 0.8966342 1.0602675 1.0923497
## 2019 0.8160798 1.1270208 1.0271255 1.2008186 0.9871876
## 2020
```

FIT A TIME SERIES MODEL

Linear Model

Since there is an upwards trend we will look at a linear model first for comparison. We plot Monthly Sales raw dataset with a blue linear model.

```
autoplot(q) + geom_smooth(method="lm")+ labs(x ="Year", y = "Monthly
Sales(1000's)", title="Monthly Sales of Apple Juice from Jan 2005 to Jul
2020")

## `geom_smooth()` using formula = 'y ~ x'
```



This may not be the best model to fit as it doesn't capture the seasonality and multiplicative effects over time.

ARIMA Model

Use the auto.arima function from the forecast R package to fit the best model and coefficients, given the default parameters including seasonality as TRUE. Note we have used the ARIMA modeling procedure as referenced

```
arimaq <- auto.arima(q)
arimaq

## Series: q
## ARIMA(0,0,0)(2,1,1)[12] with drift
##
## Coefficients:
##          sar1    sar2     sma1   drift
##        0.2632  0.0834  -0.7876  -9.458
## s.e.   0.1810  0.1277   0.1601   0.792
```

```
##
## sigma^2 = 89789:  log likelihood = -1247.06
## AIC=2504.12   AICc=2504.47   BIC=2519.94
```

The ARIMA(2,1,1)(0,1,0)[12] model parameters are lag 1 differencing (d), an autoregressive term of second lag (p) and a moving average model of order 1 (q). Then the seasonal model has an autoregressive term of first lag (D) at model period 12 units, in this case months.

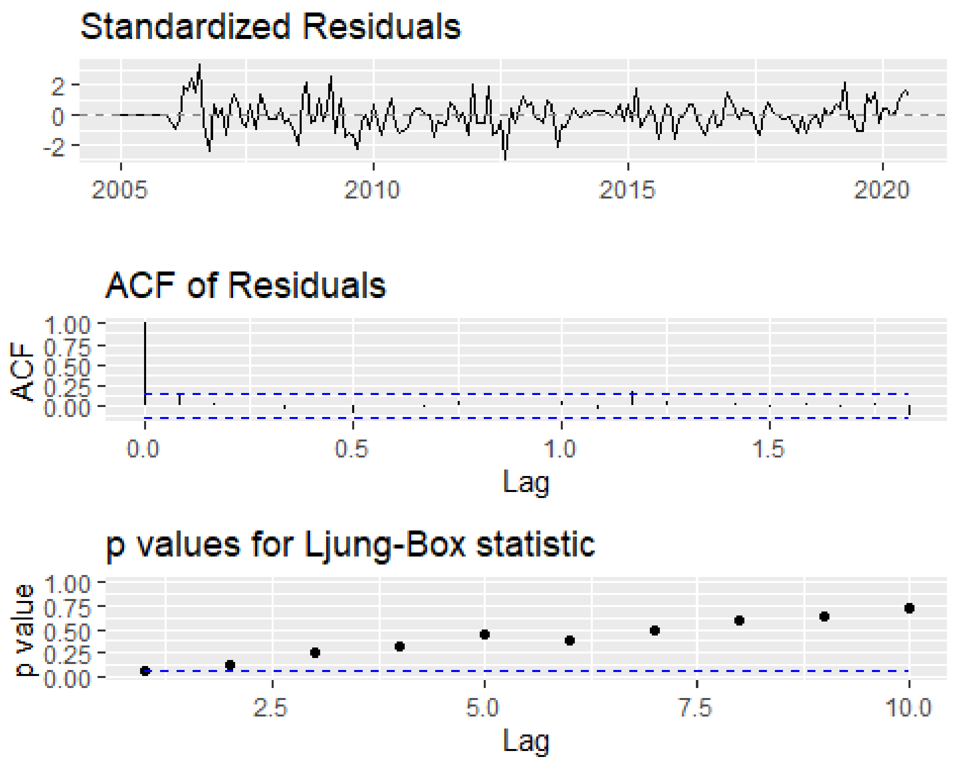
The ARIMA fitted model is:

$$\hat{Y} = 0.2632Y_{t-2} + 0.0834Y_{t-12} - 0.7876e_{t-1} + E$$

where E is some error.

The ggtsdiag function from ggfortify R package performs model diagnostics of the residuals and the acf. will include a autocovariance plot.

```
ggtsdiag(arimaq)
```

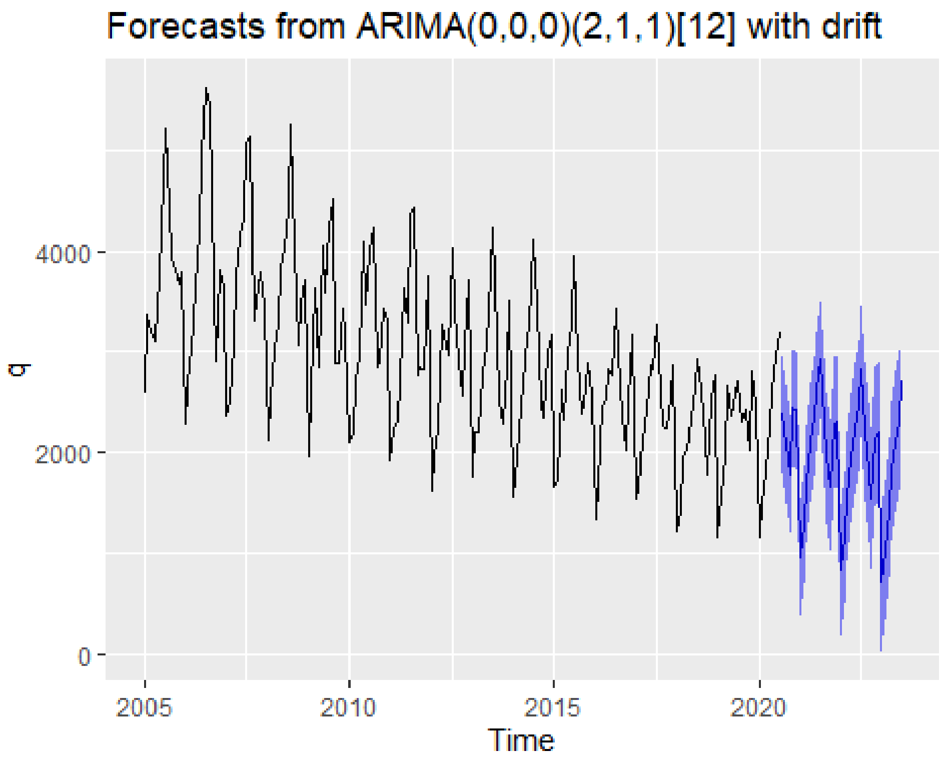


The residual plots appear to be centered around 0 as noise, with no pattern. the arima model is a fairly good fit.

CALCULATE FORECASTS

Finally we can plot a forecast of the time series using the forecast function, again from the forecast R package, with a 95% confidence interval where h is the forecast horizon periods in months.

```
forecastAP <- forecast(arimaAP, level = c(95), h = 36)
autoplot(forecastAP)
```



To summarize, this has been an exercise in ARIMA modeling and using time series R packages ggfortify, tseries and forecast. It is a good basis to move on to more complicated time series datasets, models and comparisons in R.