Colgate-Crest

Code

Arturo Sánchez Palacio

Diclaimer

Though the following code is perfectly valid for the analysis of the Colgate and Crest's market shares between 1958 and 1963 this code is not merely a study over this period but also an introduction to the study of time series. Because of this, some parts of this code may look tedious or redundant for and advanced programmer or data analyst.

Loading and exploring the data.

We start loading the data from the .xlsx file:

Hide

```
library(openxlsx)
data <- read.xlsx("data.xlsx", colNames = TRUE, detectDates = TRUE)
head(data)
```

Hide

```
tail(data)
```

Once we have loaded the data we chech for non-defined values

Hide

```
sum(is.na(data))
```

```
[1] 0
```

The data is complete.

To get a better idea of the data we build two time series one for Colgate and one for Crest:

Hide

```
colgate <- ts(data$Colgate, start = 1958, frequency = 52)
crest <- ts(data$Crest, start = 1958, frequency = 52)
```

We also create an object where we store both series:

Hide

```
mydata <- ts(data[,c(3,4)], start = 1958, frequency = 52)
```

Once the time series are built we can plot the market shares along the time:

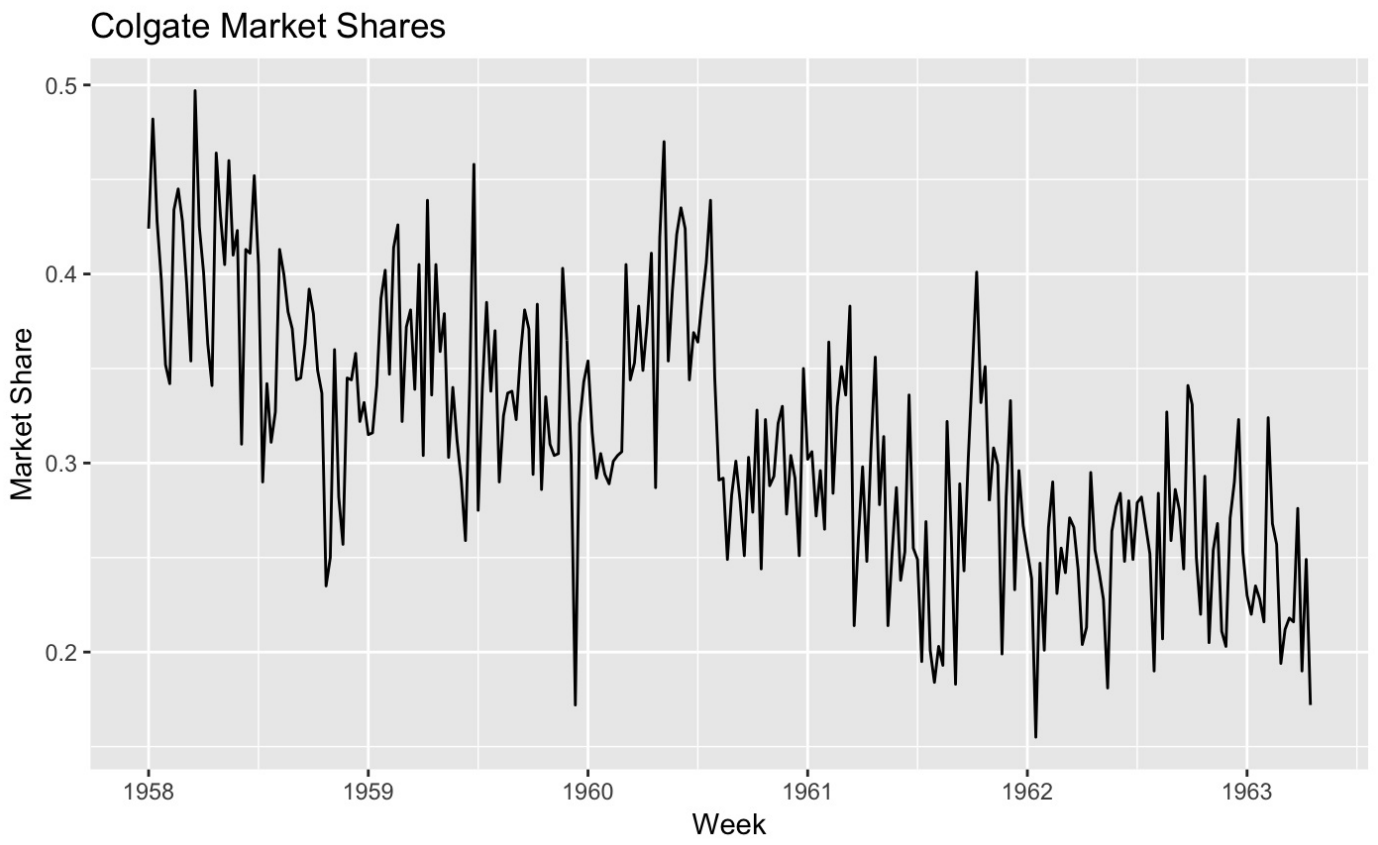
Hide

```
library(fpp2)
```

```
Loading required package: ggplot2
Loading required package: forecast
Loading required package: fma
Loading required package: expsmooth
```

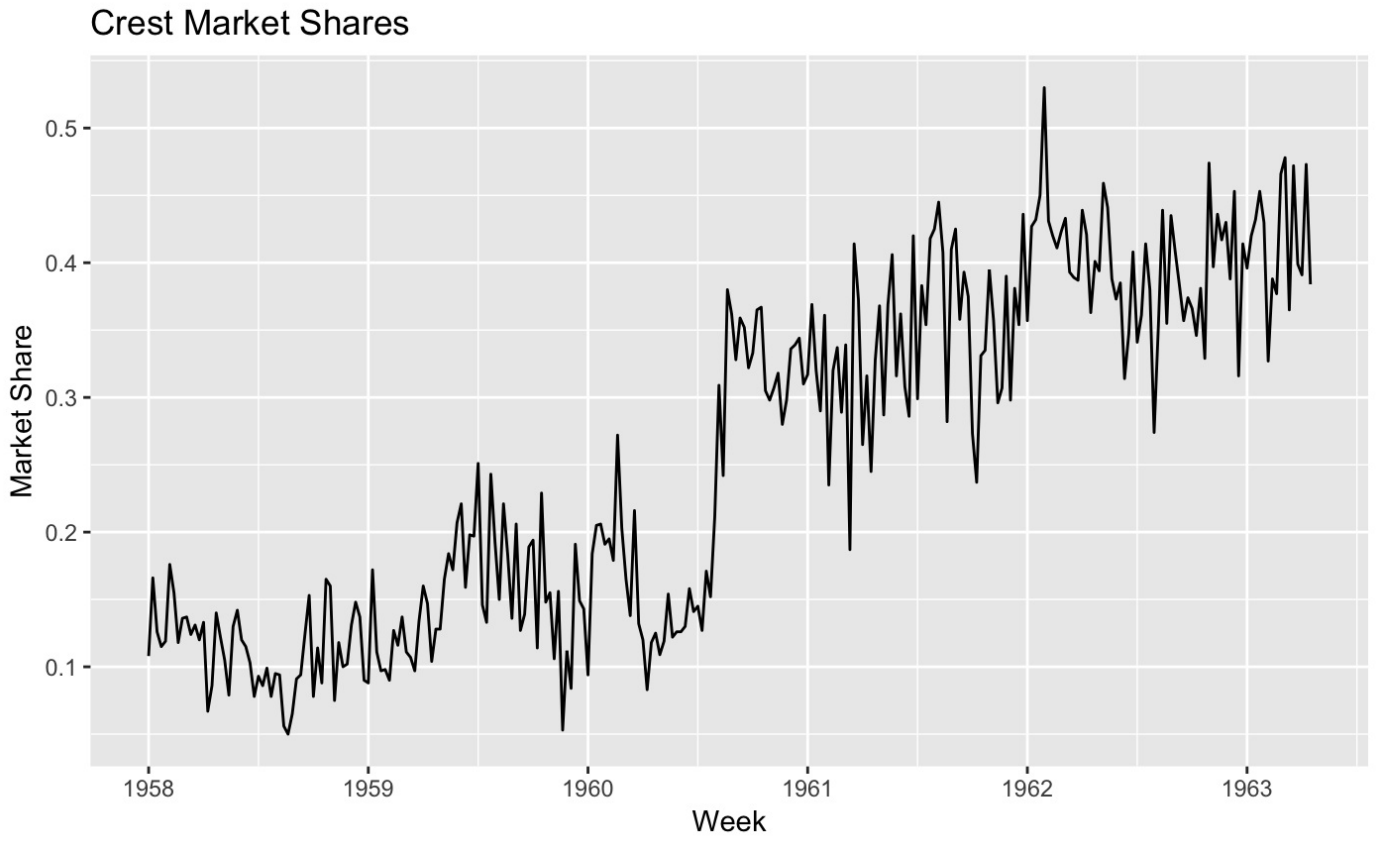
Hide

```
autoplot(colgate) +
  ggtitle("Colgate Market Shares") +
  xlab("Week") +
  ylab("Market Share")
```



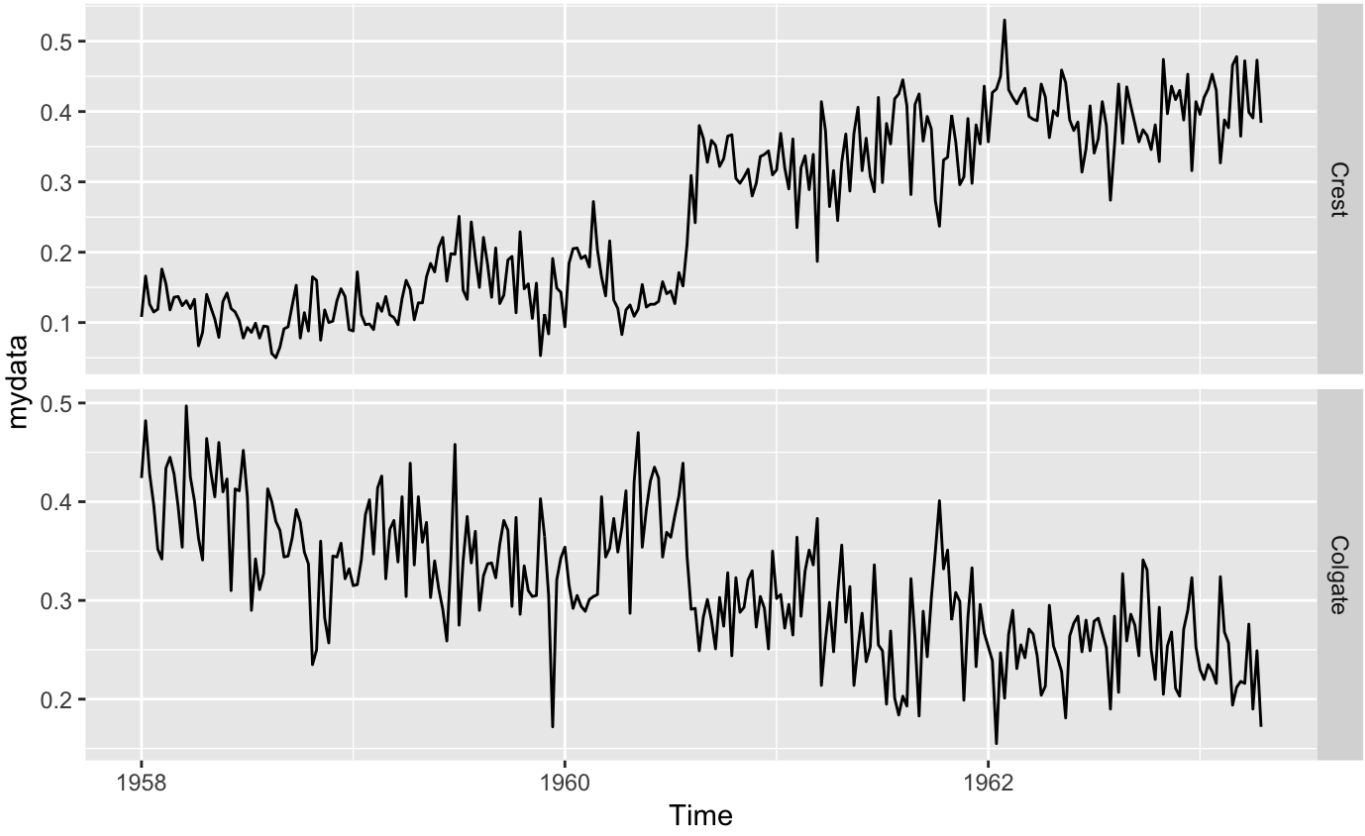
Hide

```
autoplot(crest) +
  ggtitle("Crest Market Shares") +
  xlab("Week") +
  ylab("Market Share")
```



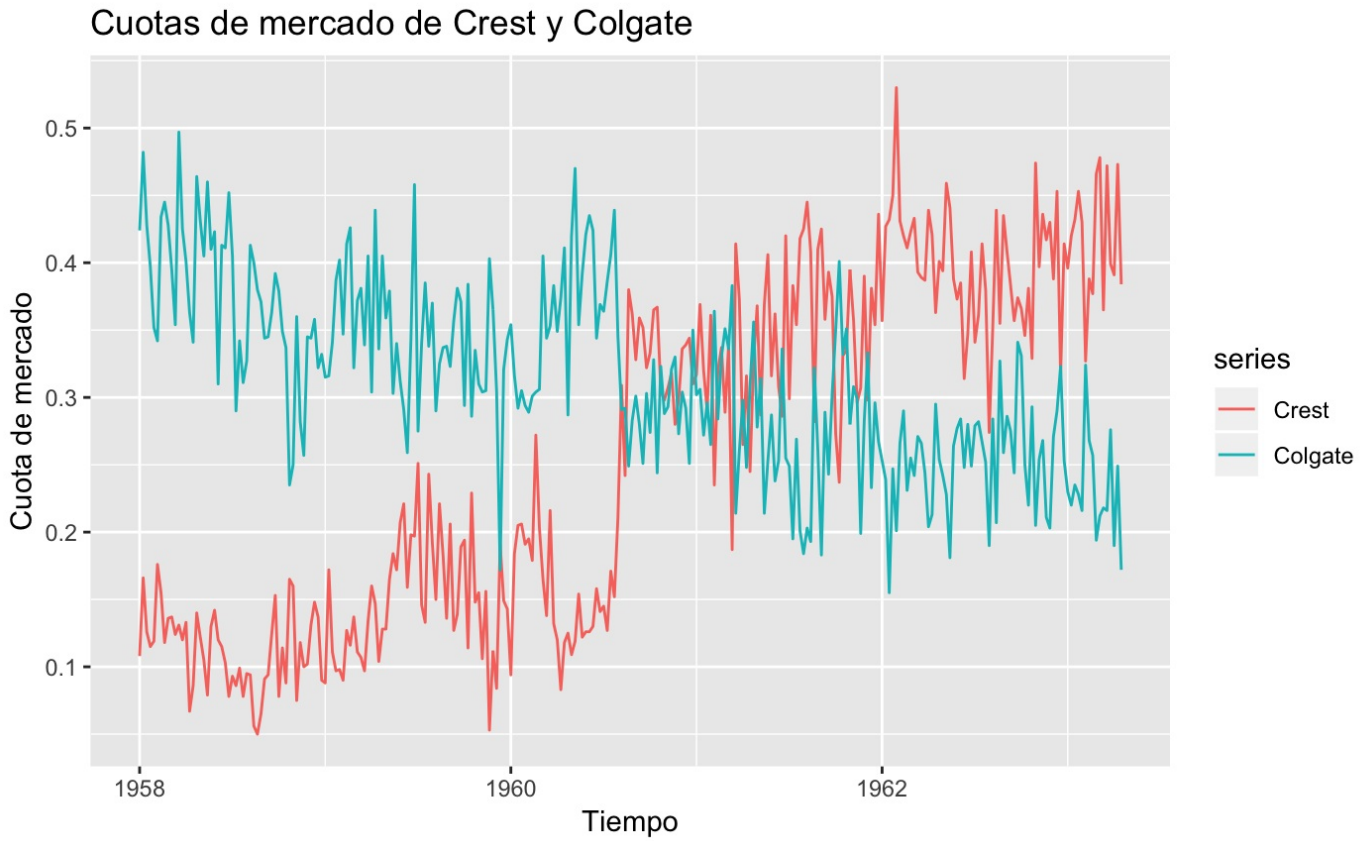
Hide

```
autoplot(mydata, facets = TRUE)
```



Hide

```
autoplot(mydata, facets = FALSE) +
  ggtitle("Cuotas de mercado de Crest y Colgate") +
  xlab("Tiempo") +
  ylab("Cuota de mercado")
```



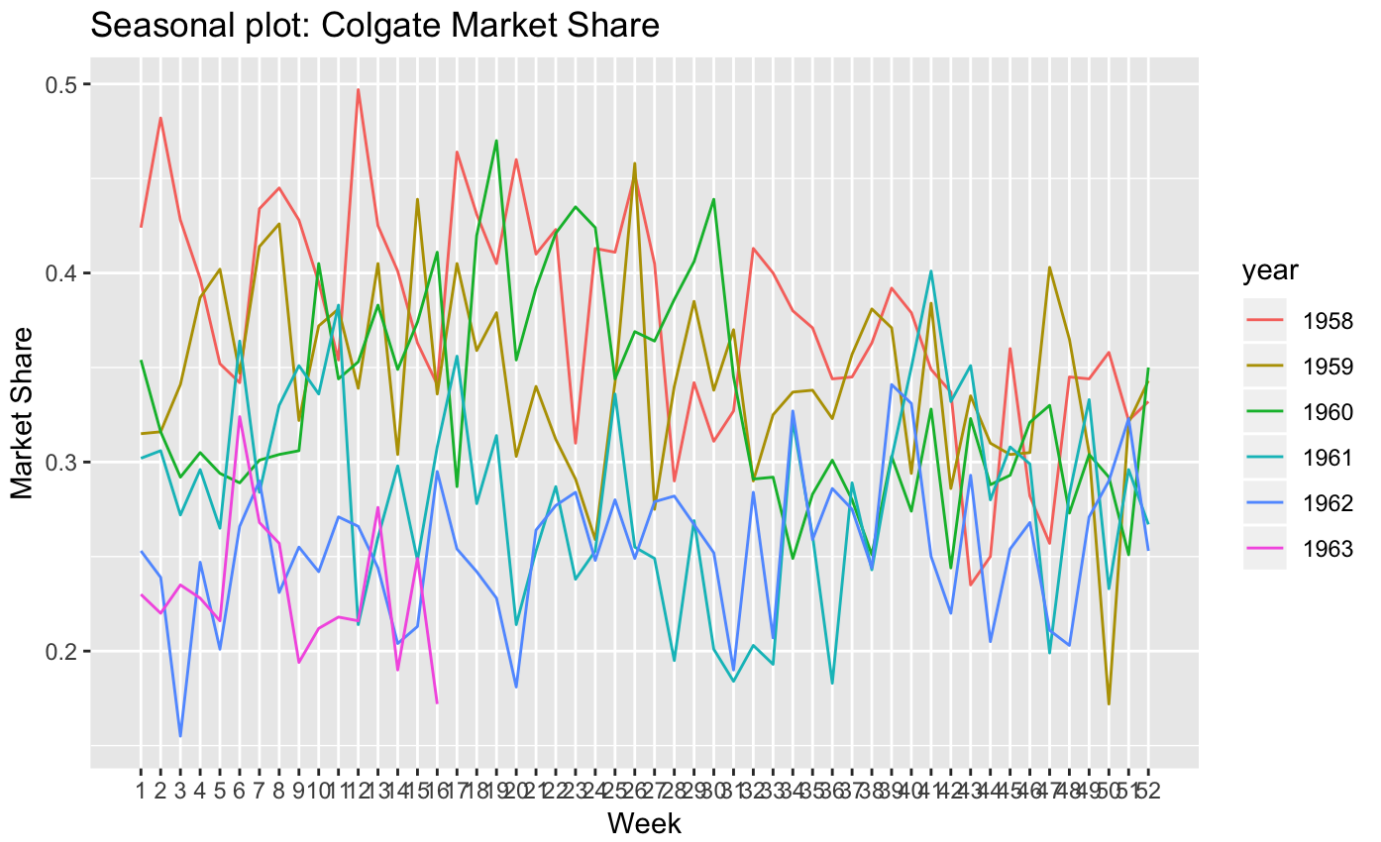
Interpretation:

The Colgate's plot shows a clear decreasing pattern, market share starts at 0.425 and falls to 0.172 in the end. The Crest's plot shows a clear increasing pattern, market share starts at 0.108 and finishes up to 0.384 reaching in the middle even higher values.

In both graphics we can see seasonabilty.

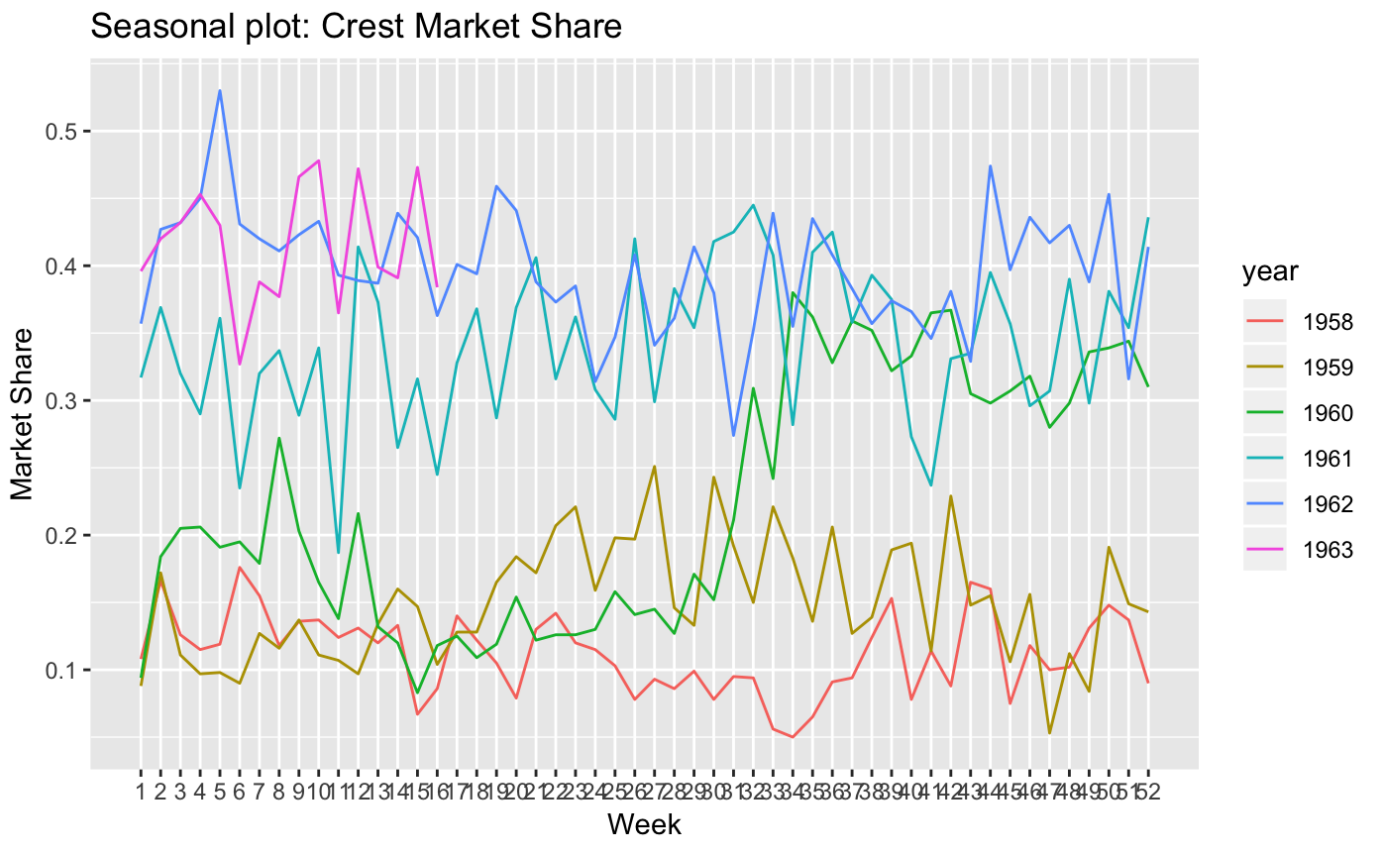
Hide

```
ggseasonplot(colgate, month.labels = TRUE, month.labels.left = TRUE) +
  ylab("Market Share") +
  ggtitle("Seasonal plot: Colgate Market Share")
```



Hide

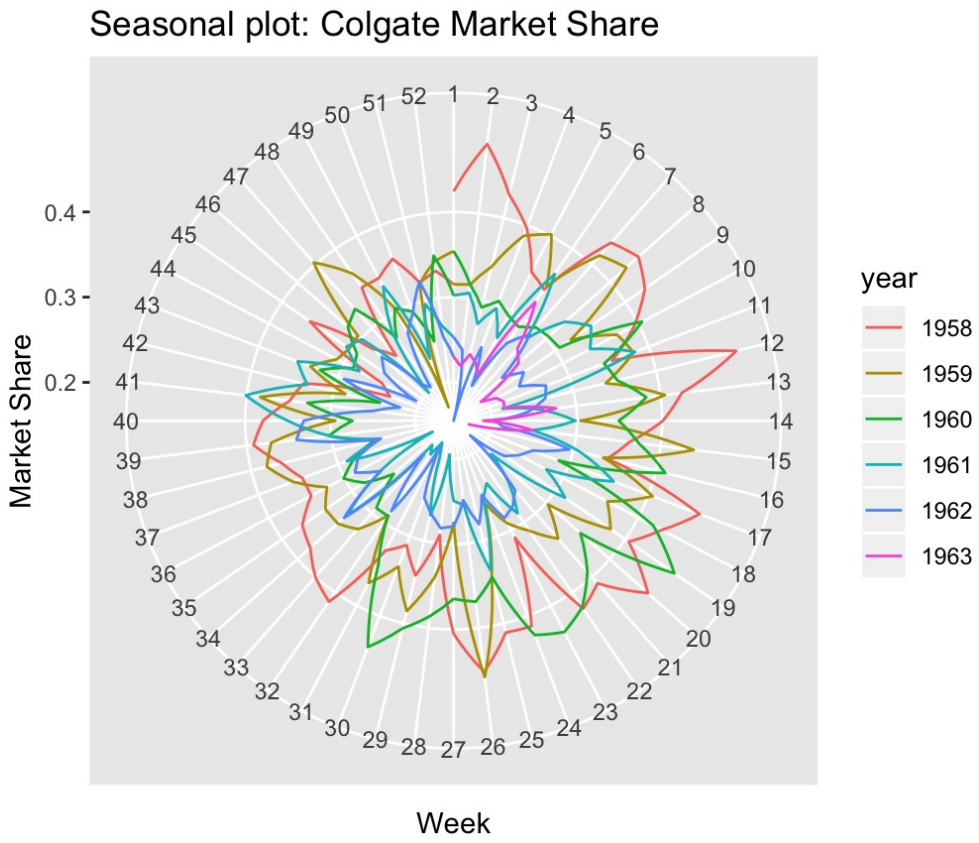
```
ggseasonplot(crest, week.labels = TRUE, week.labels.left = TRUE) +
  ylab("Market Share") +
  ggtitle("Seasonal plot: Crest Market Share")
```



We can see a strong relationship between both market shares. The biggest descents in Colgate's shares (Weeks 29 to 32 in 1960 and 10 to 12 in 1961) coincide with Crest's biggest ascents.

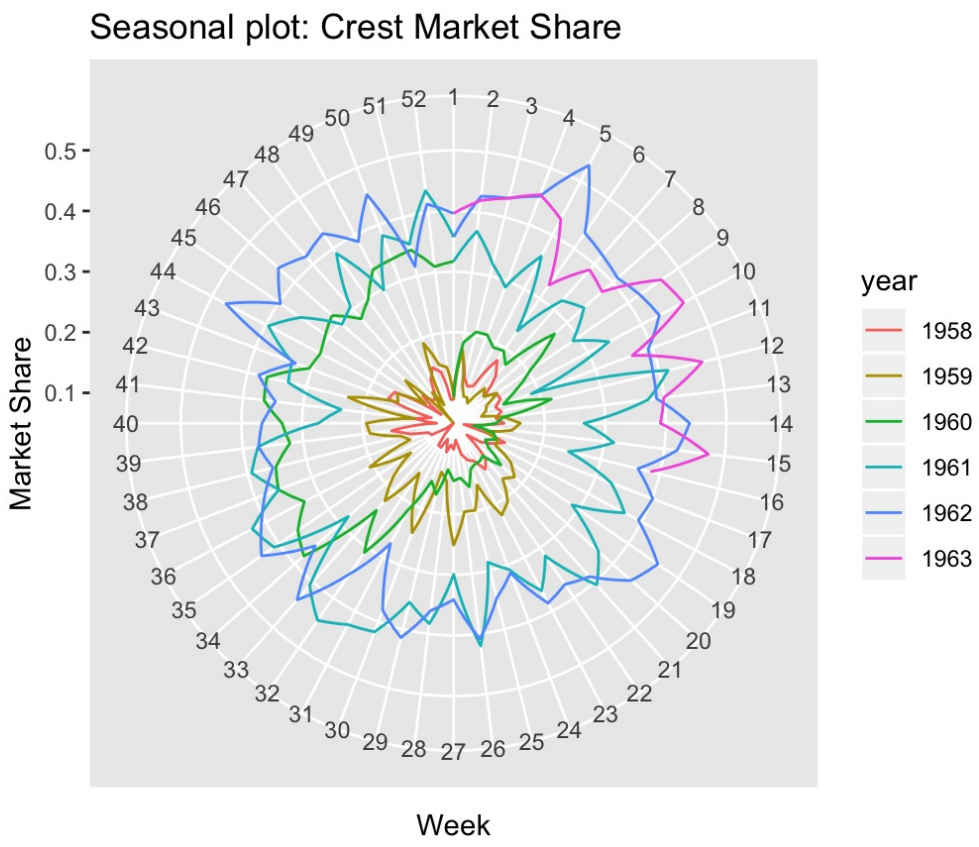
Hide

```
ggseasonplot(colgate, polar = TRUE) +
  ylab("Market Share") +
  ggtitle("Seasonal plot: Colgate Market Share")
```



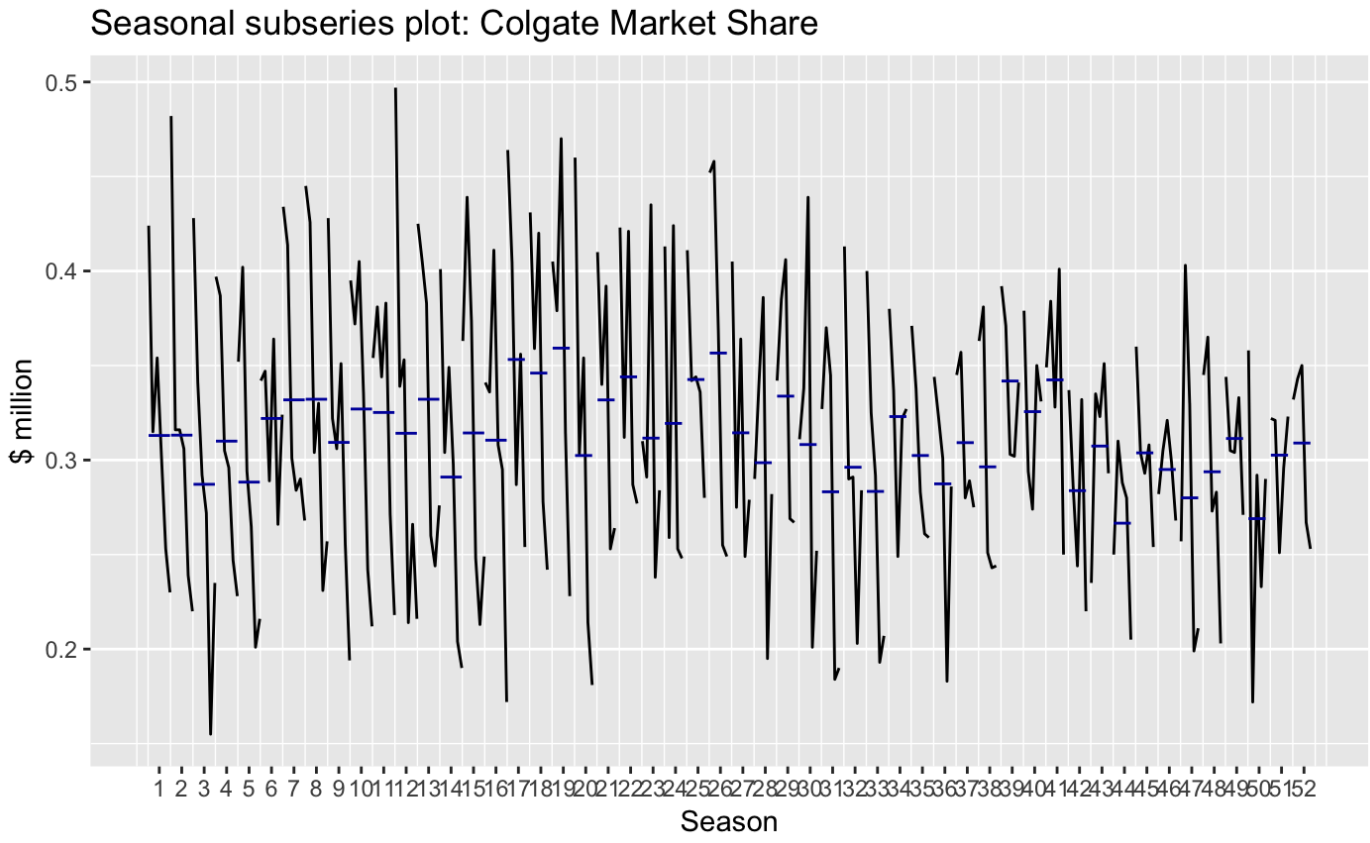
Hide

```
ggseasonplot(crest, polar = TRUE) +
  ylab("Market Share") +
  ggtitle("Seasonal plot: Crest Market Share")
```



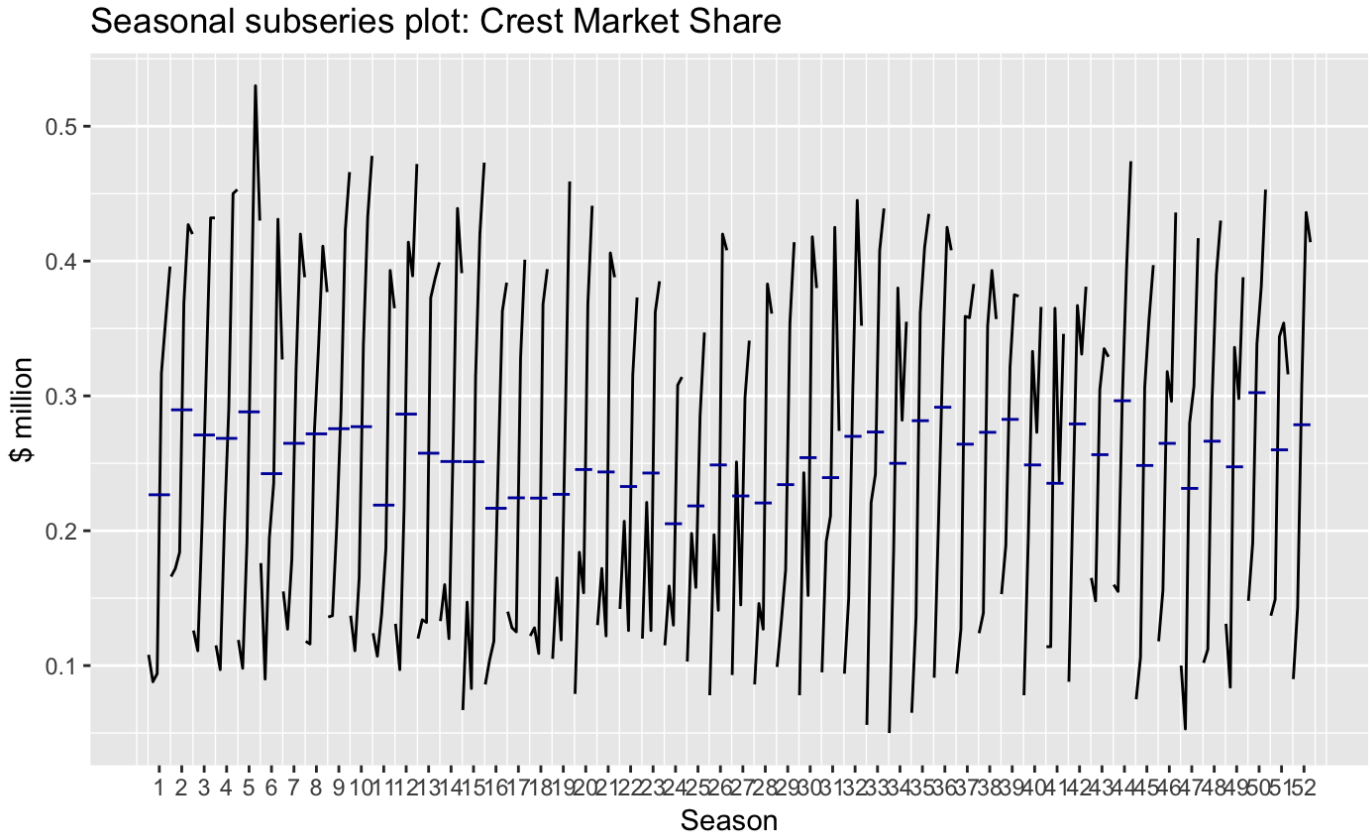
Hide

```
ggsubseriesplot(colgate) +
  ylab("$ million") +
  ggtitle("Seasonal subseries plot: Colgate Market Share")
```



Hide

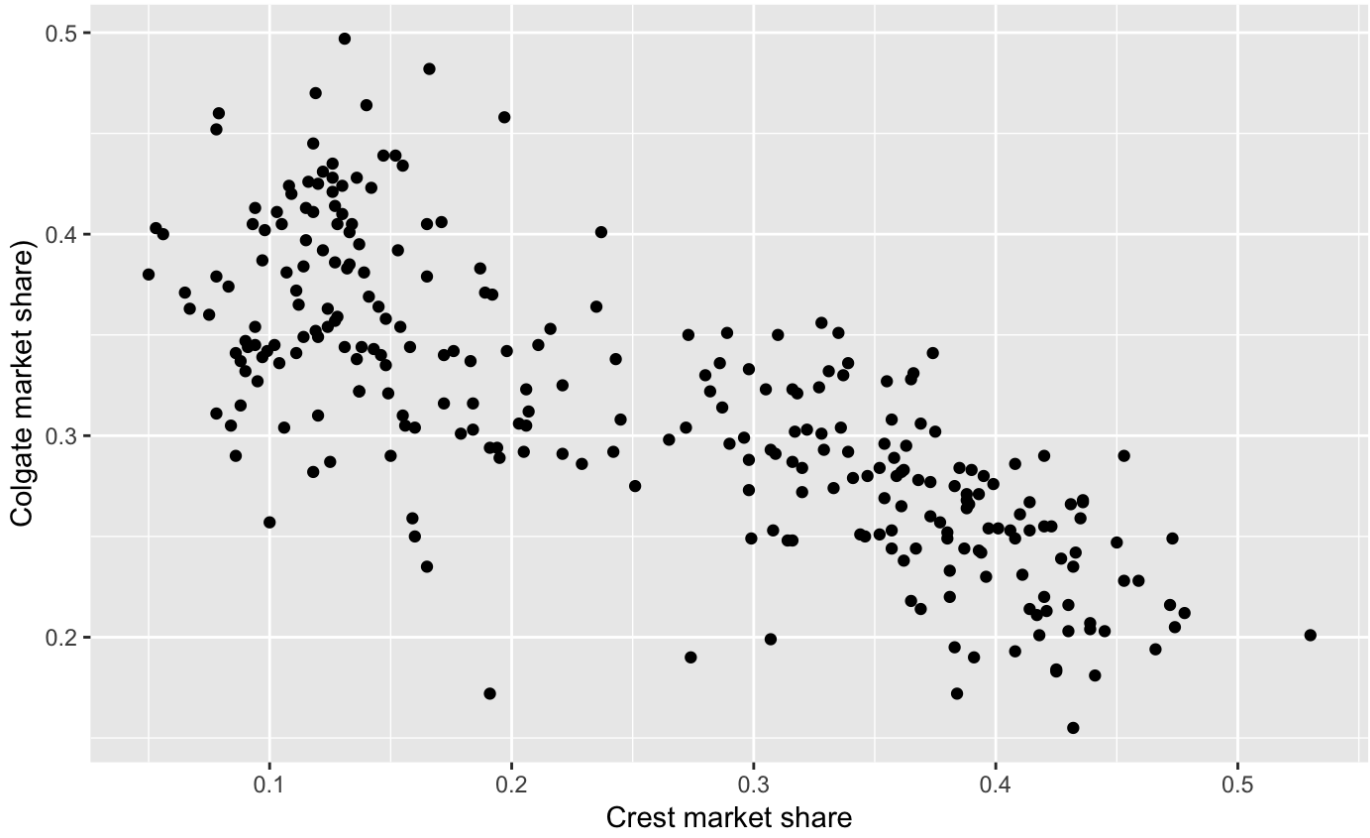
```
ggsubseriesplot(crest) +
  ylab("$ million") +
  ggtitle("Seasonal subseries plot: Crest Market Share")
```



It is possible to compare the two time series in order to evaluate the relationship between them:

Hide

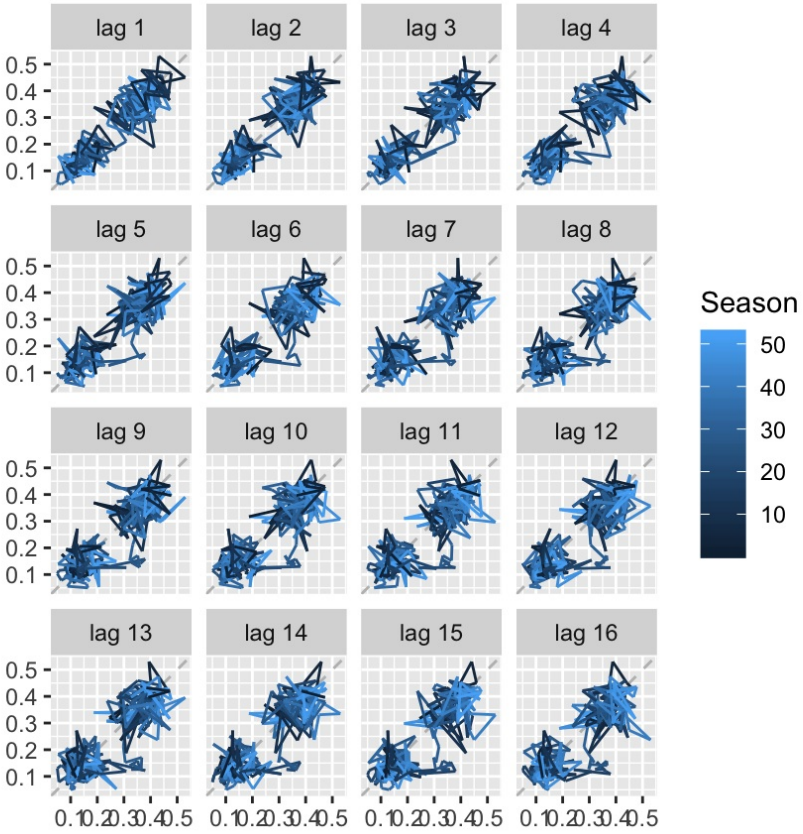
```
qplot(crest, colgate) +
  ylab("Colgate market share)") + xlab("Crest market share")
```



As we have seen before as Crest grows, Colgate decreases. Since they have the same target client this is quite a reasonable statement.

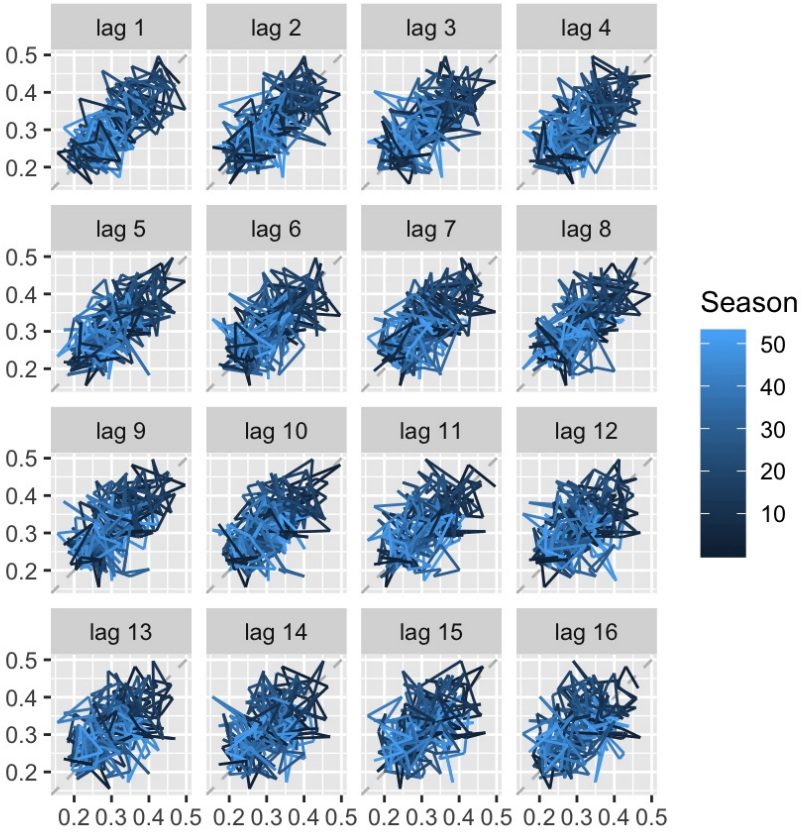
Hide

```
gglagplot(crest)
```



Hide

```
gglagplot(colgate)
```

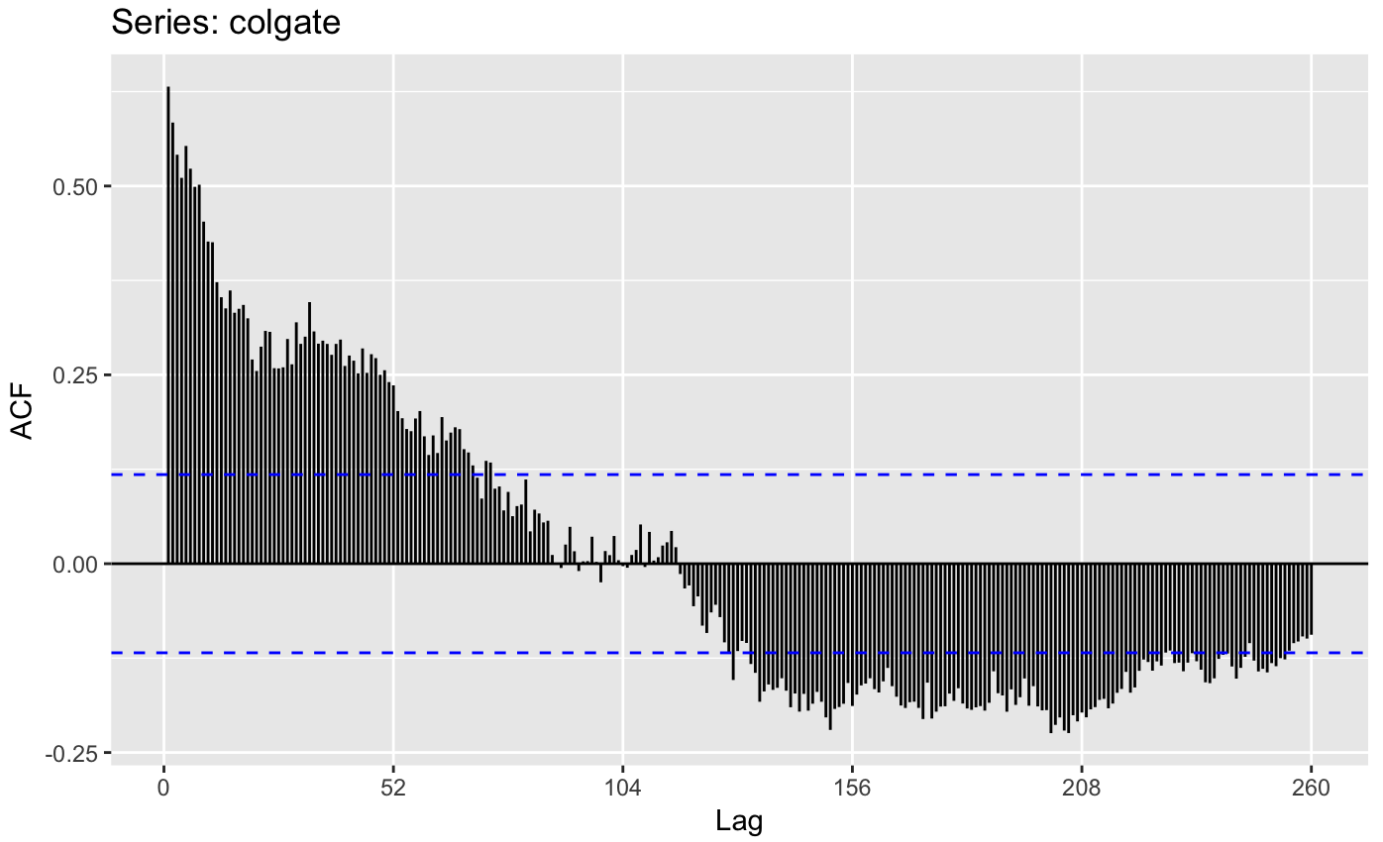


The relationship is strongly positive at crest lags, reflecting the strong seasonality in the data. For Colgate lags the plot is not easy to read.

Hide

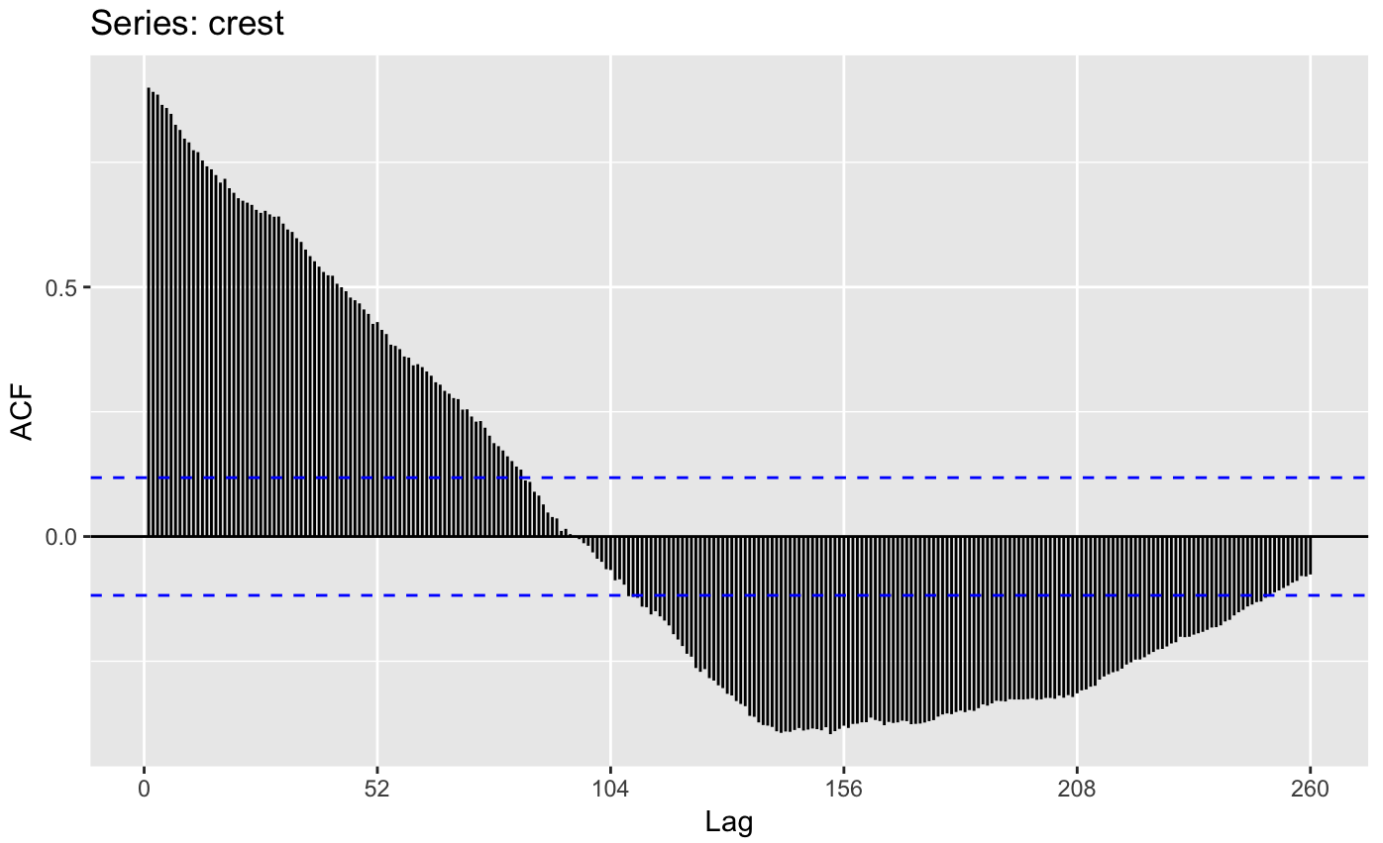
Hide

```
ggAcf(colgate, lag.max = 260)
```



Hide

```
ggAcf(crest, lag.max = 260)
```



When data have a trend, the autocorrelations for small lags tend to be large and positive because observations nearby in time are also nearby in size. So the ACF of trended time series tend to have positive values that slowly decrease as the lags increase.

When data are seasonal, the autocorrelations will be larger for the seasonal lags (at multiples of the seasonal frequency) than for other lags.

When data are both trended and seasonal, you see a combination of these effects. This is the case when working with Colgate and Crest.

First (easy) predictions

Now we are going to split the observations in a training set and a test set. The training set will be composed by the observations in the years 1958-1962 (both included) and we will use as a test set the sixteen weeks from 1963.

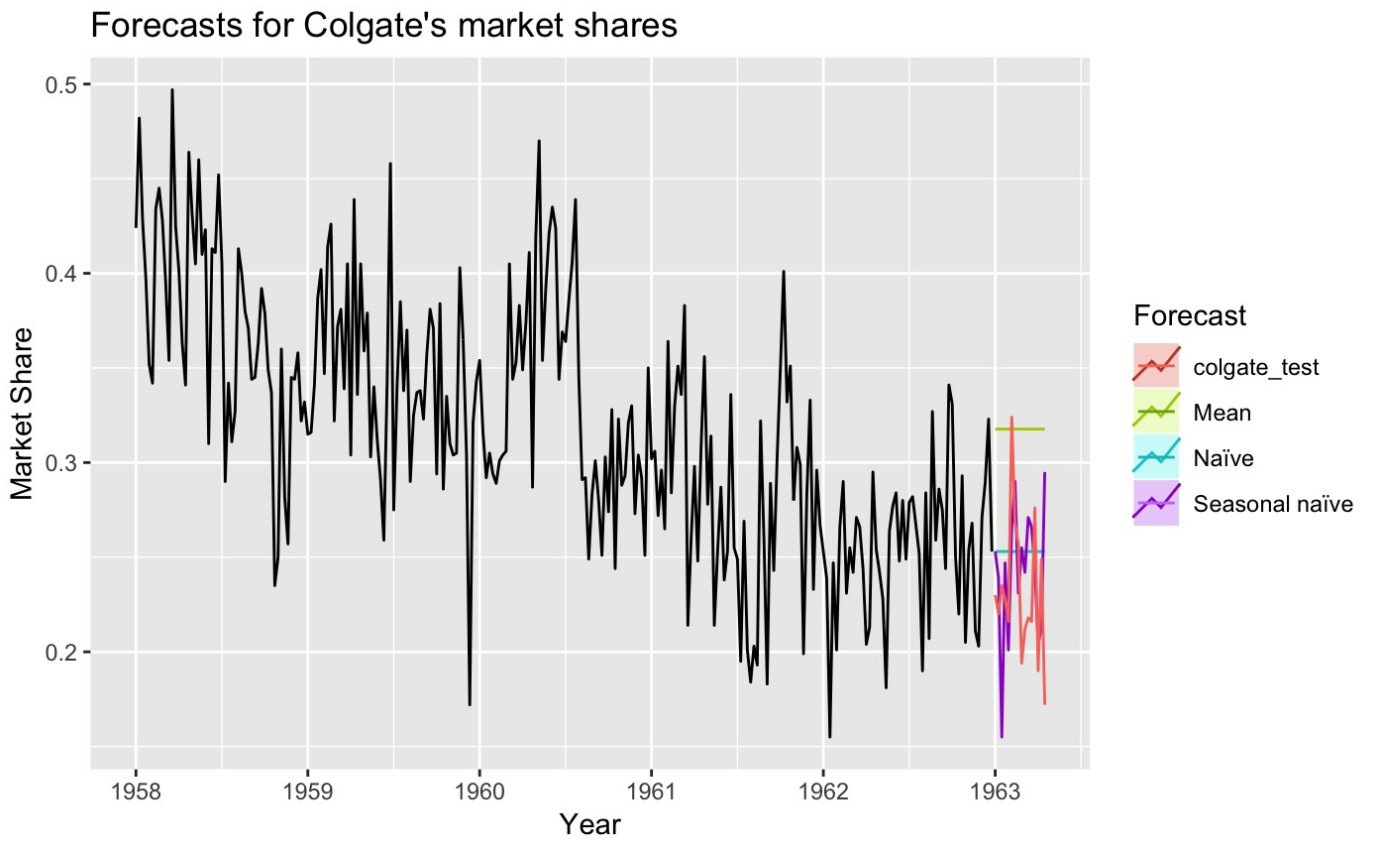
Hide

```
set.seed(12345)
colgate_training <-  window(colgate, start = 1958,end = c(1962, 52))
crest_training <- window(crest, start = 1958, c(1962, 52))
colgate_test <- window(colgate, start = 1963)
crest_test <- window(crest, start = 1963)
training_data <- data[1:260,]
```

Once this is done we will start by setting some very obvious methods as a basis on which we can compare the models we will be building later. Sometimes one of these simple methods will be the best forecasting method available; but in many cases, these methods will serve as benchmarks rather than the method of choice. That is, any forecasting methods we develop will be compared to these simple methods to ensure that the new method is better than these simple alternatives. If not, the new method is not worth considering.

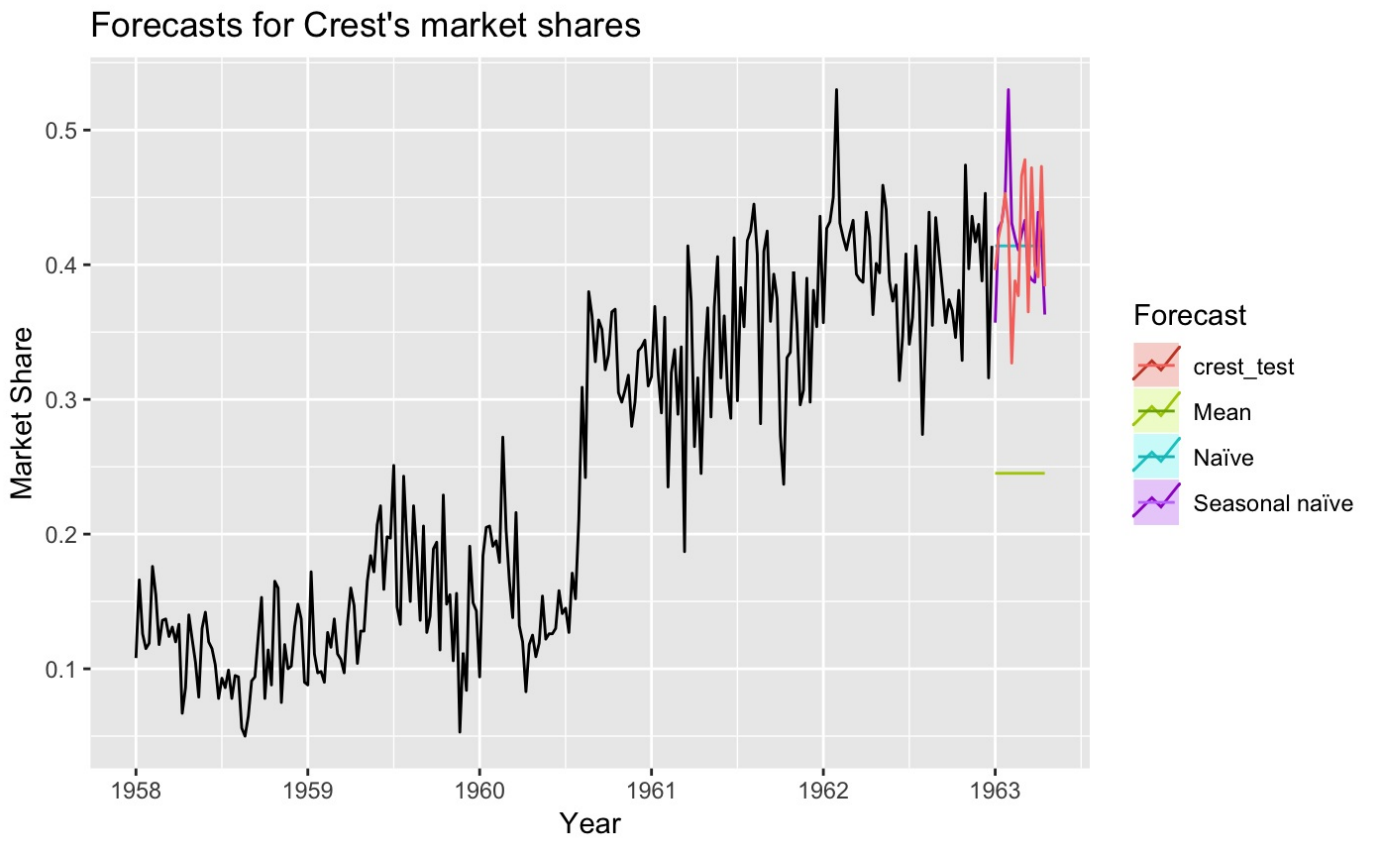
Hide

```
autoplot(colgate_training) +
  autolayer(meanf(colgate_training, h = 16),
    series = "Mean", PI = FALSE) +
  autolayer(naive(colgate_training, h = 16),
    series = "Naïve", PI = FALSE) +
  autolayer(snaive(colgate_training, h = 16),
    series = "Seasonal naïve", PI = FALSE) +
  autolayer(colgate_test) +
  ggtitle("Forecasts for Colgate's market shares") +
  xlab("Year") + ylab("Market Share") +
  guides(colour = guide_legend(title = "Forecast"))
```



Hide

```
autoplot(crest_training) +
  autolayer(meanf(crest_training, h = 16),
    series = "Mean", PI = FALSE) +
  autolayer(naive(crest_training, h = 16),
    series = "Naïve", PI = FALSE) +
  autolayer(snaive(crest_training, h = 16),
    series = "Seasonal naïve", PI = FALSE) +
  autolayer(crest_test) +
  ggtitle("Forecasts for Crest's market shares") +
  xlab("Year") + ylab("Market Share") +
  guides(colour = guide_legend(title = "Forecast"))
```

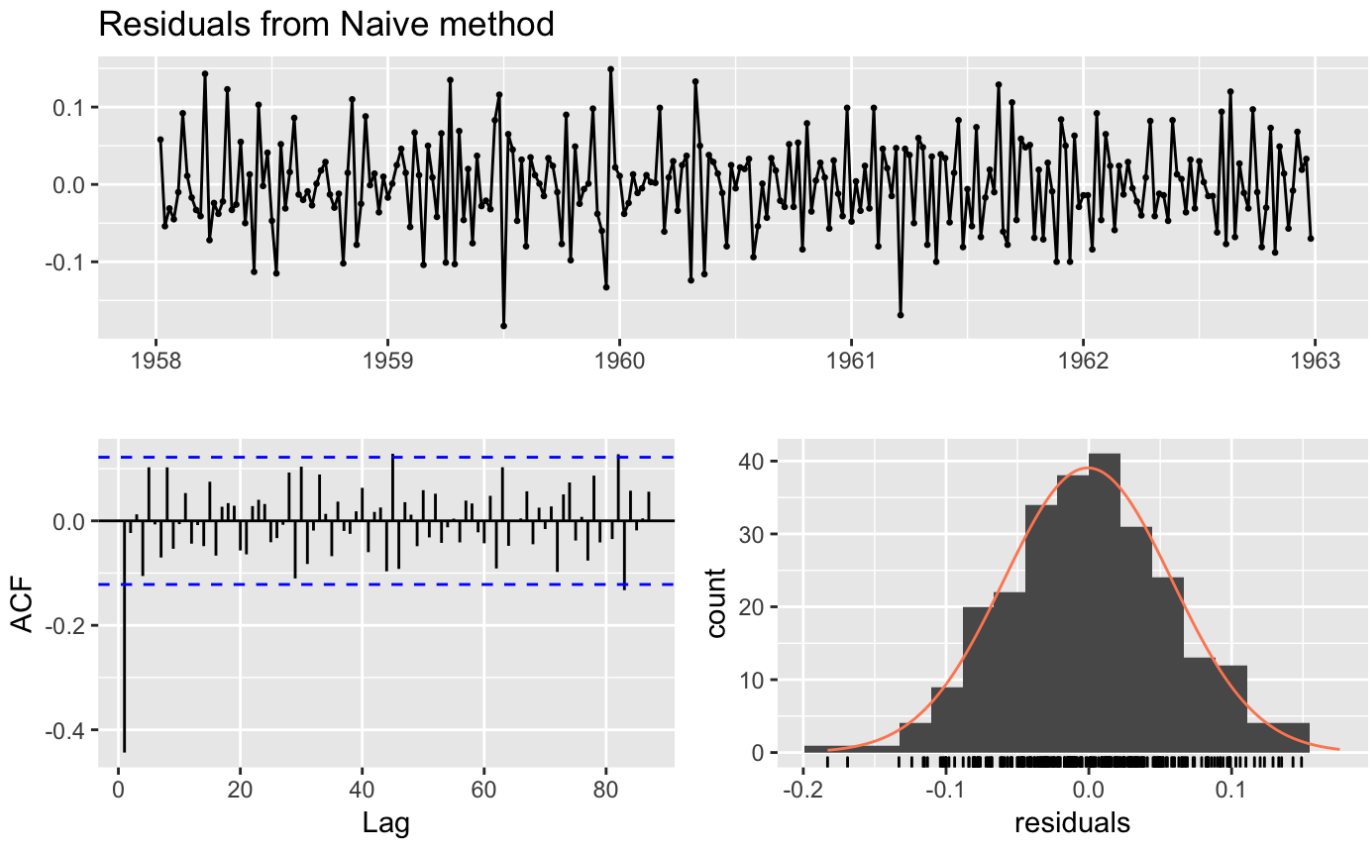


As we can see for both options the seasonal naïve seems like the best option and yet we it is not a great one. This justifies searching for more complex models. But first let's explore the residuals from one of these predictions, for example the naïve one:

Hide

```
checkresiduals(naive(colgate_training))
```

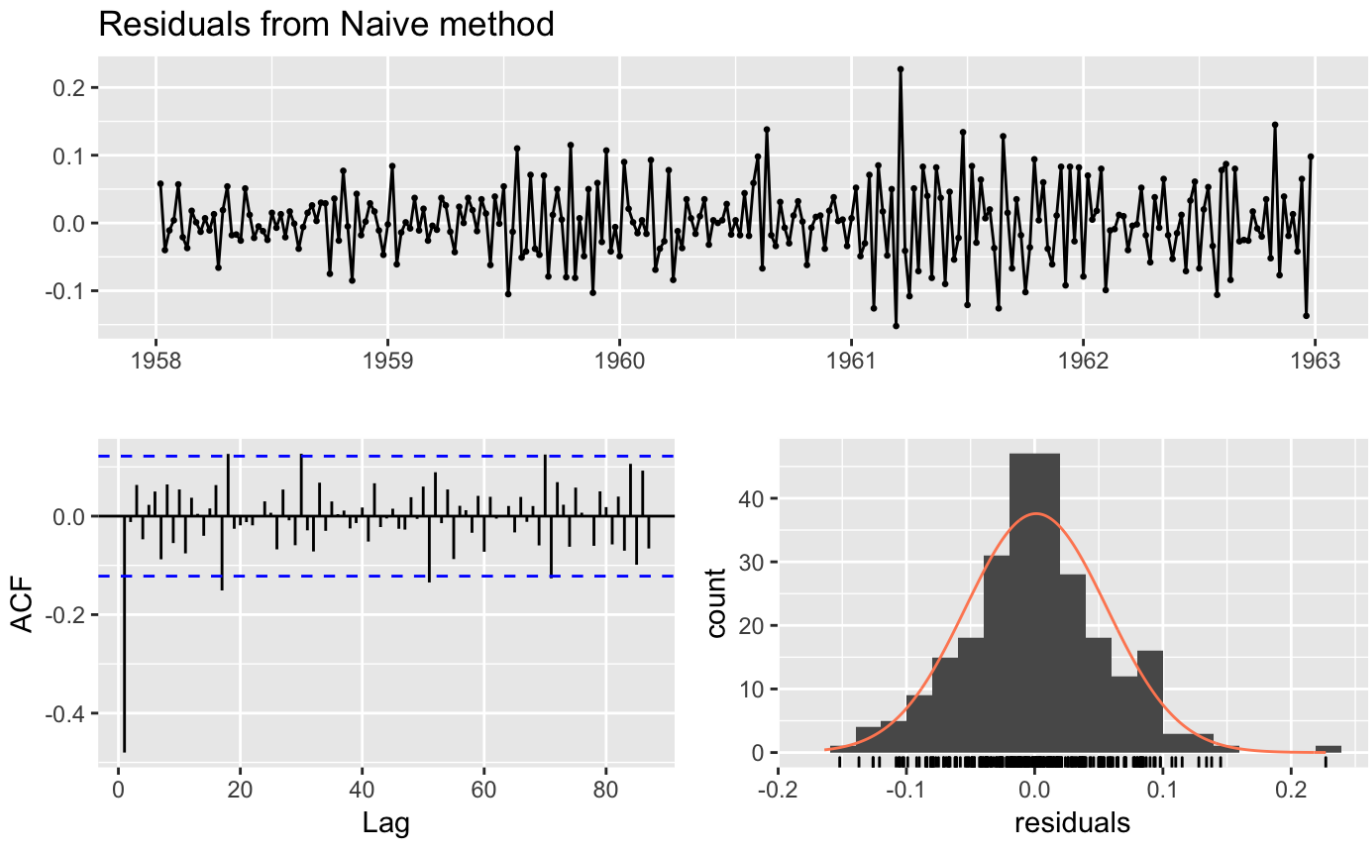
```
    Ljung-Box test

data:  Residuals from Naive method
Q* = 104.79, df = 52, p-value = 2.02e-05

Model df: 0.   Total lags used: 52
```



Hide

```
checkresiduals(naive(crest_training))
```

```

    Ljung-Box test

data:  Residuals from Naive method
Q* = 107.73, df = 52, p-value = 9.07e-06

Model df: 0.   Total lags used: 52
```



There are so many reasons why this forecast is not good. First, we can see from the ACF plot that lots of spikes cross the blue line in both forecasts. Second both forecasts as well have a large stadistic associated and a really tiny p-value. This makes us think that the distribution from the residuals is far from white noise.

As a didachtical exercises we are also going to calculate the accuracy:

Hide

```
accuracy(naive(colgate_training), colgate_test)
```

```
                      ME       RMSE        MAE       MPE     MAPE      MASE       ACF1 Theil's U
Training set -0.0006602317 0.05866820 0.04666795 -2.157966 15.58708 0.7466873 -0.4435503        NA
Test set     -0.0146000000 0.03792097 0.03260000 -8.143909 13.95731 0.5216000  0.1631756 0.8698155
```

Hide

```
accuracy(naive(crest_training), crest_test)
```

```
                     ME       RMSE        MAE        MPE      MAPE      MASE       ACF1 Theil's U
Training set 0.001181467 0.05495663 0.04191506 -3.1562901 20.429905 0.4623393 -0.4799130        NA
Test set     0.002700000 0.04344537 0.03630000 -0.5192954  9.013967 0.4004030  0.2449732 0.8158517
```

Hide

```
accuracy(snaive(colgate_training), colgate_test)
```

```
                     ME       RMSE       MAE        MPE     MAPE  MASE       ACF1 Theil's U
Training set -0.03017308 0.07922048 0.0625000 -13.369109 22.59943 1.000  0.3767356        NA
Test set     -0.01043750 0.04998437 0.0413125  -7.003501 18.88563 0.661 -0.1749880  1.005611
```

Hide

```
accuracy(snaive(crest_training), crest_test)
```

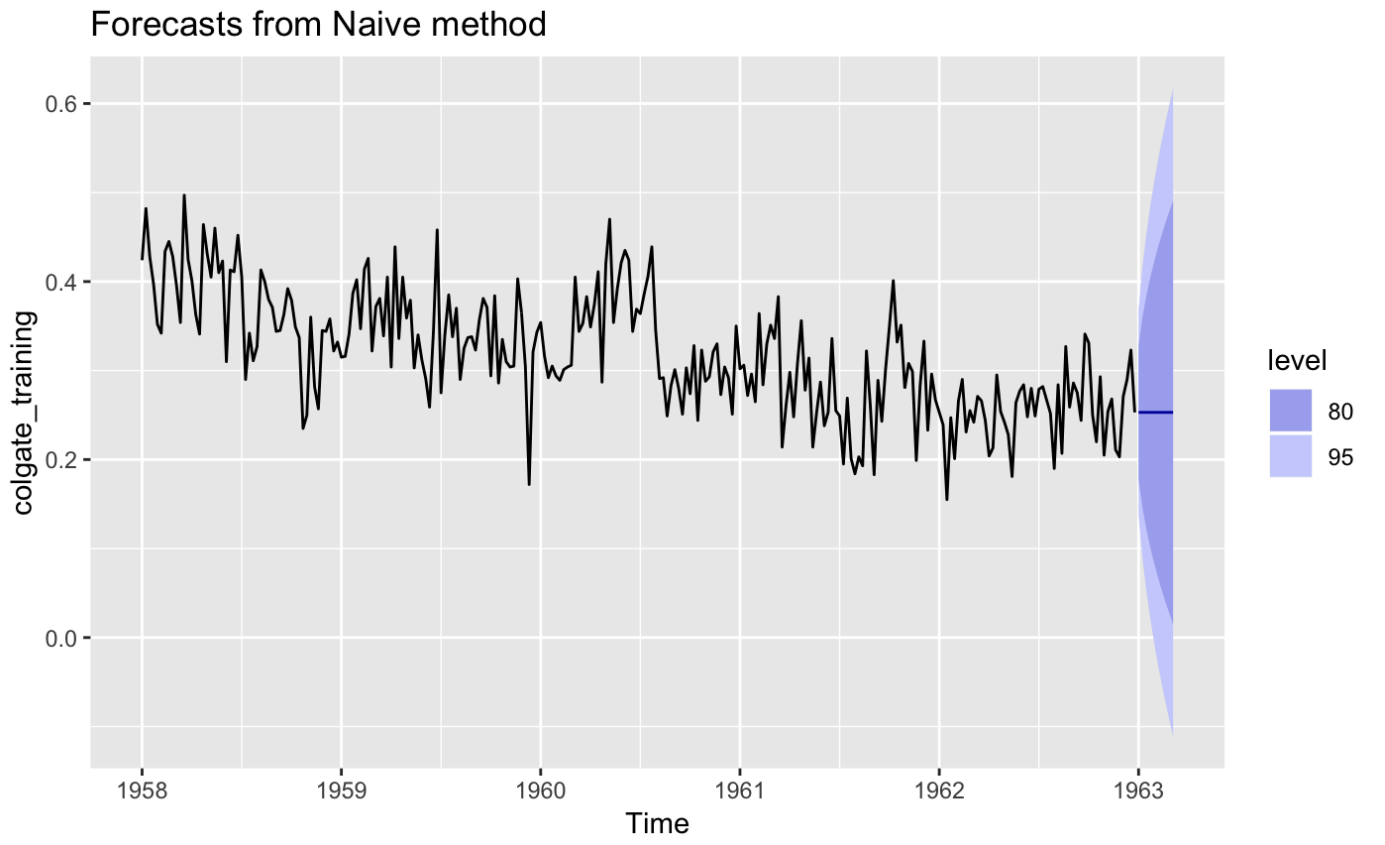
```
                     ME       RMSE        MAE       MPE     MAPE      MASE      ACF1 Theil's U
Training set  0.07115865 0.11536854 0.09065865 20.294564 33.10118 1.0000000 0.6463840        NA
Test set     -0.00343750 0.05106063 0.04068750 -1.733319 10.00921 0.4487989 0.2345732 0.7657561
```

The best method would be the naïve approximation since it has the lowest MASE (and we are going to prior this indicator).

Some times it is useful to have an interval in which we are surer that the prediction is contained. In the following plots we show the prediction and the intervalos of 80% and 95% which means that the probability of the estimate of being in this interval is 0.8 or 0.95:

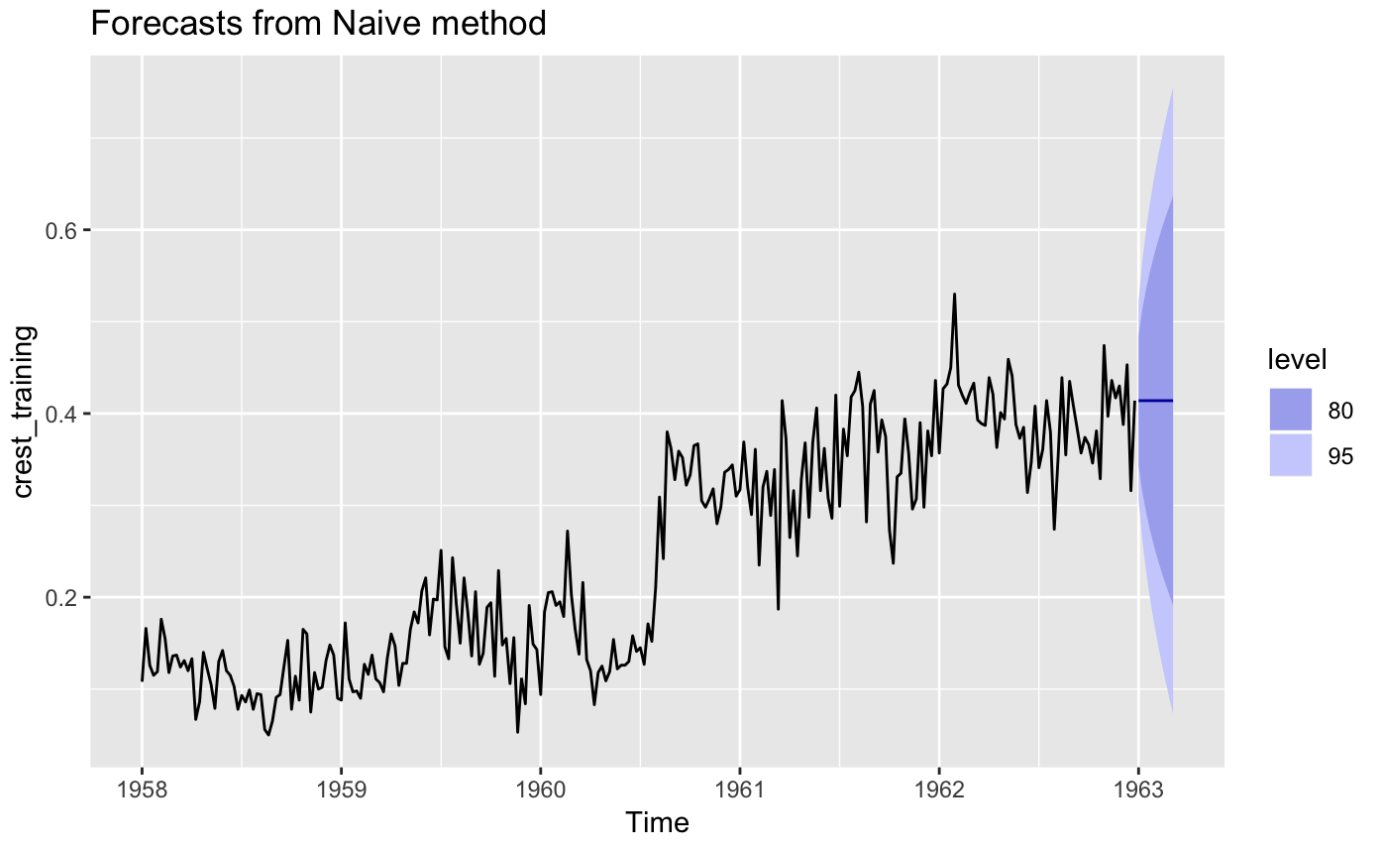
Hide

```
autoplot(naive(colgate_training))
```



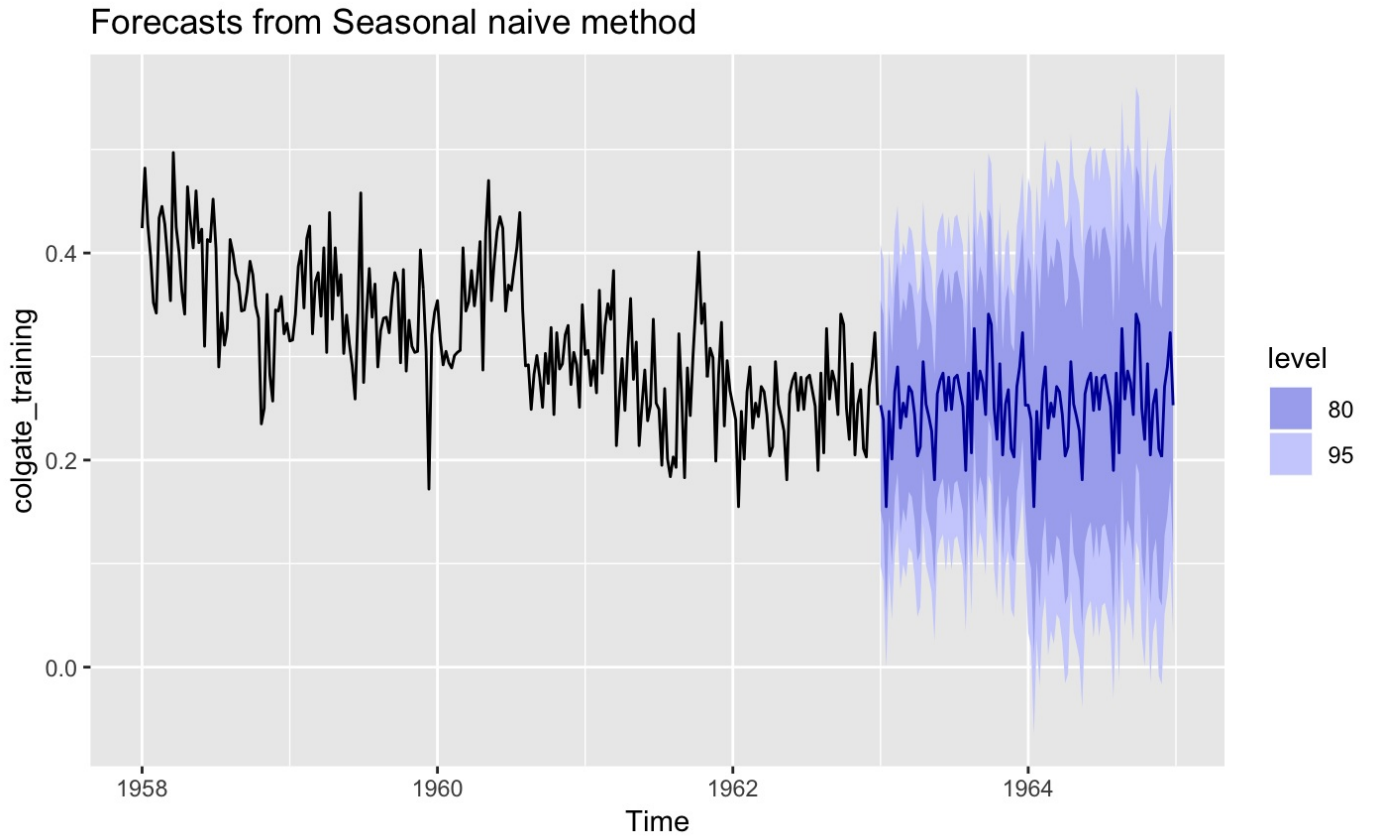
Hide

```
autoplot(naive(crest_training))
```



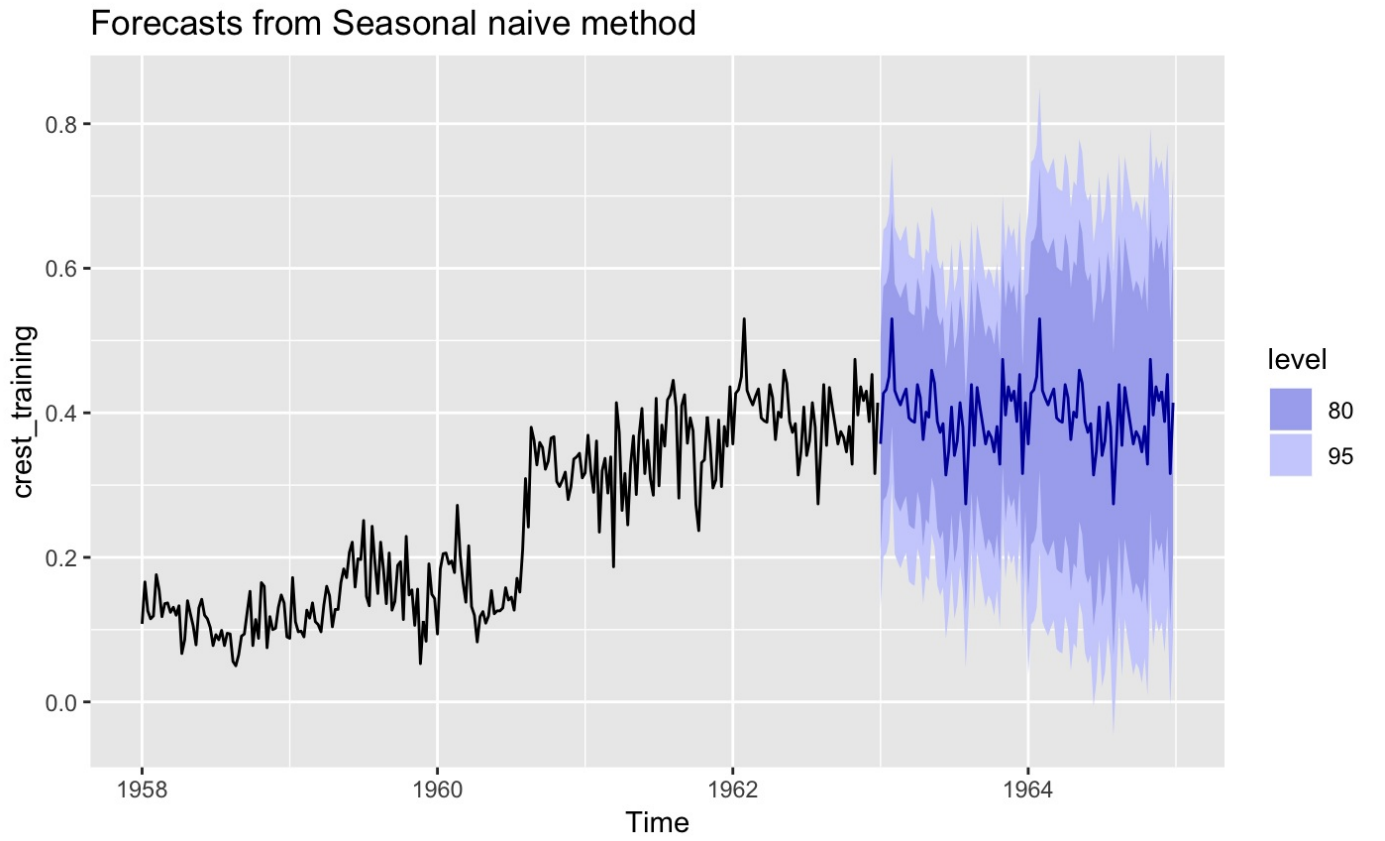
Hide

```
autoplot(snaive(colgate_training))
```



Hide

```
autoplot(snaive(crest_training))
```



When a normal distribution for the forecast errors is an unreasonable assumption, one alternative is to use bootstrapping, which only assumes that the forecast errors are uncorrelated.

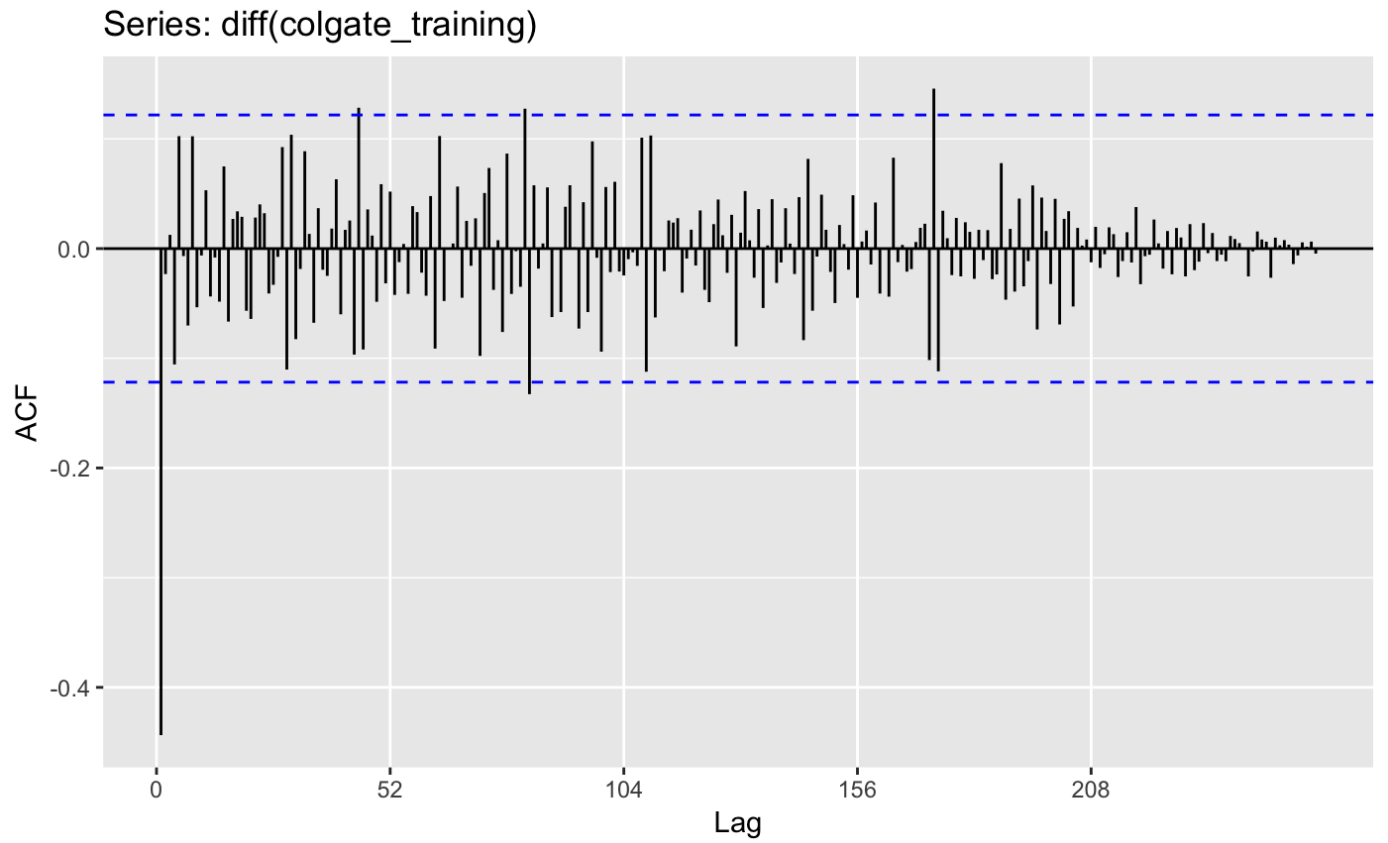
Hide

```
naive(colgate_training, bootstrap = TRUE)
```

Next step is building an ARIMA model: In order to do that we try to convert the time series in stationary series. First we differentiate:

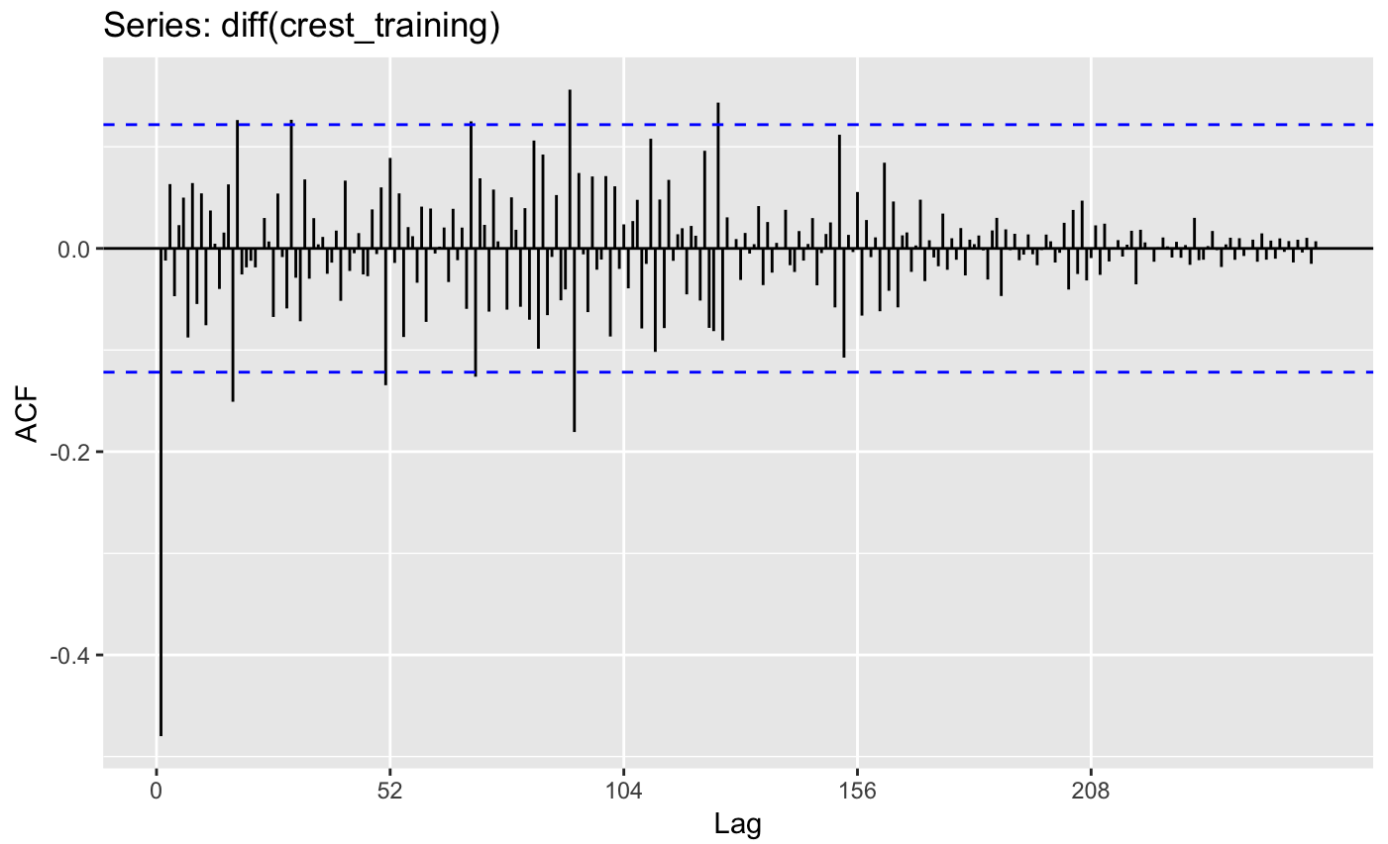
Hide

```
ggAcf(diff(colgate_training), lag.max = 260)
```



Hide

```
ggAcf(diff(crest_training), lag.max = 260)
```



Hide

```
Box.test(diff(colgate_training), type = "Ljung-Box")
```

```
    Box-Ljung test

data:  diff(colgate_training)
X-squared = 51.547, df = 1, p-value = 6.989e-13
```

Hide

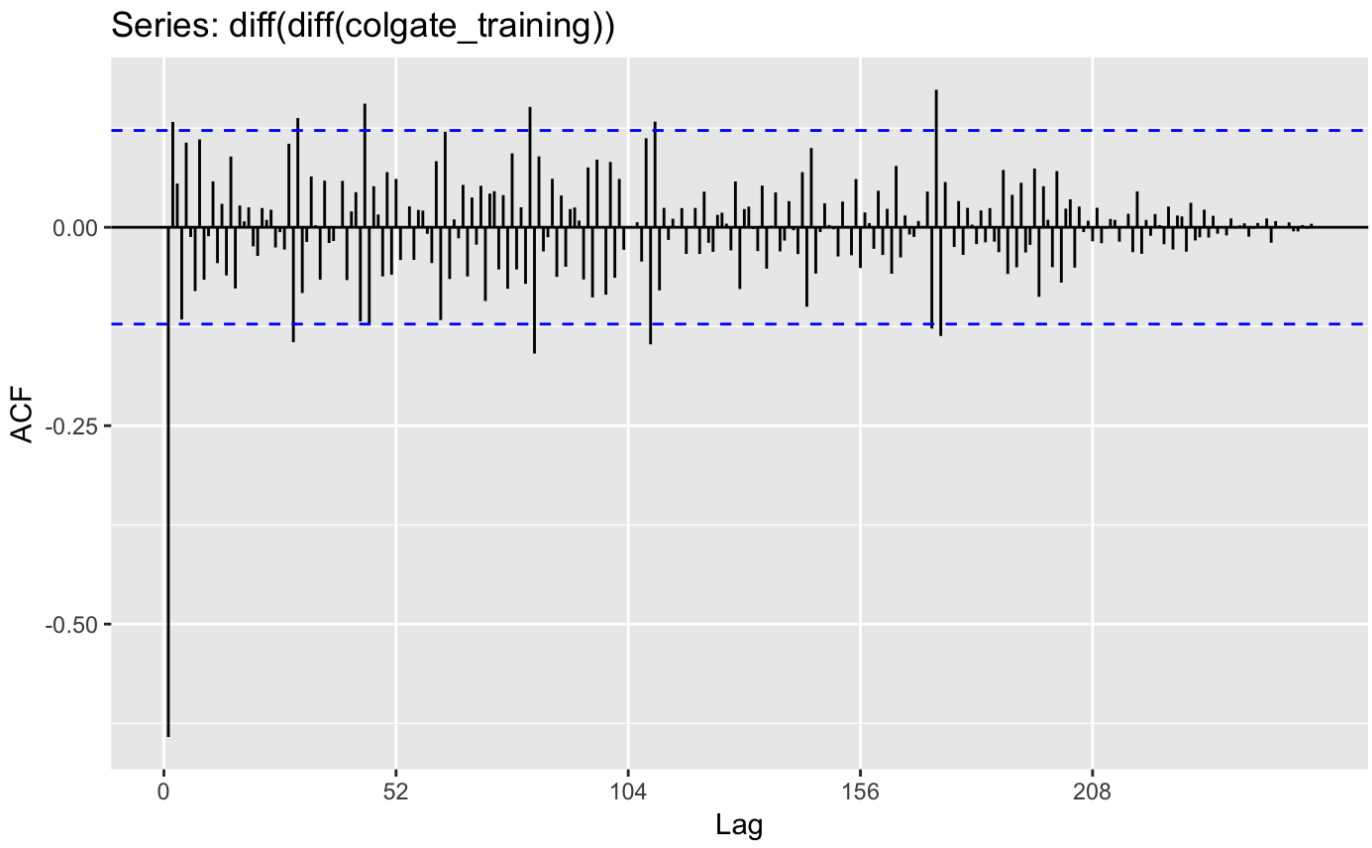
```
Box.test(diff(crest_training), type = "Ljung-Box")
```

```
    Box-Ljung test

data:  diff(crest_training)
X-squared = 60.346, df = 1, p-value = 7.994e-15
```

This appears not to be enough so we are going to differentiate over the differentiated series:

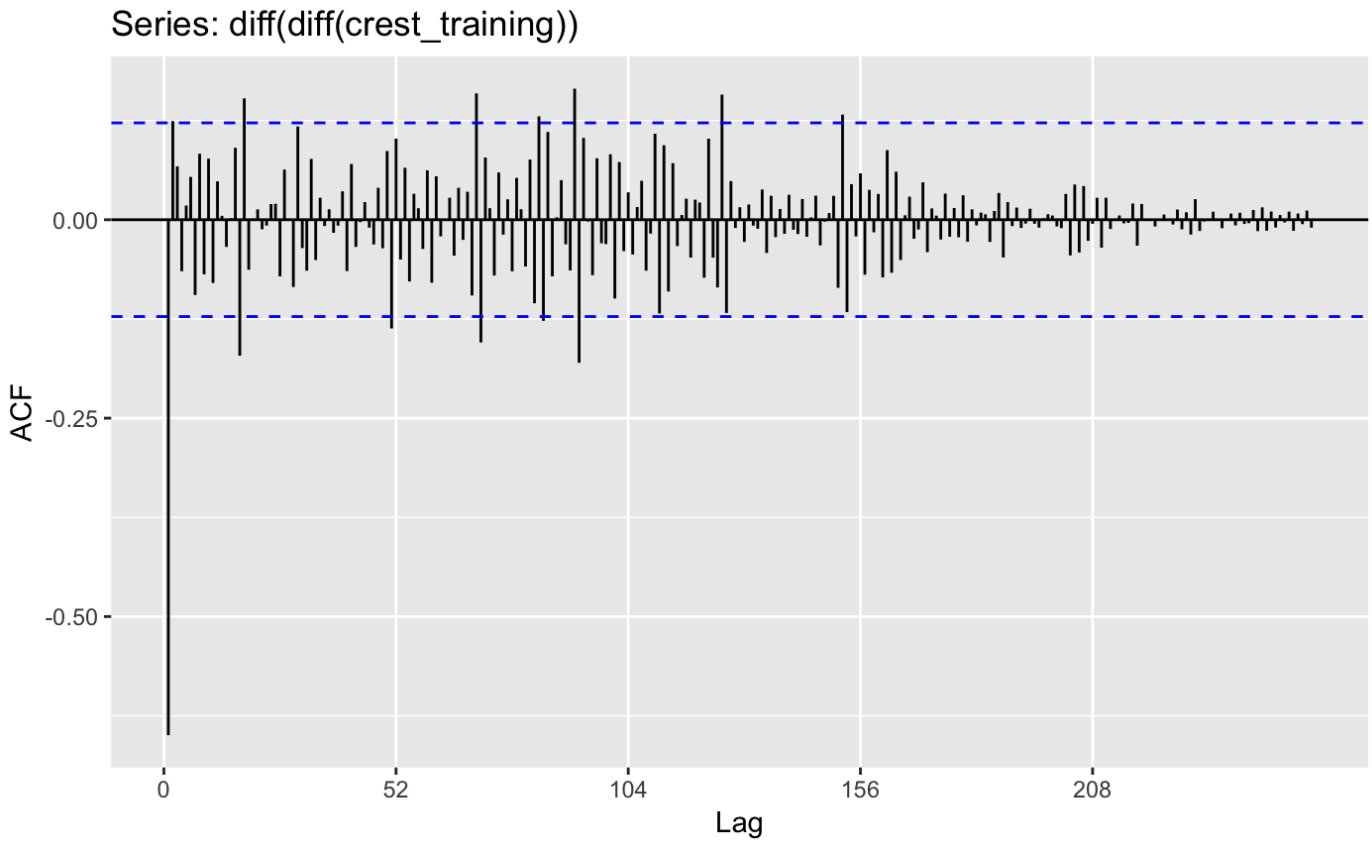
Hide

```
ggAcf(diff(diff(colgate_training)), lag.max = 260)
```



Hide

```
ggAcf(diff(diff(crest_training)), lag.max = 260)
```



Hide

```
Box.test(diff(diff(colgate_training)), type = "Ljung-Box")
```

```
    Box-Ljung test

data:  diff(diff(colgate_training))
X-squared = 107.68, df = 1, p-value < 2.2e-16
```

Hide

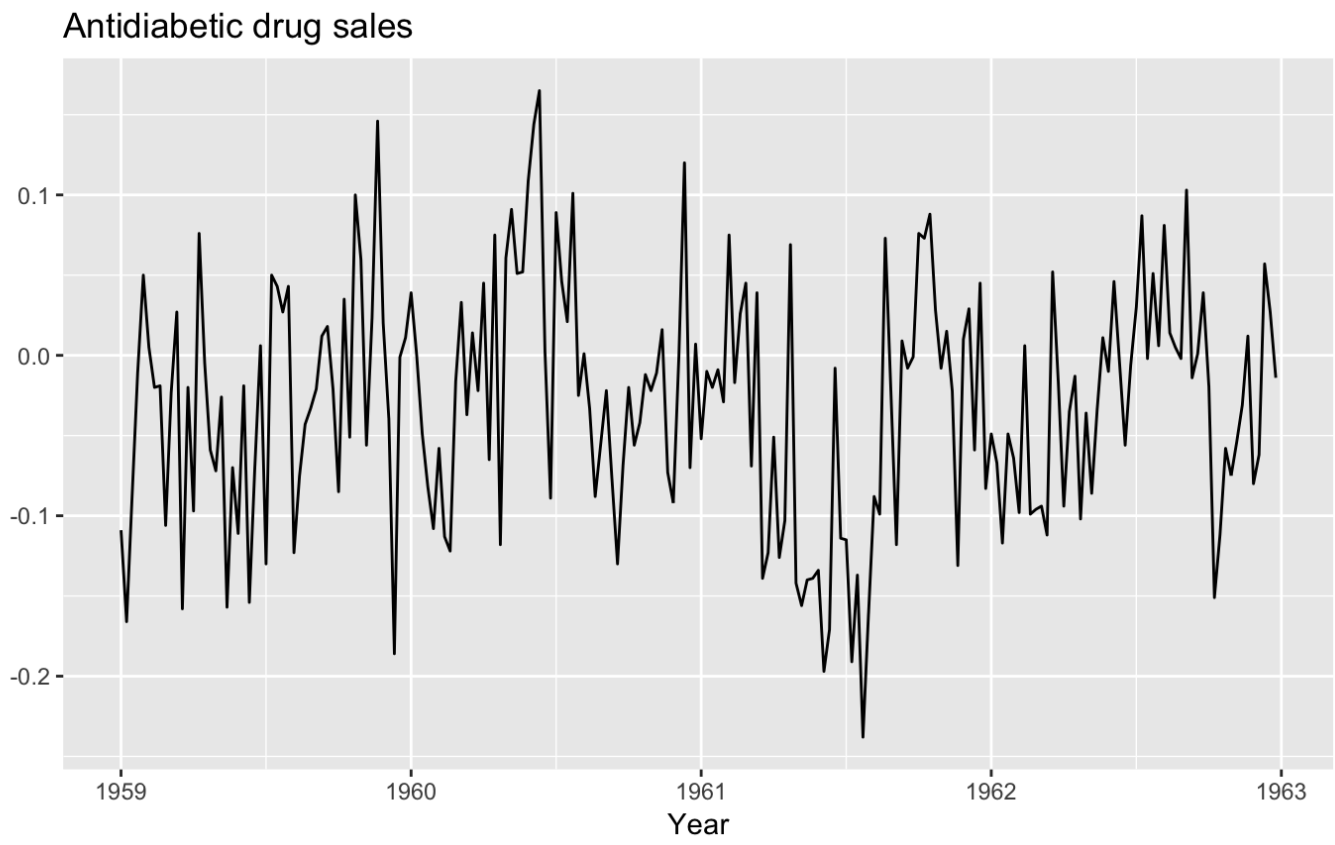
```
Box.test(diff(diff(crest_training)), type = "Ljung-Box")
```

```
    Box-Ljung test

data:  diff(diff(crest_training))
X-squared = 110.08, df = 1, p-value < 2.2e-16
```

This looks like a bad idea so we are going to try other techniques.

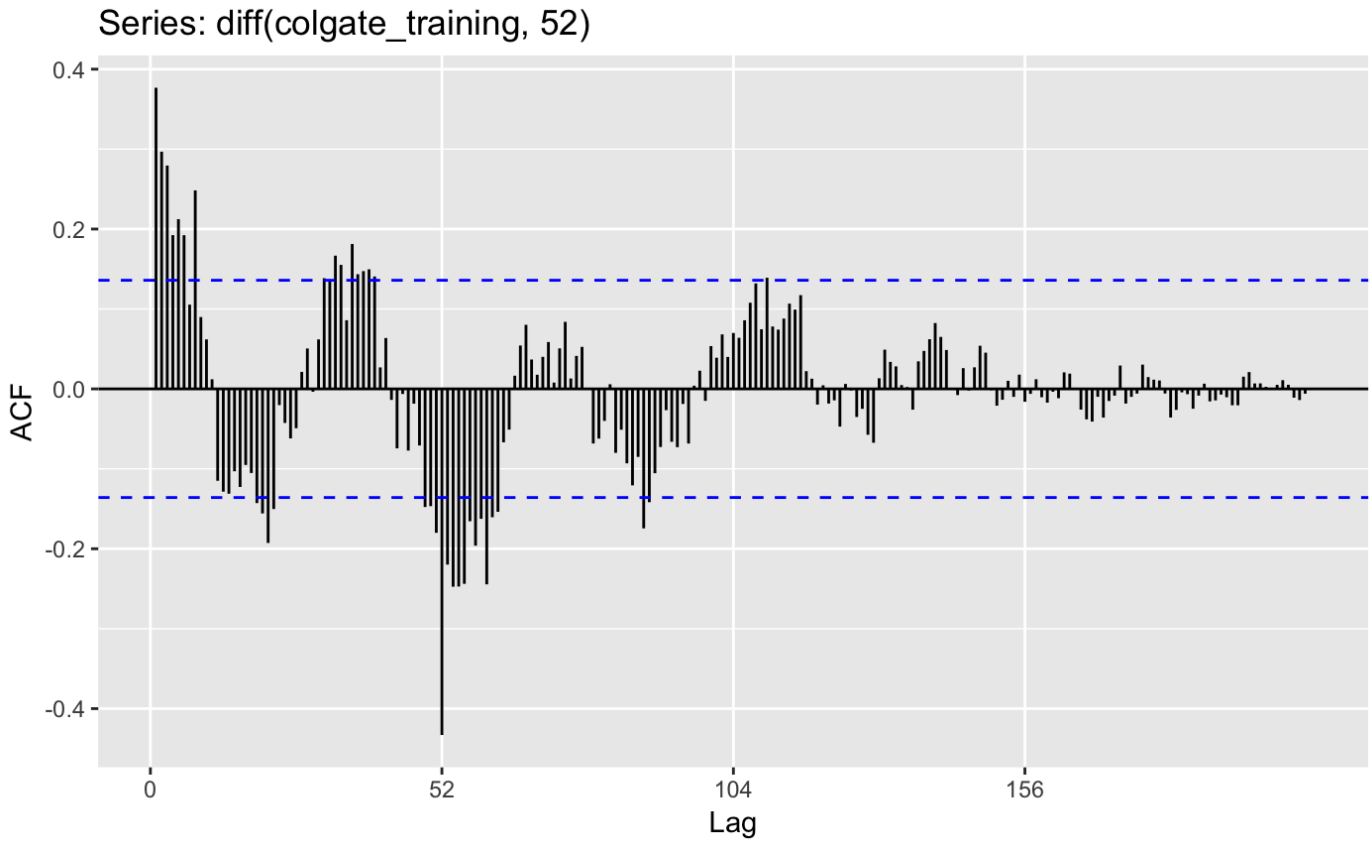
Hide

```
autoplot(diff(colgate_training,52)) +
    xlab("Year") + ylab("") +
    ggtitle("Antidiabetic drug sales")
```



Hide

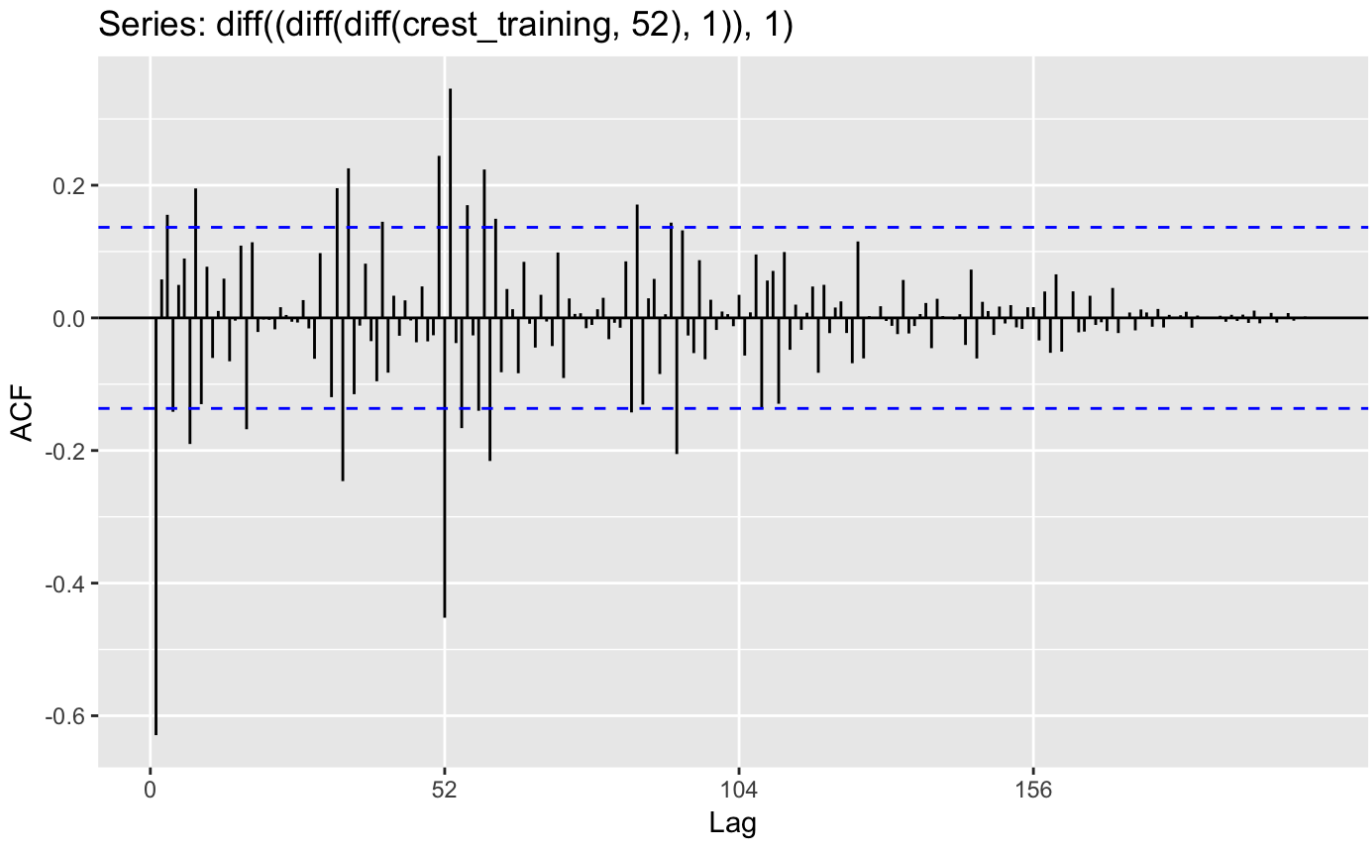
```
ggAcf(diff(colgate_training,52), lag.max = 260)
```



Let's try one last thing:

Hide

```
ggAcf(diff((diff(diff(crest_training,52),1)),1), lag.max = 260)
```



One way of knowing if it is necessary to differentiate is to run a KPSS hypothesis test. In this test, the null hypothesis is that the data are stationary, and we look for evidence that the null hypothesis is false. Consequently, small p-values (e.g., less than 0.05) suggest that differencing is required.

Hide

```
library(urca)
summary(ur.kpss(colgate_training))
```

```
#######################
# KPSS Unit Root Test #
#######################

Test is of type: mu with 5 lags.

Value of test-statistic is: 3.0286

Critical value for a significance level of:
                10pct  5pct 2.5pct  1pct
critical values 0.347 0.463  0.574 0.739
```

Hide

```
summary(ur.kpss(crest_training))
```

```
#######################
# KPSS Unit Root Test #
#######################

Test is of type: mu with 5 lags.

Value of test-statistic is: 3.9322

Critical value for a significance level of:
                10pct  5pct 2.5pct  1pct
critical values 0.347 0.463  0.574 0.739
```

In both cases the stadistic is much bigger than the 1% critical value, indicating that the null hypothesis is rejected. That is, the data are not

stationary. We can difference the data, and apply the test again.

Hide

```
summary(ur.kpss(diff(colgate_training)))
```

```
#######################
# KPSS Unit Root Test #
#######################

Test is of type: mu with 5 lags.

Value of test-statistic is: 0.0163

Critical value for a significance level of:
                10pct  5pct 2.5pct  1pct
critical values 0.347 0.463  0.574 0.739
```

Hide

```
summary(ur.kpss(diff(crest_training)))
```

```
#######################
# KPSS Unit Root Test #
#######################

Test is of type: mu with 5 lags.

Value of test-statistic is: 0.0237

Critical value for a significance level of:
                10pct  5pct 2.5pct  1pct
critical values 0.347 0.463  0.574 0.739
```

This time, the test statistic is tiny, and well within the range we would expect for stationary data. So we can conclude that the differenced data are stationary.

Through the function ndiffs() we can estimate how many times it is necessary to differentiate:

Hide

```
ndiffs(colgate_training)
```

```
[1] 1
```

Hide

```
ndiffs(crest_training)
```

```
[1] 1
```

The function nsdiffs() indicates how many times is necessary to seasonal differentiate a series. In this case we can see it is not necessary:

Hide

```
nsdiffs(colgate_training)
```

```
[1] 0
```

Hide

```
nsdiffs(crest_training)
```

```
[1] 0
```

Fitting the ARIMA models

Next we are going to try to fit an ARIMA model for each series. Here we are supposing the data is seasonal:

Hide

```
(fit_colgate <- auto.arima(colgate_training, seasonal = TRUE, stepwise = FALSE, approximation = FALSE))
```

```
Series: colgate_training
ARIMA(0,1,1)

Coefficients:
          ma1
      -0.7586
s.e.   0.0467

sigma^2 estimated as 0.002312:  log likelihood=418.57
AIC=-833.14   AICc=-833.09   BIC=-826.02
```

Hide

```
(fit_crest <- auto.arima(crest_training, seasonal = TRUE, stepwise = FALSE, approximation = FALSE))
```

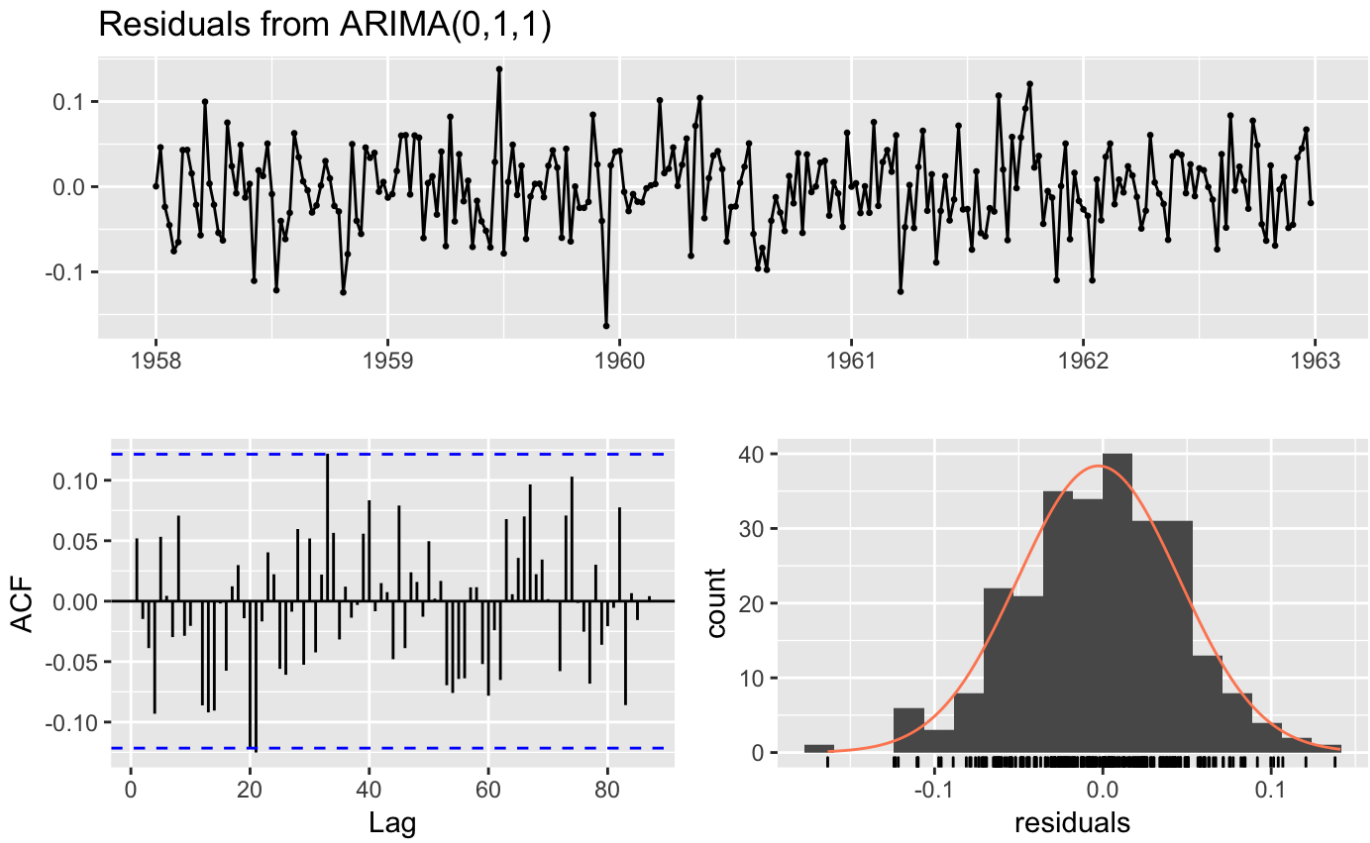
```
Series: crest_training
ARIMA(0,1,1)

Coefficients:
          ma1
      -0.6494
s.e.   0.0448

sigma^2 estimated as 0.002054:  log likelihood=434.08
AIC=-864.15   AICc=-864.1   BIC=-857.04
```

The prediction intervals for ARIMA models are based on assumptions that the residuals are uncorrelated and normally distributed. If either of these assumptions does not hold, then the prediction intervals may be incorrect. For this reason, always plot the ACF and histogram of the residuals to check the assumptions before producing prediction intervals.

Hide

```
checkresiduals(fit_colgate)
```

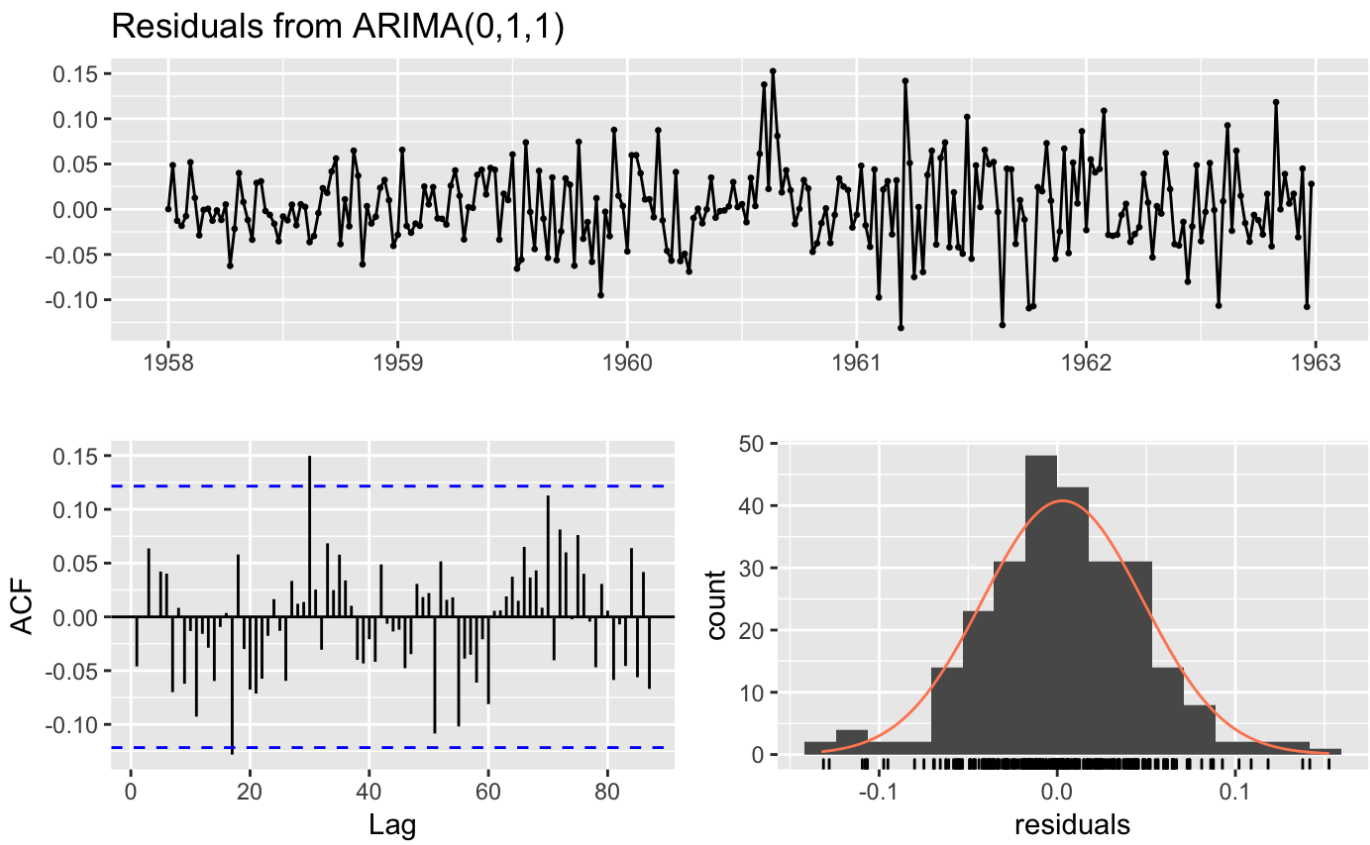
```

    Ljung-Box test

data:  Residuals from ARIMA(0,1,1)
Q* = 42.207, df = 51, p-value = 0.8049

Model df: 1.   Total lags used: 52
```



Hide

```
checkresiduals(fit_crest)
```

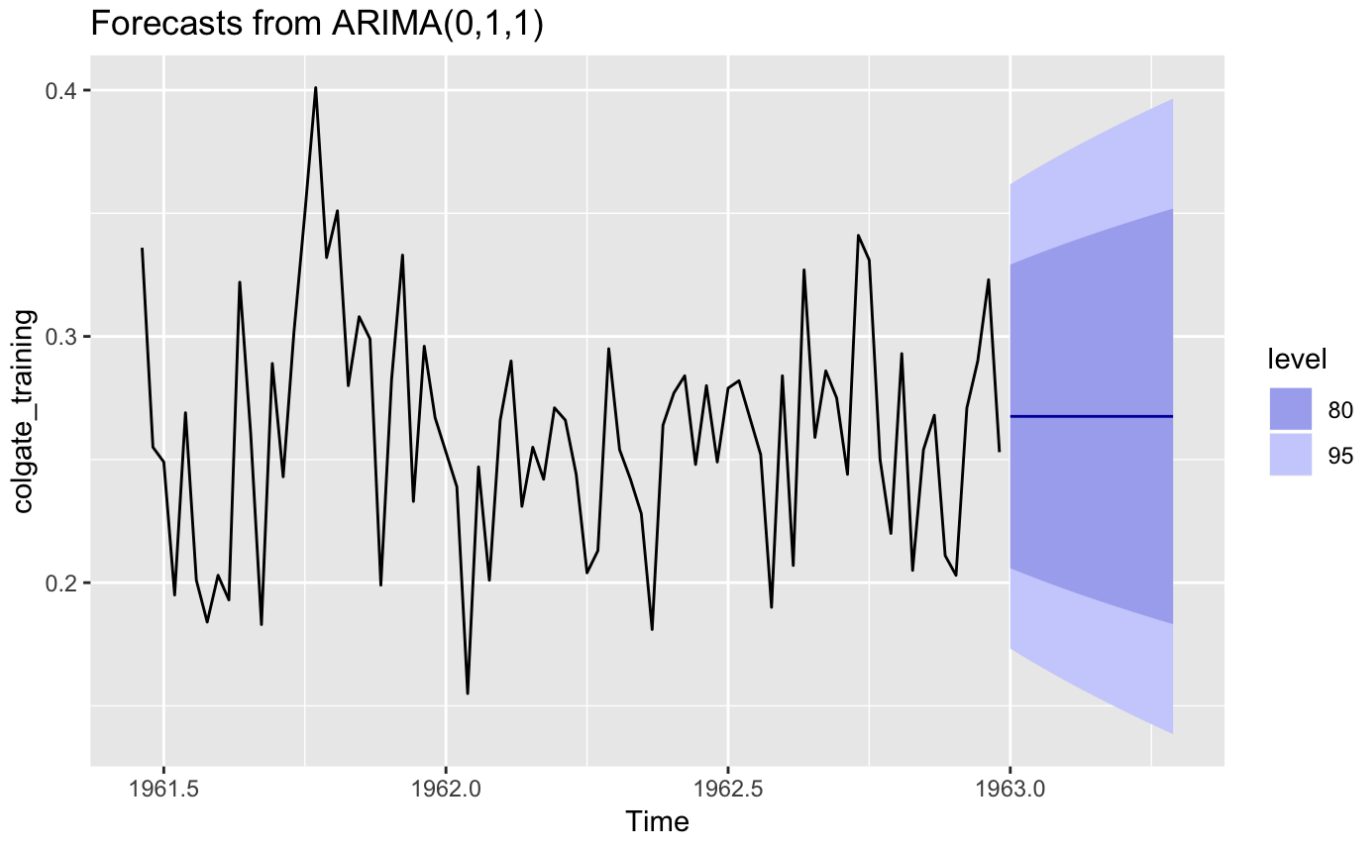
```
    Ljung-Box test

data:  Residuals from ARIMA(0,1,1)
Q* = 38.886, df = 51, p-value = 0.893

Model df: 1.   Total lags used: 52
```



And now that the residuals are white noise we can compute the forecast for both series:

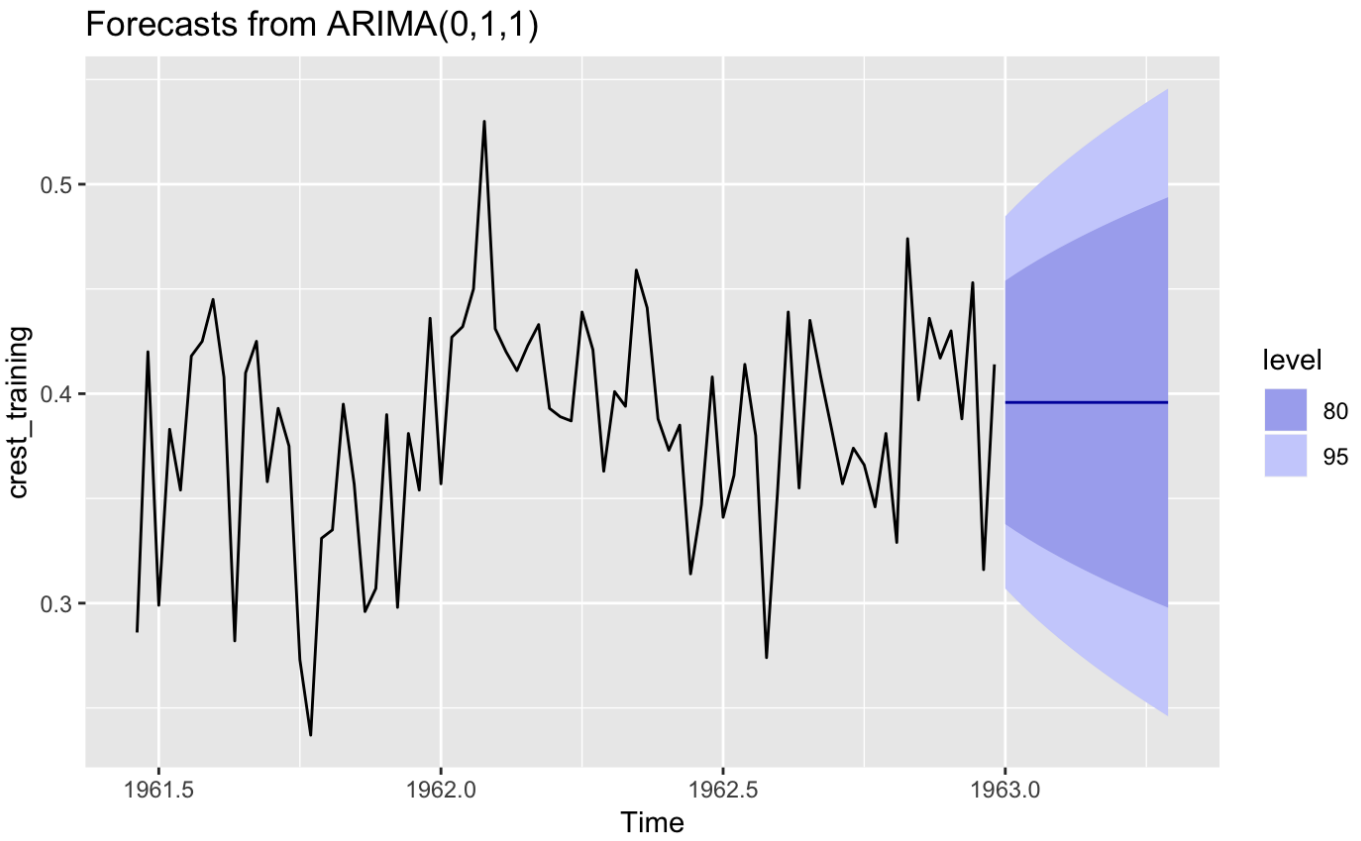
Hide

```
fit_colgate %>% forecast(h = 16) %>% autoplot(include = 80)
```



Hide

```
fit_crest %>% forecast(h = 16) %>% autoplot(include = 80)
```



Let's check its accuracy on the test set:

(Note. Because the model was not re-estimated, the "residuals" obtained here are actually one-step forecast errors. Consequently, the results produced from the accuracy() command are actually on the test set (despite the output saying "Training set").)

Hide

```
colgate.test <- Arima(colgate_test, model = fit_colgate)
accuracy(colgate.test)
```

```
                      ME       RMSE        MAE       MPE     MAPE MASE       ACF1
Training set -0.003197035 0.03878165 0.02788997 -3.788818 12.29345  NaN -0.1149332
```

Hide

```
crest.test <- Arima(crest_test, model = fit_crest)
accuracy(crest.test)
```

```
                     ME       RMSE        MAE        MPE     MAPE MASE       ACF1
Training set 0.001937123 0.05026208 0.04073115 -0.7771382 9.977746  NaN -0.2258668
```

Fitting outliers model

R provides (more precisely the tsoutliers package provides) functions to detect outliers.

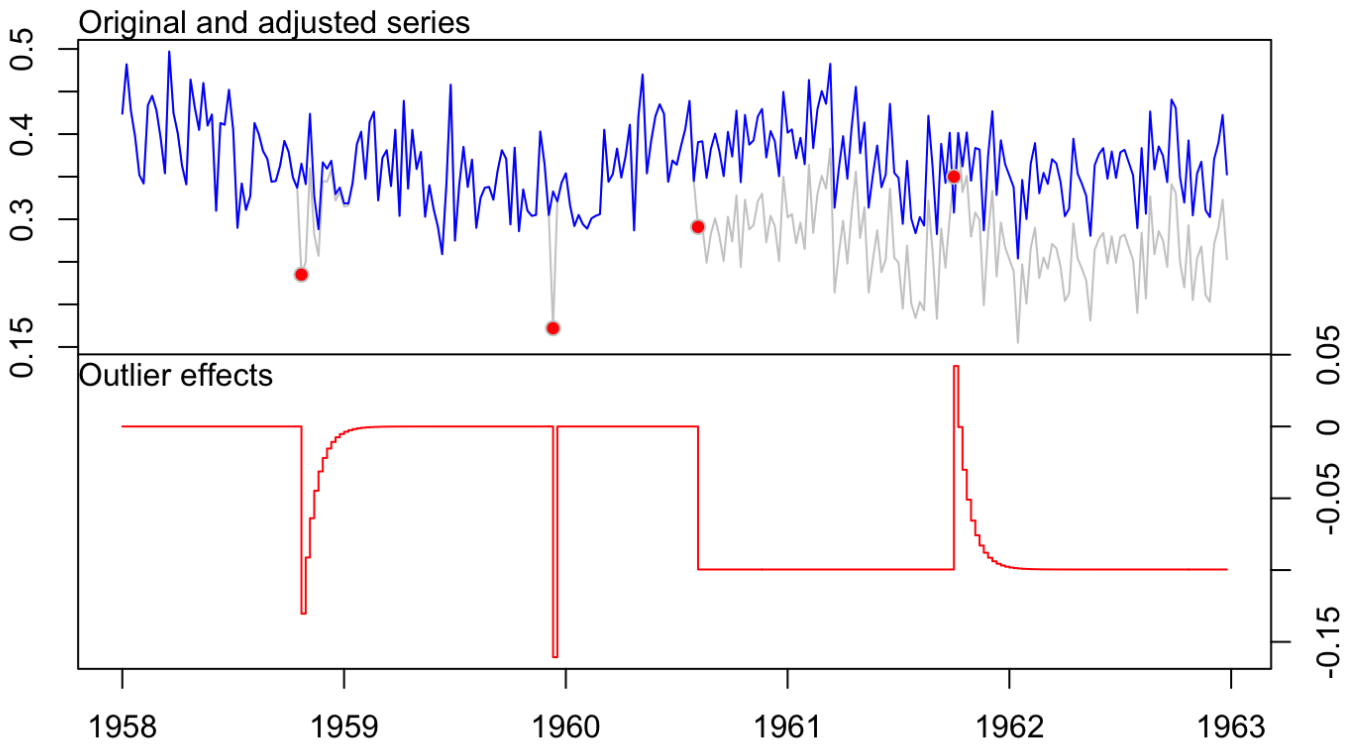
Hide

```
library(tsoutliers)
(colgate_outlier <- tso(colgate_training, types = c("TC", "AO", "LS", "IO", "SLS")))
```

```
Series: colgate_training
Regression with ARIMA(0,1,1)(1,0,0)[52] errors

Coefficients:
          ma1     sar1     TC43    AO102    LS136   TC196
      -0.8621  -0.0036  -0.1303  -0.1607  -0.0996  0.1417
s.e.   0.0333   0.0698   0.0346   0.0418   0.0223  0.0344

sigma^2 estimated as 0.001918:  log likelihood=445.04
AIC=-876.09   AICc=-875.64   BIC=-851.19

Outliers:
```

Hide

```
plot(colgate_outlier)
```



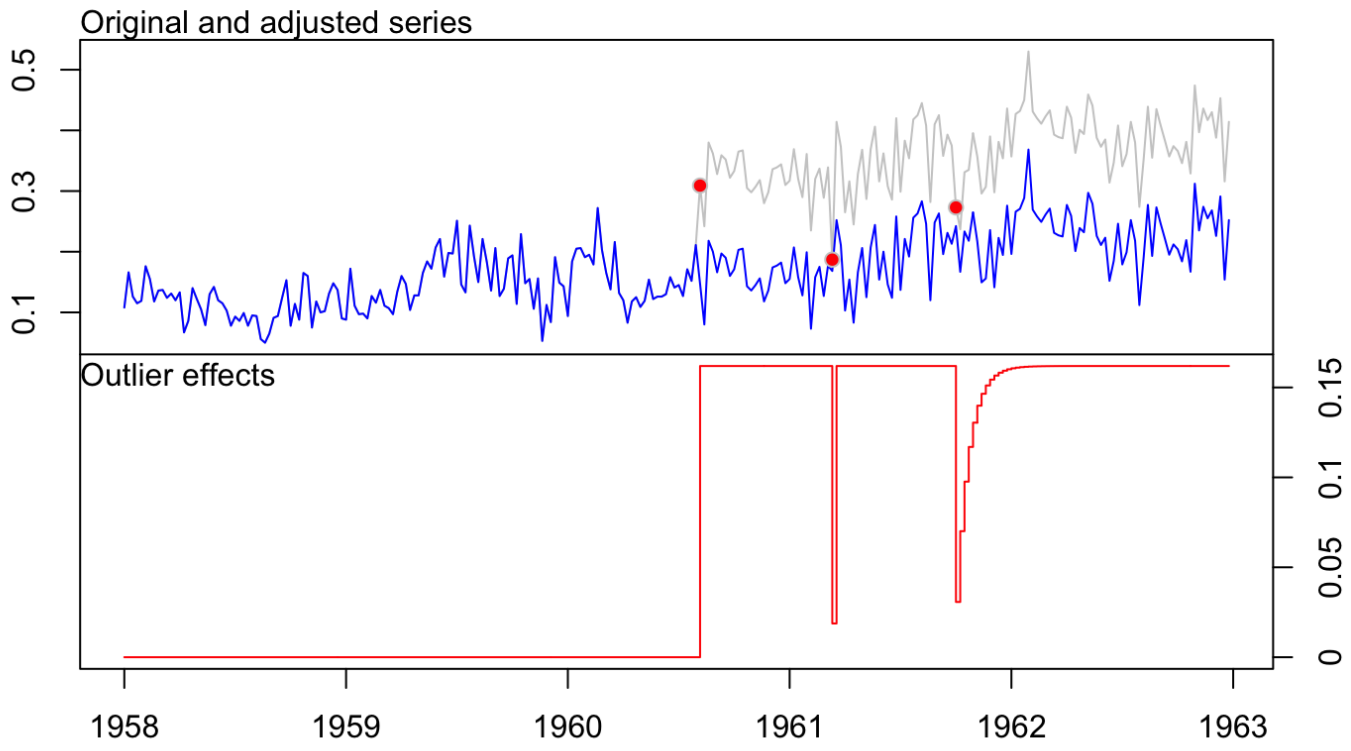
Hide

```
(crest_outlier <- tso(crest_training, types = c("TC", "AO", "LS", "IO", "SLS")))
```

```
Series: crest_training
Regression with ARIMA(0,1,1) errors

Coefficients:
          ma1   LS136    AO167    TC196
      -0.7725  0.1619  -0.1432  -0.1312
s.e.   0.0477  0.0270   0.0387   0.0342

sigma^2 estimated as 0.001712:  log likelihood=458.97
AIC=-907.95   AICc=-907.71   BIC=-890.16

Outliers:
```

Hide

```
plot(crest_outlier)
```



Hide

```
summary(crest_outlier)
```

```
         Length Class      Mode
outliers   5    data.frame list
y        260    ts         numeric
yadj     260    ts         numeric
cval       1    -none-     numeric
fit       19    ARIMA      list
effects  260    ts         numeric
times      3    -none-     numeric
```

These process has given us two great advantages. Firstly, we have detected all the outliers and plotted how the adjusted series would be. Secondly, we have built two new ARIMA models with remarkably better AICc coeffitients.

Now we are going to create new forecasts for this models:

Hide

```
(fit_colgate_outlier <- auto.arima(colgate_outlier$yadj, seasonal = TRUE, stepwise = FALSE, approximation = FALSE))
```

```
Series: colgate_outlier$yadj
ARIMA(1,0,1) with non-zero mean

Coefficients:
         ar1      ma1    mean
      0.9587  -0.8274  0.3692
s.e.  0.0271   0.0490  0.0106

sigma^2 estimated as 0.001866:  log likelihood=449.22
AIC=-890.45   AICc=-890.29   BIC=-876.2
```

Hide

```
(fit_crest_outlier <- auto.arima(crest_outlier$yadj, seasonal = TRUE, stepwise = FALSE, approximation = FALSE))
```

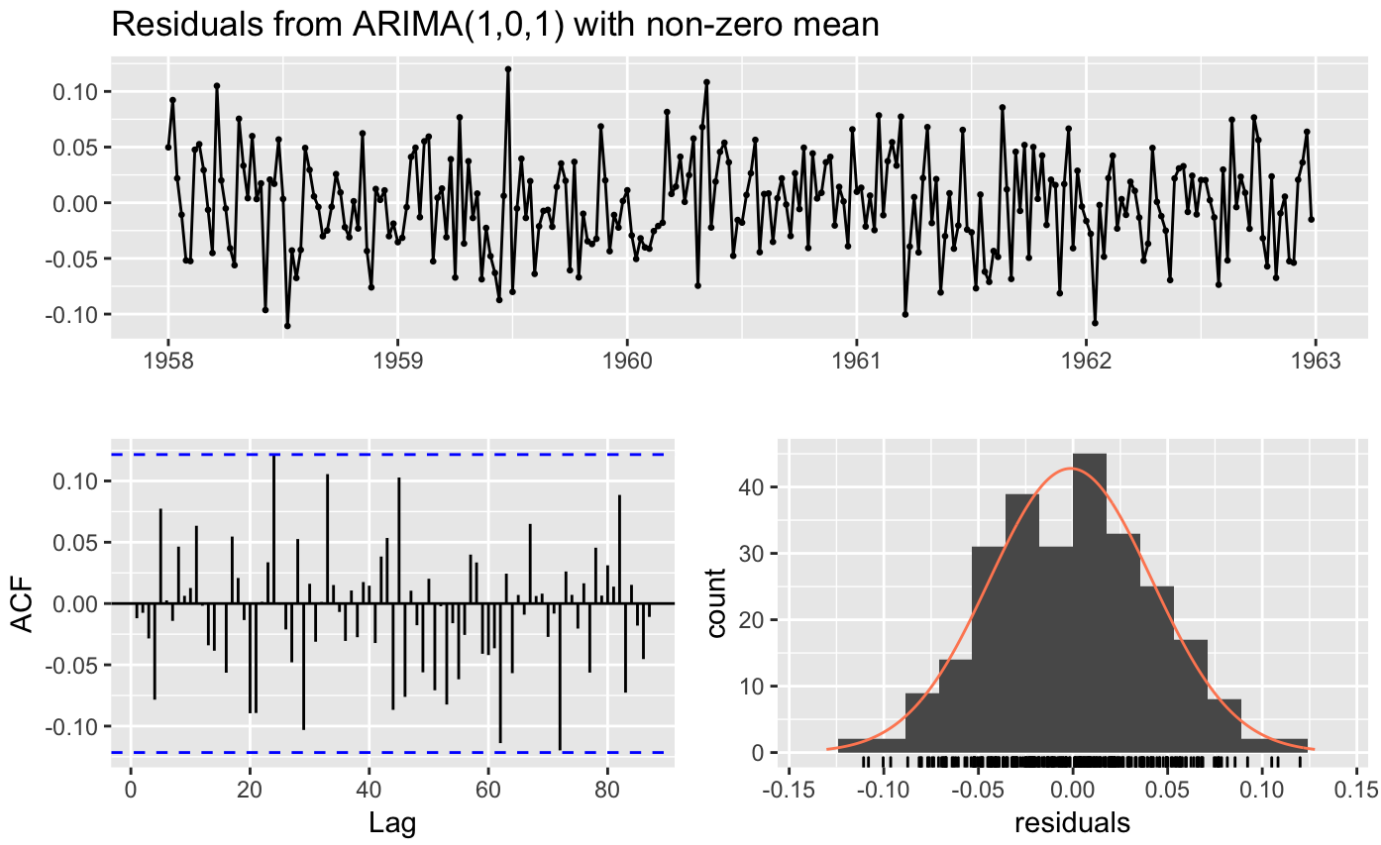
```
Series: crest_outlier$yadj
ARIMA(0,1,1)

Coefficients:
          ma1
      -0.7724
s.e.   0.0459

sigma^2 estimated as 0.001692:  log likelihood=458.97
AIC=-913.95   AICc=-913.9   BIC=-906.83
```

We see again a quite good improval in the AICc coeffitient. This will be our definite model for prediction. We check its residuals and compute the forecasts:

Hide

```
checkresiduals(fit_colgate_outlier)
```

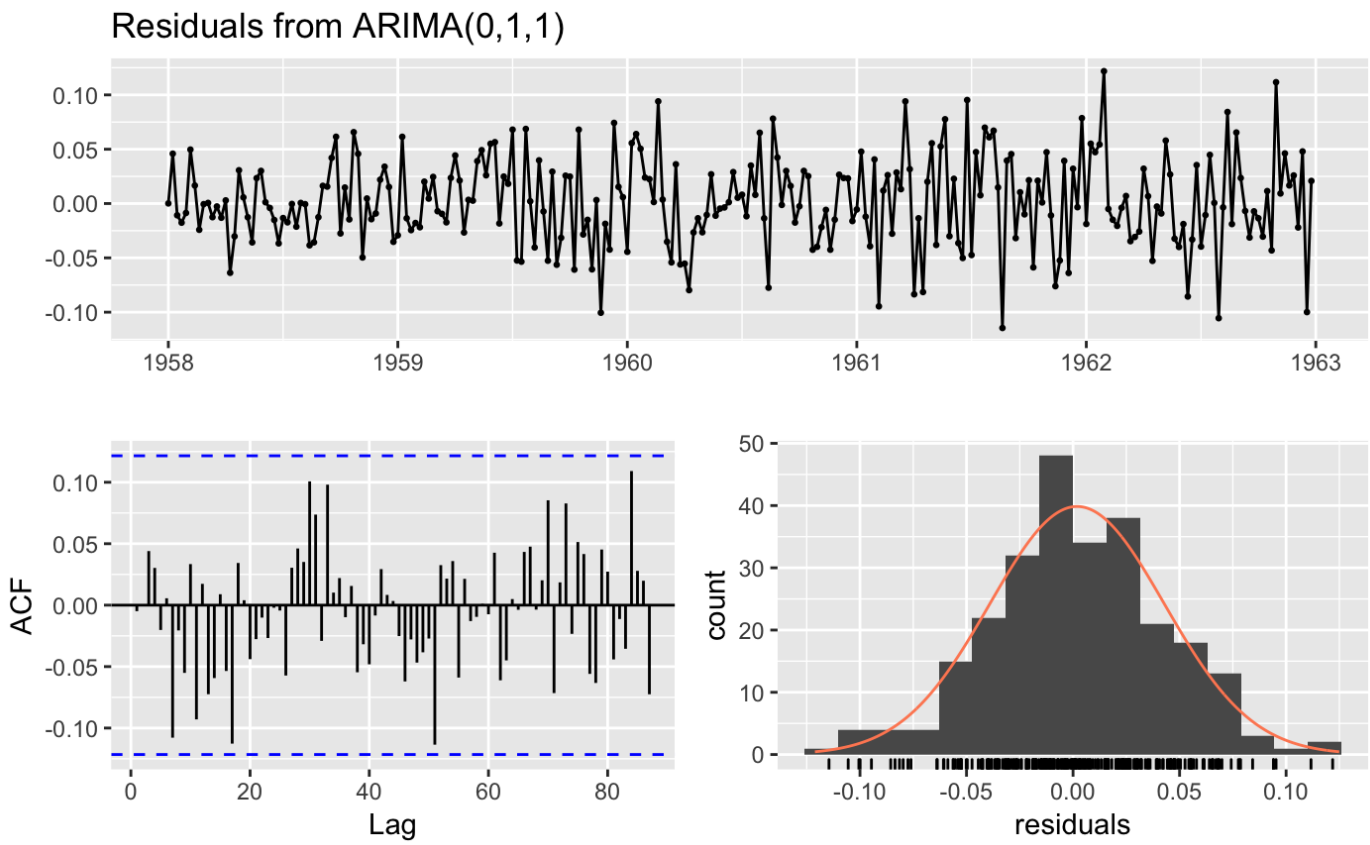
```
    Ljung-Box test

data:  Residuals from ARIMA(1,0,1) with non-zero mean
Q* = 38.533, df = 49, p-value = 0.8588

Model df: 3.   Total lags used: 52
```



Hide

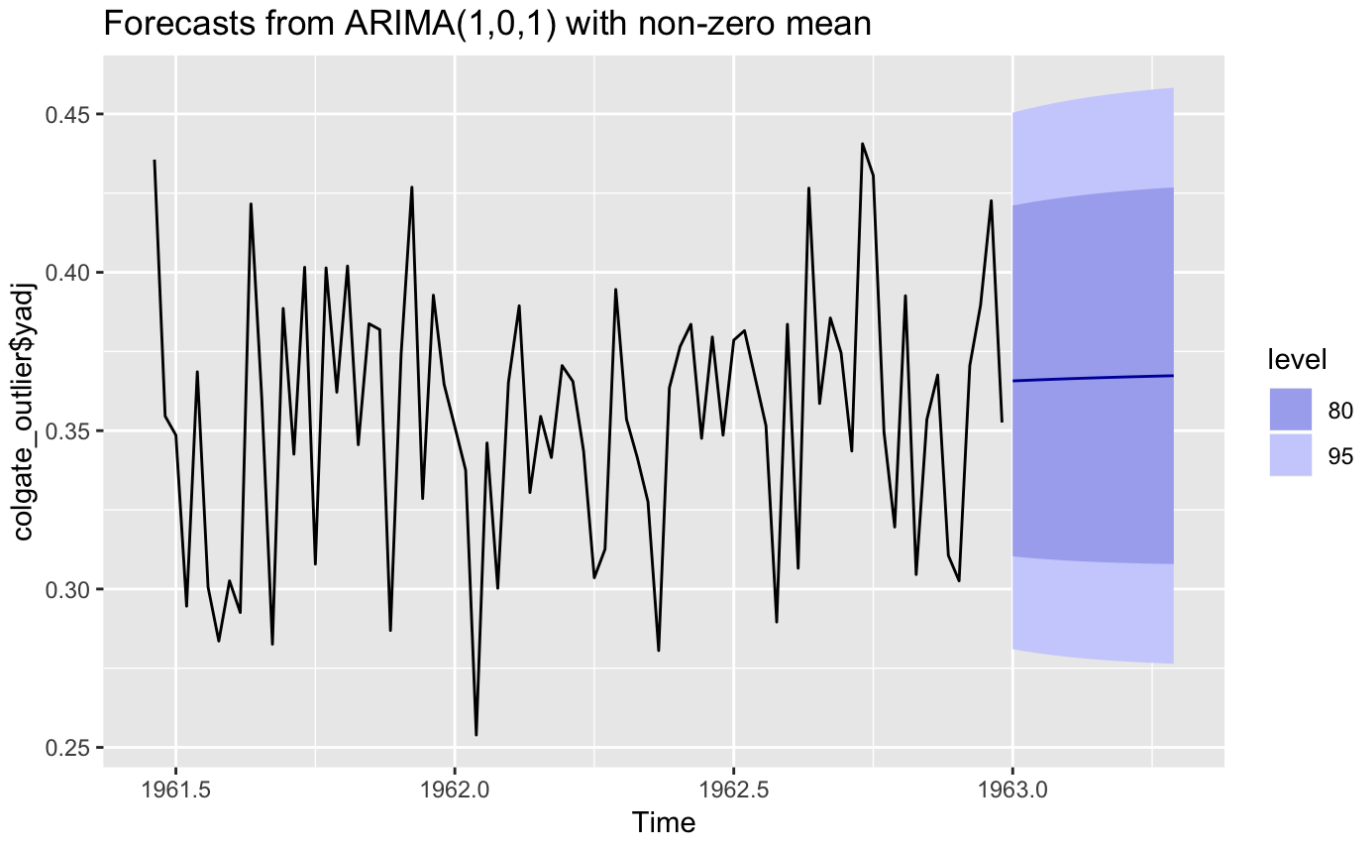
```
checkresiduals(fit_crest_outlier)
```

```
    Ljung-Box test

data:  Residuals from ARIMA(0,1,1)
Q* = 35.999, df = 51, p-value = 0.9446

Model df: 1.   Total lags used: 52
```



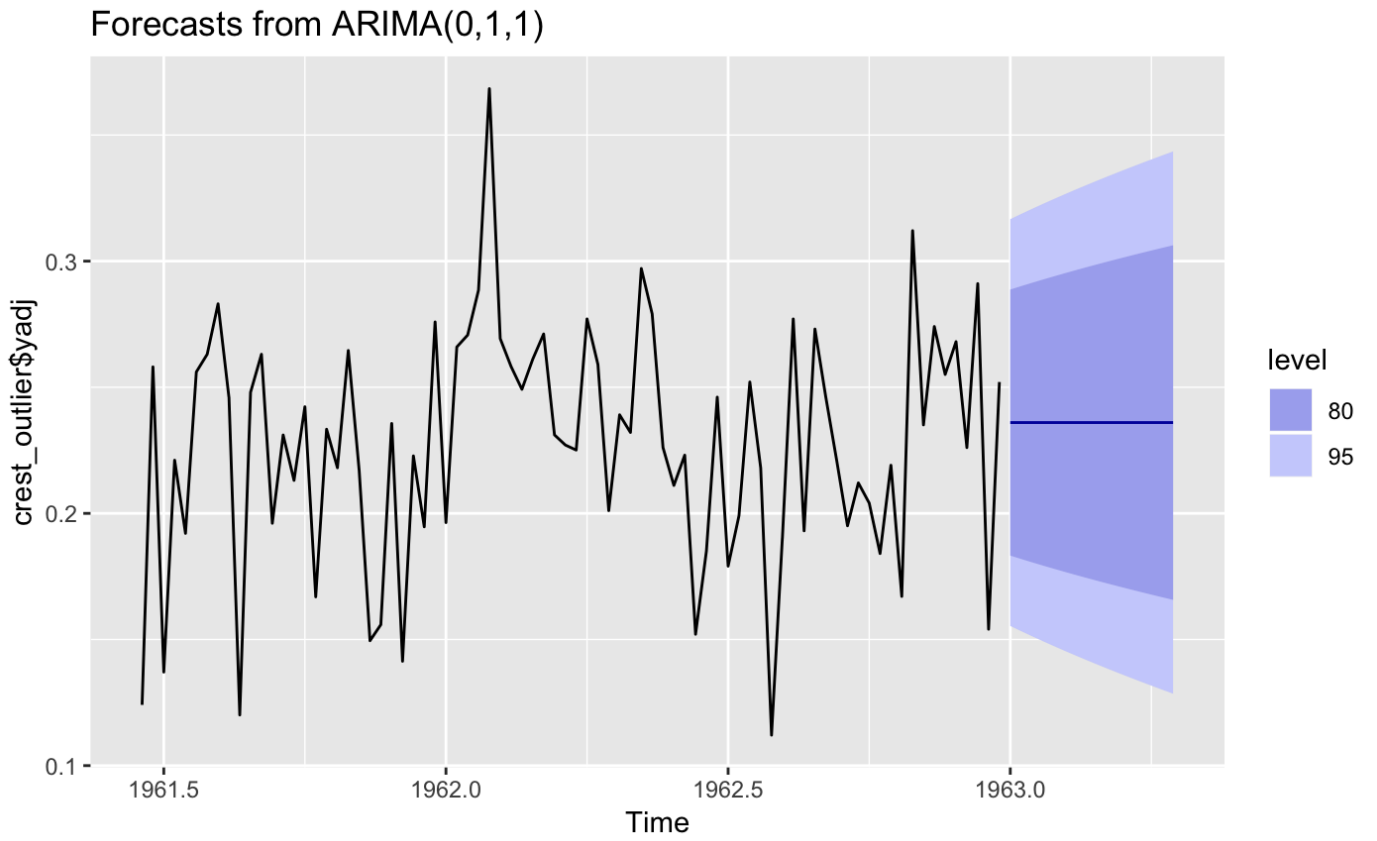
Hide

```
fit_colgate_outlier %>% forecast(h = 16) %>% autoplot(include = 80)
```



Hide

```
fit_crest_outlier %>% forecast(h = 16) %>% autoplot(include = 80)
```



Note that Arima() does not re-estimate in this case. Instead, the model obtained previously (and stored as cafe.train) is applied to the test data. Because the model was not re-estimated, the "residuals" obtained here are actually one-step forecast errors. Consequently, the results produced from the accuracy() command are actually on the test set (despite the output saying "Training set").

Hide

```
colgate.test <- Arima(colgate_test, model = fit_colgate_outlier)
accuracy(colgate.test)
```

```
                     ME       RMSE        MAE       MPE     MAPE MASE       ACF1
Training set -0.05689585 0.07107308 0.06340114 -27.29515 29.36141  NaN 0.08552327
```

Hide

```
crest.test <- Arima(crest_test, model = fit_crest_outlier)
accuracy(crest.test)
```

```
                      ME       RMSE        MAE        MPE     MAPE MASE       ACF1
Training set 0.002414962 0.04782108 0.03947827 -0.6235671 9.657663  NaN -0.1928718
```

This looks quite similar to the previous predictions but the big difference is that now we have a more firm idea this is true (the AICc has been reduced, the plots show white noise and the Ljung-Box test bears it out).

Causal Impact

Note. This part is not necessary for the assignment but gives us a better understanding of the data.

In this section we will work with the aggregated dataset, so called, mydata.

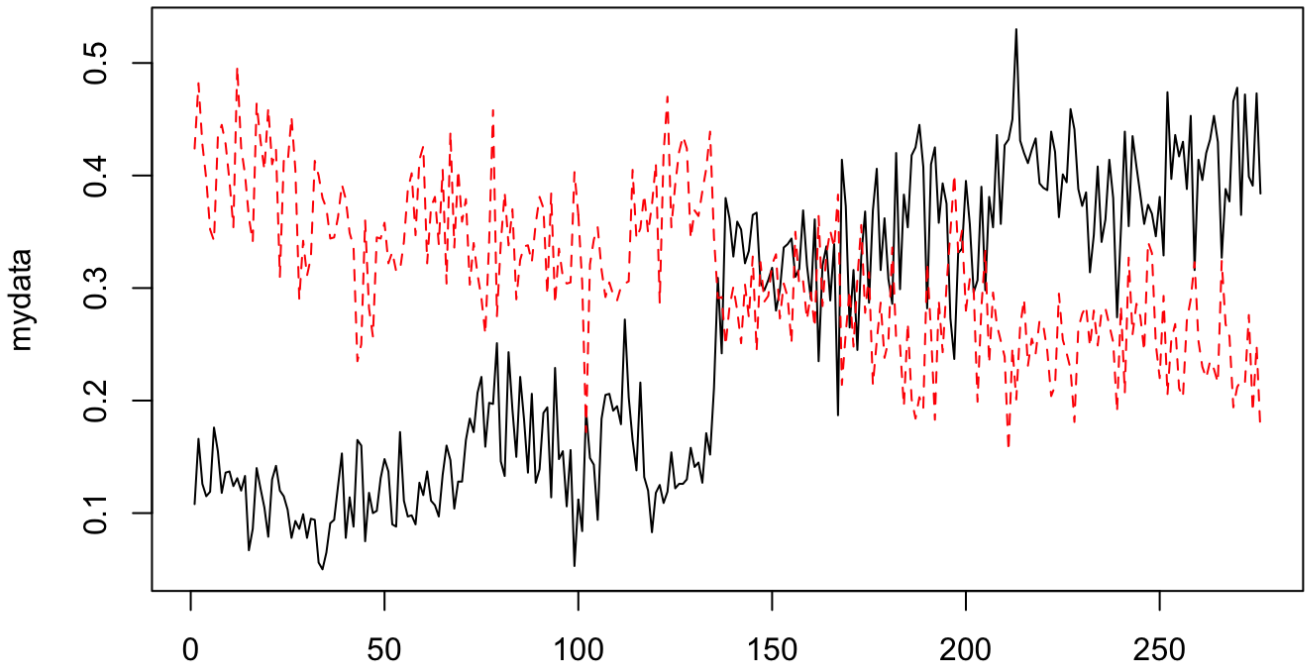
Hide

```
dim(mydata)
```

```
[1] 276   2
```

Hide

```
matplot(mydata, type = "l" )
```



In the above graphic we see Colgate's market share in red and Crest's market share in black.

Now to use the library Causal Impact we require a zoo object so we are going to create one:

Hide

```
library(zoo)
period <- seq(as.Date('1958/01/08'), as.Date('1962/12/26'), by = 'week')
data_zoo <- zoo(cbind(training_data$Crest, training_data$Colgate), period)
crest_zoo <- zoo(training_data$Crest, period)
colgate_zoo <- zoo(training_data$Colgate, period)
```

To estimate a causal effect, we begin by specifying which period in the data should be used for training the model (pre-intervention period) and which period for computing a counterfactual prediction (post-intervention period). We have checked before that the effects of the intervention start the 31st week of 1960 which corresponds to the 135th register:

Hide

```
pre.period <- as.Date(c("1958-01-08","1960-07-27"))
post.period <- as.Date(c("1960-08-03","1962-12-26"))
```

Now we are ready to perform the causal analysis. We have two choices here: performing with the whole data or for each brand. We do both firstly cause technically is really easy. We'll talk about the interpretation later:

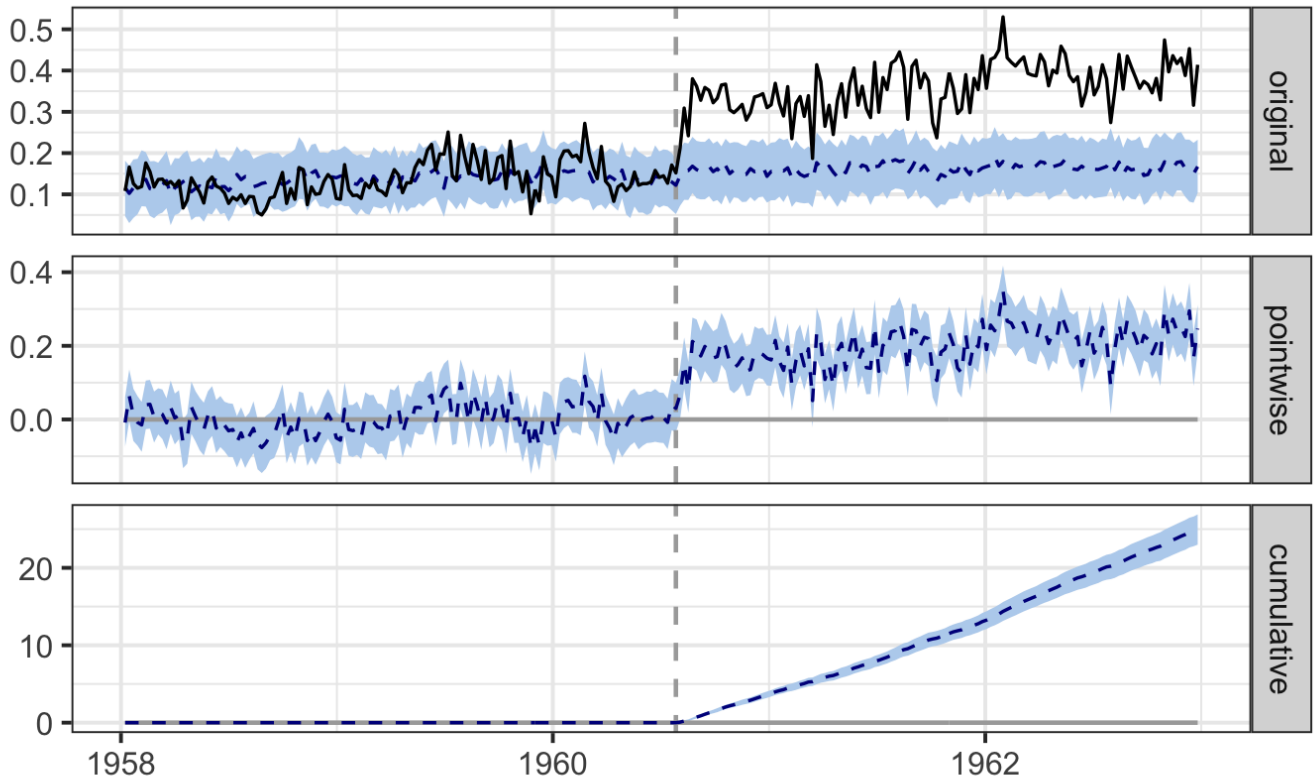
Hide

```
library(CausalImpact)
```

```
Loading required package: bsts
Loading required package: BoomSpikeSlab
Loading required package: Boom
Loading required package: MASS

Attaching package: 'MASS'

The following objects are masked from 'package:fma':

    cement, housing, petrol


Attaching package: 'Boom'

The following object is masked from 'package:stats':

    rWishart

Loading required package: xts
```

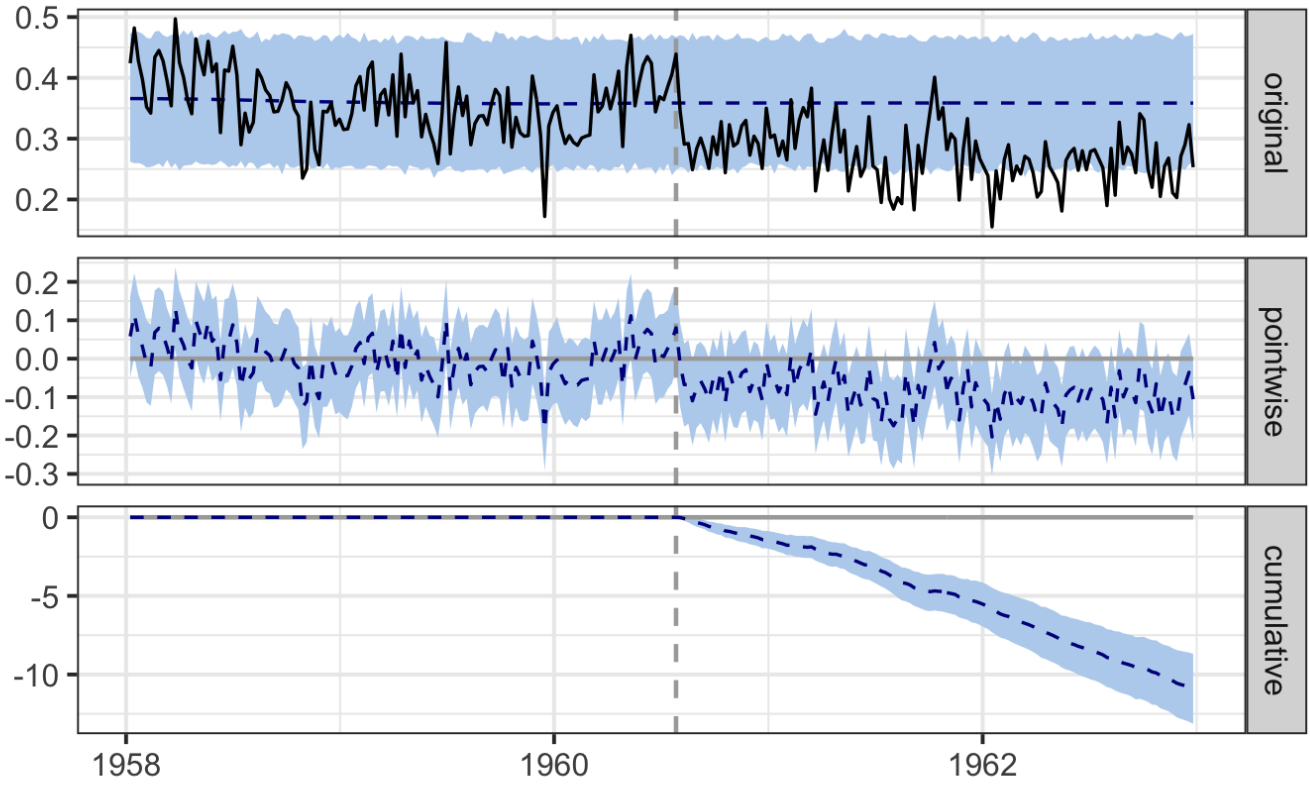
Hide

```
impact_combined <- CausalImpact(data_zoo, pre.period, post.period)
plot(impact_combined)
```



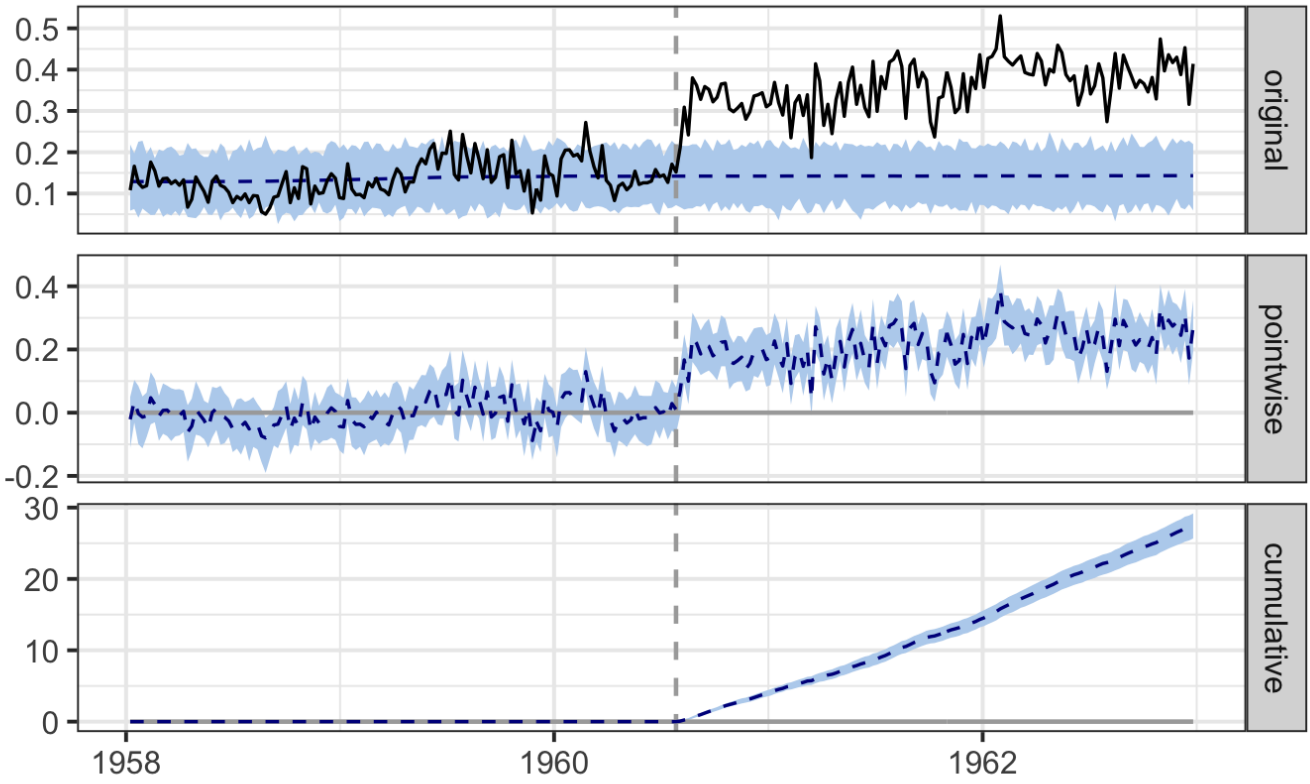
Hide

```
impact_Colgate <- CausalImpact(colgate_zoo, pre.period, post.period)
impact_Crest <- CausalImpact(crest_zoo, pre.period, post.period)
plot(impact_Colgate)
```



Hide

```
plot(impact_Crest)
```



Hide

```
print("Summary for the whole data")
```

```
[1] "Summary for the whole data"
```

Hide

```
summary(impact_combined)
```

```
Posterior inference {CausalImpact}

                         Average         Cumulative
Actual                   0.36            45.49
Prediction (s.d.)        0.16 (0.0079)   20.47 (0.9993)
95% CI                   [0.15, 0.18]    [18.60, 22.49]

Absolute effect (s.d.)   0.2 (0.0079)    25.0 (0.9993)
95% CI                   [0.18, 0.21]    [23.00, 26.89]

Relative effect (s.d.)   122% (4.9%)     122% (4.9%)
95% CI                   [112%, 131%]    [112%, 131%]

Posterior tail-area probability p:   0.00108
Posterior prob. of a causal effect:  99.89189%

For more details, type: summary(impact, "report")
```

Hide

```
print("Summary for Colgate")
```

```
[1] "Summary for Colgate"
```

Hide

```
summary(impact_Colgate)
```

```
Posterior inference {CausalImpact}

                         Average          Cumulative
Actual                   0.27             34.34
Prediction (s.d.)        0.36 (0.009)     45.19 (1.136)
95% CI                   [0.34, 0.38]     [43.03, 47.47]

Absolute effect (s.d.)   -0.086 (0.009)   -10.850 (1.136)
95% CI                   [-0.1, -0.069]   [-13.1, -8.688]

Relative effect (s.d.)   -24% (2.5%)      -24% (2.5%)
95% CI                   [-29%, -19%]     [-29%, -19%]

Posterior tail-area probability p:   0.00108
Posterior prob. of a causal effect:  99.89166%

For more details, type: summary(impact, "report")
```

Hide

```
print("Summary for Crest")
```

```
[1] "Summary for Crest"
```

Hide

```
summary(impact_Crest)
```

```
Posterior inference {CausalImpact}

                         Average         Cumulative
Actual                   0.36            45.49
Prediction (s.d.)        0.14 (0.0074)   17.96 (0.9302)
95% CI                   [0.13, 0.16]    [16.32, 19.82]

Absolute effect (s.d.)   0.22 (0.0074)   27.53 (0.9302)
95% CI                   [0.2, 0.23]     [25.7, 29.17]

Relative effect (s.d.)   153% (5.2%)     153% (5.2%)
95% CI                   [143%, 162%]    [143%, 162%]

Posterior tail-area probability p:   0.01042
Posterior prob. of a causal effect:  98.958%

For more details, type: summary(impact, "report")
```

Which interpretation would be:

Hide

```
print("WHOLE DATA")
```

```
[1] "WHOLE DATA"
```

Hide

```
summary(impact_combined, "report")
```

```
Analysis report {CausalImpact}


During the post-intervention period, the response variable had an average value of approx. 0.36. By contrast
, in the absence of an intervention, we would have expected an average response of 0.16. The 95% interval of
this counterfactual prediction is [0.15, 0.18]. Subtracting this prediction from the observed response yield
s an estimate of the causal effect the intervention had on the response variable. This effect is 0.20 with a
95% interval of [0.18, 0.21]. For a discussion of the significance of this effect, see below.

Summing up the individual data points during the post-intervention period (which can only sometimes be meani
ngfully interpreted), the response variable had an overall value of 45.49. By contrast, had the intervention
not taken place, we would have expected a sum of 20.47. The 95% interval of this prediction is [18.60, 22.49
].

The above results are given in terms of absolute numbers. In relative terms, the response variable showed an
increase of +122%. The 95% interval of this percentage is [+112%, +131%].

This means that the positive effect observed during the intervention period is statistically significant and
unlikely to be due to random fluctuations. It should be noted, however, that the question of whether this in
crease also bears substantive significance can only be answered by comparing the absolute effect (0.20) to t
he original goal of the underlying intervention.

The probability of obtaining this effect by chance is very small (Bayesian one-sided tail-area probability p
= 0.001). This means the causal effect can be considered statistically significant.
```

Hide

```
print("")
```

```
[1] ""
```

Hide

```
print("COLGATE")
```

```
[1] "COLGATE"
```

Hide

```
summary(impact_Colgate, "report")
```

```
Analysis report {CausalImpact}


During the post-intervention period, the response variable had an average value of approx. 0.27. By contrast
, in the absence of an intervention, we would have expected an average response of 0.36. The 95% interval of
this counterfactual prediction is [0.34, 0.38]. Subtracting this prediction from the observed response yield
s an estimate of the causal effect the intervention had on the response variable. This effect is -0.086 with
a 95% interval of [-0.10, -0.069]. For a discussion of the significance of this effect, see below.

Summing up the individual data points during the post-intervention period (which can only sometimes be meani
ngfully interpreted), the response variable had an overall value of 34.34. By contrast, had the intervention
not taken place, we would have expected a sum of 45.19. The 95% interval of this prediction is [43.03, 47.47
].

The above results are given in terms of absolute numbers. In relative terms, the response variable showed a
decrease of-24%. The 95% interval of this percentage is [-29%, -19%].

This means that the negative effect observed during the intervention period is statistically significant. If
the experimenter had expected a positive effect, it is recommended to double-check whether anomalies in the
control variables may have caused an overly optimistic expectation of what should have happened in the respo
nse variable in the absence of the intervention.

The probability of obtaining this effect by chance is very small (Bayesian one-sided tail-area probability p
= 0.001). This means the causal effect can be considered statistically significant.
```

Hide

```
print("")
```

```
[1] ""
```

Hide

```
print("CREST")
```

```
[1] "CREST"
```

Hide

```
summary(impact_Crest, "report")
```

```
Analysis report {CausalImpact}


During the post-intervention period, the response variable had an average value of approx. 0.36. By contrast
, in the absence of an intervention, we would have expected an average response of 0.14. The 95% interval of
this counterfactual prediction is [0.13, 0.16]. Subtracting this prediction from the observed response yield
s an estimate of the causal effect the intervention had on the response variable. This effect is 0.22 with a
95% interval of [0.20, 0.23]. For a discussion of the significance of this effect, see below.

Summing up the individual data points during the post-intervention period (which can only sometimes be meani
ngfully interpreted), the response variable had an overall value of 45.49. By contrast, had the intervention
not taken place, we would have expected a sum of 17.96. The 95% interval of this prediction is [16.32, 19.82
].

The above results are given in terms of absolute numbers. In relative terms, the response variable showed an
increase of +153%. The 95% interval of this percentage is [+143%, +162%].

This means that the positive effect observed during the intervention period is statistically significant and
unlikely to be due to random fluctuations. It should be noted, however, that the question of whether this in
crease also bears substantive significance can only be answered by comparing the absolute effect (0.22) to t
he original goal of the underlying intervention.

The probability of obtaining this effect by chance is very small (Bayesian one-sided tail-area probability p
= 0.01). This means the causal effect can be considered statistically significant.
```

Let’s recall where the outliers were:

Hide

```
tso(colgate_training, types = c("TC", "AO", "LS", "IO", "SLS"))
```

```
Series: colgate_training
Regression with ARIMA(0,1,1)(1,0,0)[52] errors

Coefficients:
          ma1     sar1     TC43    AO102    LS136   TC196
      -0.8621  -0.0036  -0.1303  -0.1607  -0.0996  0.1417
s.e.   0.0333   0.0698   0.0346   0.0418   0.0223  0.0344

sigma^2 estimated as 0.001918:  log likelihood=445.04
AIC=-876.09   AICc=-875.64   BIC=-851.19

Outliers:
```

Hide

```
tso(crest_training, types = c("TC", "AO", "LS", "IO", "SLS"))
```

```
Series: crest_training
Regression with ARIMA(0,1,1) errors

Coefficients:
          ma1   LS136    AO167    TC196
      -0.7725  0.1619  -0.1432  -0.1312
s.e.   0.0477  0.0270   0.0387   0.0342

sigma^2 estimated as 0.001712:  log likelihood=458.97
AIC=-907.95   AICc=-907.71   BIC=-890.16

Outliers:
```

Hide

```
(outliers_colgate_idx <- colgate_outlier$outliers$ind)
```

```
[1]  43 102 136 196
```

Hide

```
(outliers_crest_idx <- crest_outlier$outliers$ind)
```

```
[1] 136 167 196
```

Intervention Model

In this section we try to figure out how important was the impact of the intervention understanding as an intervention the ADA declaration and the marketing campaign:

Hide

```
arimax_model_crest <- arimax(crest_training,
                      order = c(0,1,1), #orden del ARIMA
                      xtransf = data.frame(I1 = (1*(seq(crest_training) == outliers_crest_idx))),  #matriz
con 1 donde haya outliers y 0 en lo demás
                      transfer = list(c(0,0)), #porque p=0
                      method = 'ML')
```

```
Error in arimax(crest_training, order = c(0, 1, 1), xtransf = data.frame(I1 = (1 *  :
  no se pudo encontrar la función "arimax"
```

Hide

```
arimax_model_colgate <- arimax(colgate_training,
                      order = c(1,0,1), #orden del ARIMA
                      xtransf = data.frame(I1 = (1*(seq(crest_training) == outliers_colgate_idx))),  #matr
iz con 1 donde haya outliers y 0 en lo demás
                      transfer = list(c(0,0)), #porque p=0
                      method = 'ML')
summary(arimax_model_crest)
```

```
Call:
arimax(x = crest_training, order = c(0, 1, 1), method = "ML", xtransf = data.frame(I1 = (1 *
    (seq(crest_training) == outliers_crest_idx))), transfer = list(c(0, 0)))

Coefficients:
          ma1   I1-MA0
      -0.6455  -0.0425
s.e.   0.0453   0.0289

sigma^2 estimated as 0.002029:  log likelihood = 435.15,  aic = -866.3

Training set error measures:
                      ME       RMSE        MAE       MPE     MAPE     MASE        ACF1
Training set 0.002985217 0.04495846 0.03423102 -2.889201 17.16969 0.816676 -0.04348054
```

Hide

```
library(lmtest)
```

```
Loading required package: zoo

Attaching package: 'zoo'

The following objects are masked from 'package:base':

    as.Date, as.Date.numeric
```

Hide

```
coeftest(arimax_model_crest)
```

```
z test of coefficients:

        Estimate Std. Error  z value Pr(>|z|)
ma1    -0.645454   0.045306 -14.2465   <2e-16 ***
I1-MA0 -0.042513   0.028922  -1.4699   0.1416
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

For our scenario, the xtranf parameter provides a value equal to 1 at the outliers time index and zero at others. The transfer parameter is a list consisting of the ARMA orders for each transfer covariate. For our scenario, we specify an AR order equal to 0:

Hide

```
arimax_model_colgate <- arimax(colgate_training,
                      order = c(0,1,1), #orden del ARIMA
                      xtransf = data.frame(I1 = (1*(seq(colgate_training) == outliers_colgate_idx))),  #ma
triz con 1 donde haya outliers y 0 en lo demás
                      transfer = list(c(0,0,0)), #porque p=0
                      method = 'ML')
summary(arimax_model_colgate)
```

```
Call:
arimax(x = colgate_training, order = c(0, 1, 1), method = "ML", xtransf = data.frame(I1 = (1 *
    (seq(colgate_training) == outliers_colgate_idx))), transfer = list(c(0,
    0)))

Coefficients:
          ma1   I1-MA0
      -0.7493  -0.0511
s.e.   0.0485   0.0318

sigma^2 estimated as 0.002281:  log likelihood = 419.85,  aic = -835.7

Training set error measures:
                      ME       RMSE        MAE       MPE    MAPE      MASE       ACF1
Training set -0.002505647 0.04766779 0.03764859 -2.961347 12.7051 0.8067333 0.05464463
```

The significance of the coefficients is then verified:

Hide

```
library(lmtest)
coeftest(arimax_model_colgate)
```

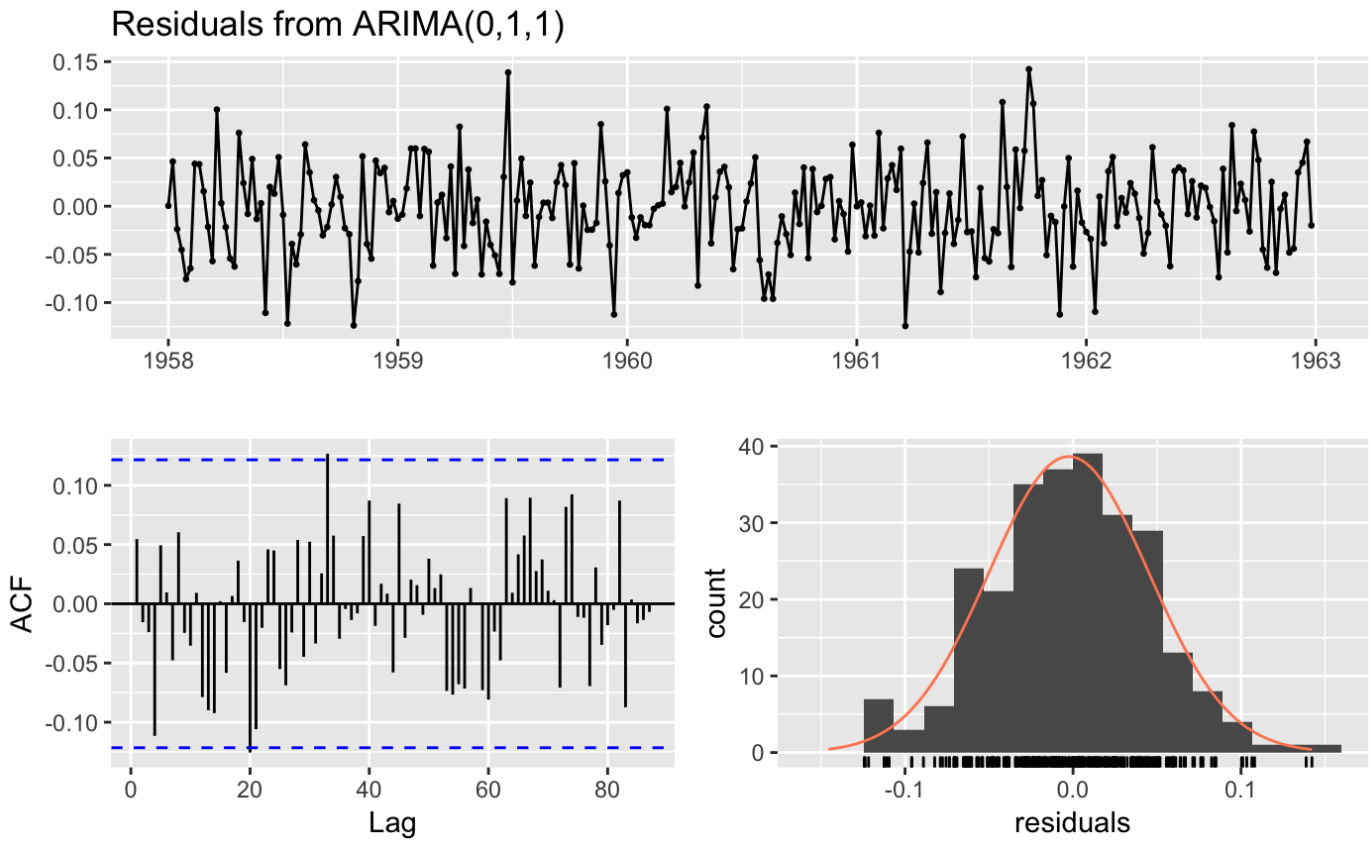
```
z test of coefficients:

        Estimate Std. Error  z value Pr(>|z|)
ma1    -0.749254   0.048506 -15.4467   <2e-16 ***
I1-MA0 -0.051097   0.031753  -1.6092   0.1076
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Lastly, let’s check the residuals:

Hide

```
checkresiduals(arimax_model_colgate)
```

```
    Ljung-Box test

data:  Residuals from ARIMA(0,1,1)
Q* = 43.311, df = 50, p-value = 0.7369

Model df: 2.   Total lags used: 52
```



Mutandis mutatis for Crest:

Hide

```
arimax_model_crest <- arimax(crest_training,
                      order = c(0,1,1), #orden del ARIMA
                      xtransf = data.frame(I1 = (1*(seq(crest_training) == outliers_crest_idx))),  #matriz con 1 donde haya outliers y 0 en lo demás
                      transfer = list(c(0,0,0)), #porque p=0
                      method = 'ML')
```

```
longitud de objeto mayor no es múltiplo de la longitud de uno menor
```

Hide

```
summary(arimax_model_crest)
```

```
Call:
arimax(x = crest_training, order = c(0, 1, 1), method = "ML", xtransf = data.frame(I1 = (1 *
    (seq(crest_training) == outliers_crest_idx))), transfer = list(c(0, 0, 0)))

Coefficients:
          ma1    I1-MA0
      -0.6455  -0.0425
s.e.   0.0453   0.0289

sigma^2 estimated as 0.002029:  log likelihood = 435.15,  aic = -866.3

Training set error measures:
                      ME       RMSE        MAE       MPE     MAPE     MASE        ACF1
Training set 0.002985217 0.04495846 0.03423102 -2.889201 17.16969 0.816676 -0.04348054
```

Hide

```
coeftest(arimax_model_crest)
```

```
z test of coefficients:

        Estimate Std. Error  z value Pr(>|z|)
ma1    -0.645454   0.045306 -14.2465   <2e-16 ***
I1-MA0 -0.042513   0.028922  -1.4699   0.1416
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Hide

```
checkresiduals(arimax_model_crest)
```

```
Error in is.element("lm", class(object)) :
  objeto 'arimax_model_crest' no encontrado
```

Transference Function

Hide

```
mod.transf.colgate <- arimax(x = colgate_training, order = c(1, 0, 1),
                             xtransf = data.frame(crest_training), # Atípico aditivo
                             transfer = list(c(0,0)), # Primero el step y luego el pulse
                             method = "ML")
mod.transf.colgate
```

```
Call:
arimax(x = colgate_training, order = c(1, 0, 1), method = "ML", xtransf = data.frame(crest_training),
    transfer = list(c(0, 0)))

Coefficients:
         ar1      ma1  intercept  crest_training-MA0
      0.9305  -0.7717     0.4322             -0.4568
s.e.  0.0452   0.0707     0.0156              0.0501

sigma^2 estimated as 0.001736:  log likelihood = 457.18,  aic = -906.35
```

Hide

```
coeftest(mod.transf.colgate)
```

```
z test of coefficients:

                    Estimate Std. Error  z value  Pr(>|z|)
ar1                 0.930525   0.045232  20.5724 < 2.2e-16 ***
ma1                -0.771749   0.070747 -10.9086 < 2.2e-16 ***
intercept           0.432169   0.015552  27.7884 < 2.2e-16 ***
crest_training-MA0 -0.456815   0.050104  -9.1173 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
```