YVR Energy Use Forecast

Yilun Lu

3/29/2020

Objective:

The main goal of this project is to develop a model to forecast monthly energy use for the Vancouver International Airport (YVR), for the next three years. If possible, we will also be investigating the reasons behind the unexpected surge or drop in energy use. Such might give us actionable insights which might aid airport management in decision making.

Background Information:

Second largest airport in Canada, Vancouver International Airport (YVR) requires huge amount of energy to light up, heat or cool, and control the massive facility each month. The energy cost usually accounts for a great portion of total expenses for airports in general. Therefore, being able to forecast the energy use accurately would better help financial analysts identify any future cost-related financial issues in the airport.

Approaches:

We are going to build a time series model to forecast YVR's energy use for the future months in the next three years. The best model will be selected from the ETS and ARIMA frameworks, based on residual diagnostics, generalization ability on the test set, and model complexity.

Bibliographical:

"U.S. Energy Information Administration - EIA - Independent Statistics and Analysis." How Much Electricity Does an American Home Use? - FAQ - U.S. Energy Information Administration (EIA), www.eia.gov/tools/faqs/faq.php?id=97&t=3.

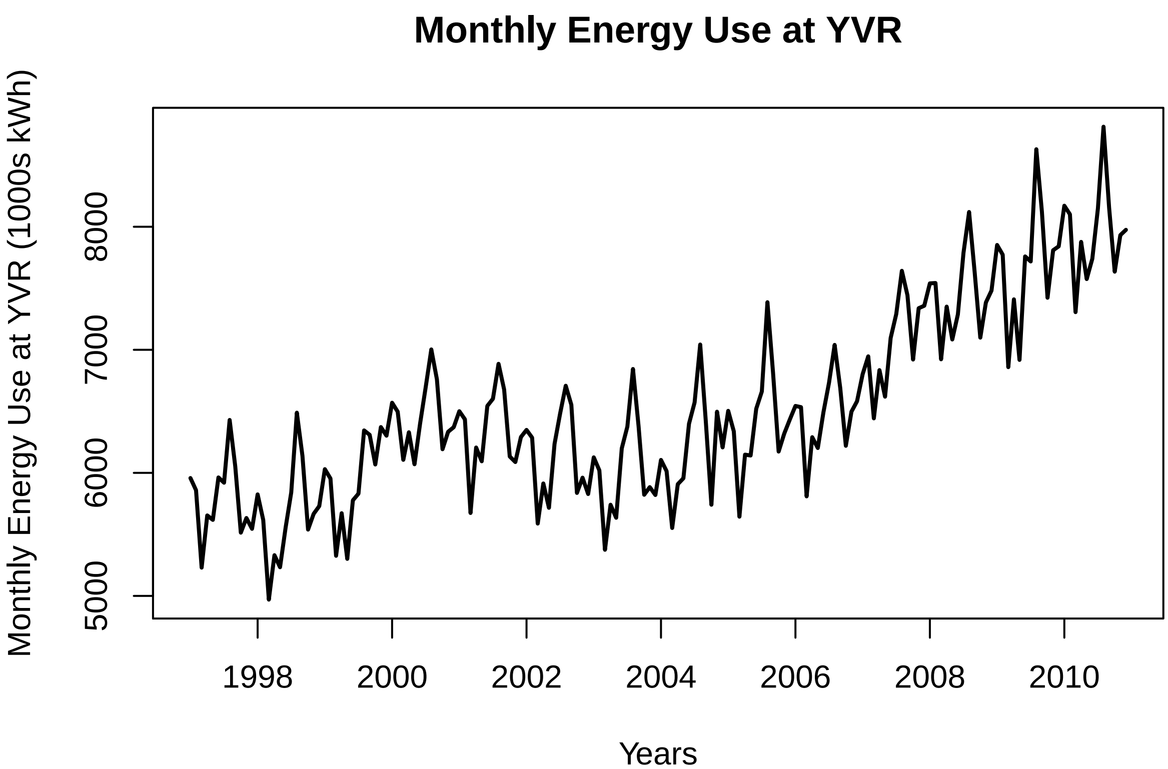
```r
library(fpp2)
```

```
## Loading required package: ggplot2

## Loading required package: forecast

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

## Registered S3 methods overwritten by 'forecast':
##   method             from
##   fitted.fracdiff    fracdiff
##   residuals.fracdiff fracdiff

## Loading required package: fma

## Loading required package: expsmooth
```

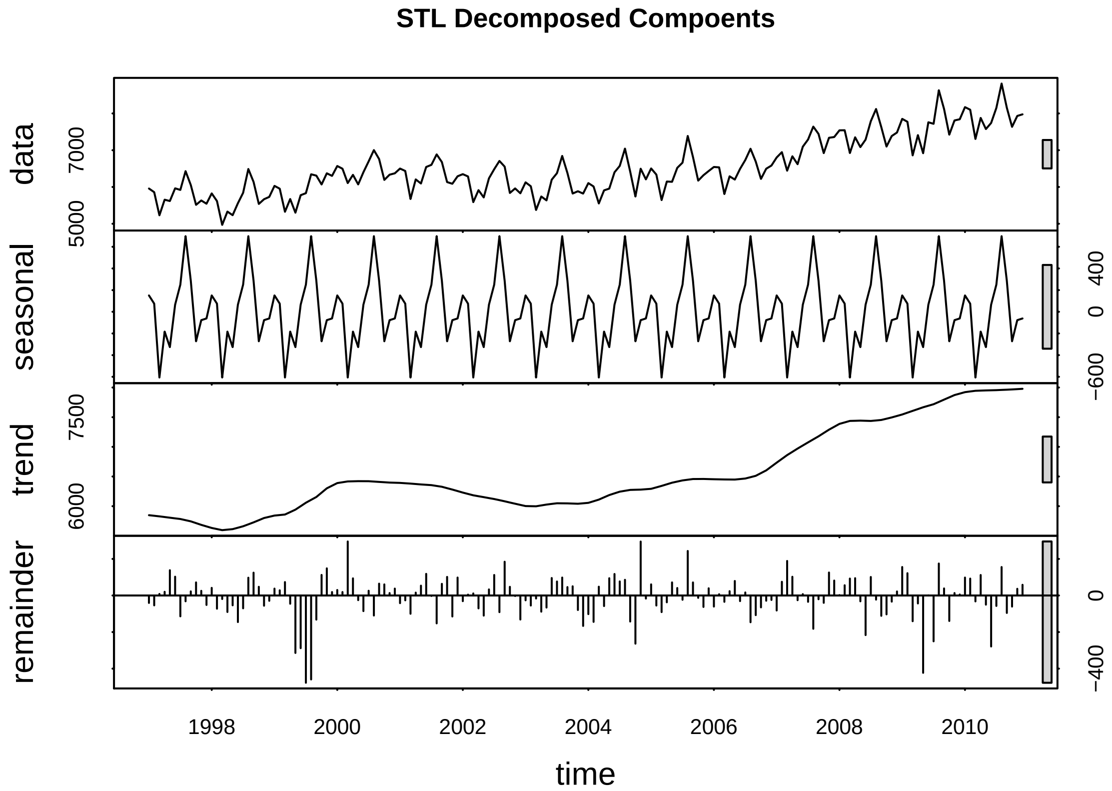
Introduction:

```r
df = read.csv("~/Documents/MBAN/BABS502/project/Energy_use_at _YVR.csv")

yvr = ts(df[, 2], start=c(1997, 1), frequency = 12) # only interested in the energy use column
```

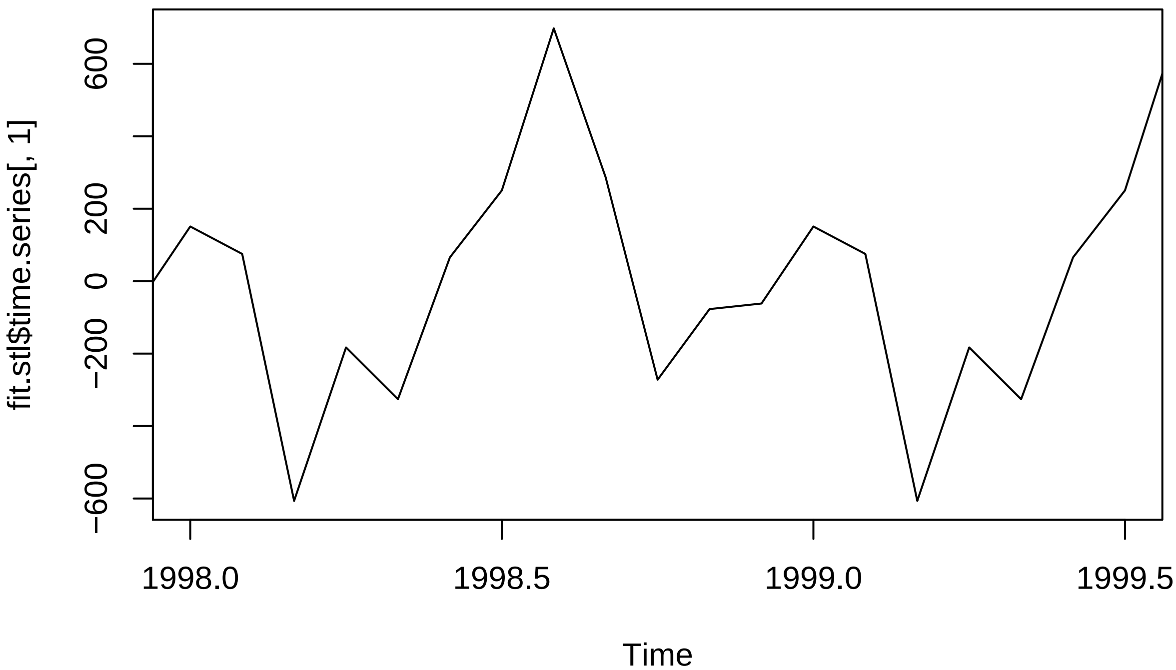
```r
# time plot
plot(yvr, xlab='Years', ylab='Monthly Energy Use at YVR (1000s kWh)', main='Monthly Energy Use at YVR',
```




```r
# STL Decomposition
fit.stl <- stl(yvr, t.window=15,
               s.window="periodic", robust=TRUE)
plot(fit.stl, main='STL Decomposed Compoents')
```




```r
plot(fit.stl$time.series[,1], xlim=c(1998.0, 1999.5))
```



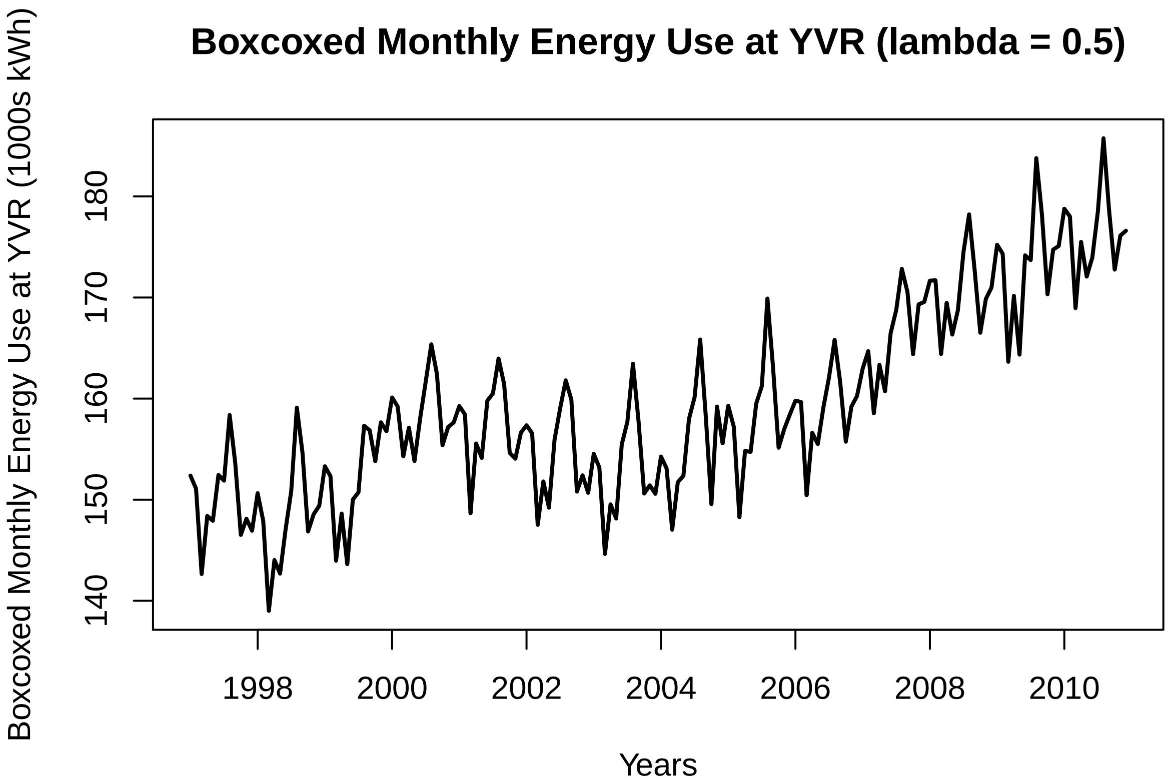
The discussion refers to the STL decomposition plot.

1) Cause for the faster increase in trend from 2007It could be that, more and more people are coming to Vancouver to live, study and do business. Therefore the increases in airlines raise the energy use.

2) Cause for the cycle from 1998 to 2003 and the unexpected energy drops between 1999 and 2000Such is likely to be caused by the creation of Energy Reduction Committee in 1999 to reduce energy use. The creation of committee might also explain the cycle from 1998 to 2003, which indicate the committee's effort was paid-off only temporarily.

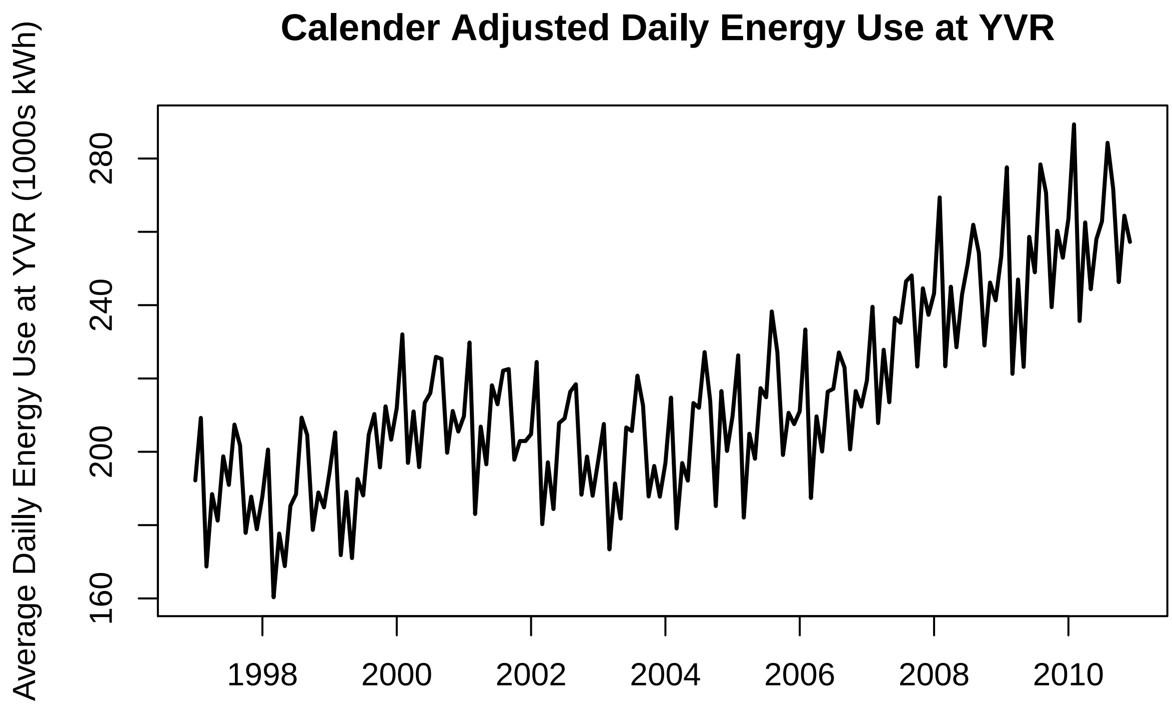
3) Explanation for the annual seasonalityThe seasonal pattern is quite intuitive. The energy use starts to rise in spring, reaches its peak in summer, drops again in fall, and eventually rises again in winter. Overall the energy consumption in summer is much greater than that in winter, probably because Vancouver's winter does not need too much heating in the airport.

```
yvr.boxcoxed = BoxCox(yvr, lambda=0.5)
plot(yvr.boxcoxed, xlab='Years', ylab='Boxcoxed Monthly Energy Use at YVR (1000s kWh)', main='Boxcoxed M
```



The purpose of the BoxCox Transformation is to even the seasonal variance across the times series so that we can better model the trend cycle (although in our case the variance isn't that heterogenous). The result of the box cox transformation isn't significant, which indicates that we might not have heterogenous seasonal variance in the first place.

```
monthdays <- rep(c(31,28,31,30,31,30,31,31,30,31,30,31),14)
monthdays[26 + (4*12)*(0:2)] <- 29
yvr.calender.adjusted = yvr/monthdays

plot(yvr.calender.adjusted, xlab='Years', ylab='Average Dailly Energy Use at YVR (1000s kWh)', main=' Ca
```

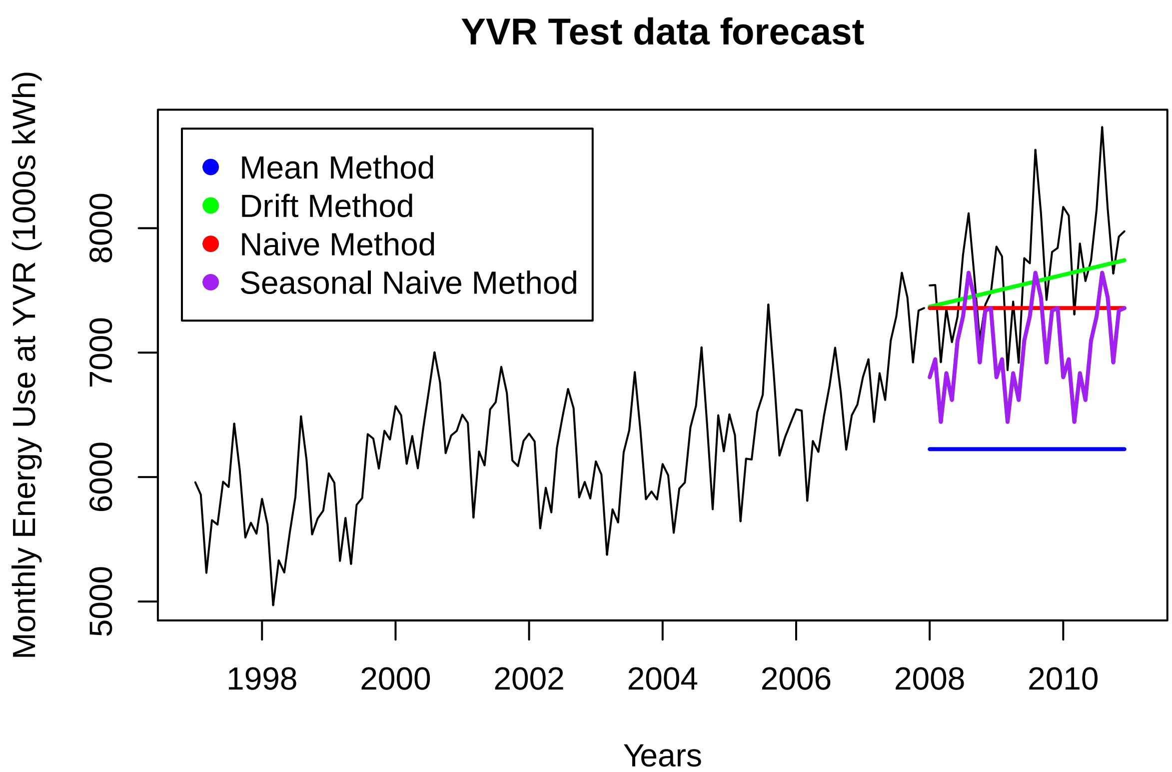


The calendar adjustment is to remove the calendar effect in our time series due to variation of number of days in each month. Therefore, by turning the monthly average energy use into daily average we expect to see the time series to be smoothened out. However, what we actually observe is that there seems to be more variations in our time series (graph above). Such means the calendar adjustment is ineffective.

```
yvr.train = window(yvr, end=c(2007, 12))
yvr.test = window(yvr, start=c(2008, 1))
```

```
# mean method (blue)
fit1.mean = meanf(yvr.train, h=36)
# drift method (green)
fit1.drift = rwf(yvr.train, drift = TRUE, h=36)
# naïve method (red)
fit1.naive = naive(yvr.train, h=36)
# seasonal naïve method (purple)
fit1.snaive = snaive(yvr.train, h=36)

# visualize the forecasts
plot(yvr.train, xlim=c(1997, 2011), ylim=c(5000, 8800), main='YVR Test data forecast',
     xlab='Years', ylab='Monthly Energy Use at YVR (1000s kWh)')
lines(yvr.test)
lines(fit1.mean$mean, col='blue', lwd=2)
lines(fit1.drift$mean, col='green', lwd=2)
lines(fit1.naive$mean, col='red', lwd=2)
lines(fit1.snaive$mean, col='purple', lwd=2)
legend(1996.8, 8800, legend=c('Mean Method', 'Drift Method', 'Naive Method', 'Seasonal Naive Method'),
```

```r
       col=c('blue', 'green', 'red', 'purple'), pch=19)
```




```r
# accuracy measures
accuracy(meanf(yvr.train, h=36), yvr.test)
```

```
##                         ME     RMSE       MAE        MPE      MAPE
## Training set -1.930644e-13  507.840  406.4286 -0.6657438  6.579549
## Test set      1.463795e+03 1528.498 1463.7955 18.7755758 18.775576
##                  MASE      ACF1 Theil's U
## Training set 1.533789 0.7140918        NA
## Test set     5.524104 0.4037248  3.138214
```

```r
accuracy(naive(yvr.train, h=36), yvr.test)
```

```
##                     ME     RMSE      MAE         MPE     MAPE     MASE
## Training set  10.68702 371.8135 310.2901 -0.02398732 5.053622 1.170980
## Test set     330.08333 550.0588 443.1944  3.98109394 5.603453 1.672537
##                    ACF1 Theil's U
## Training set -0.1845823        NA
## Test set      0.4037248   1.11998
```

```r
accuracy(rwf(yvr.train, h=36, drift = TRUE), yvr.test)
```

```
##                          ME     RMSE      MAE        MPE     MAPE     MASE
## Training set -2.638220e-13 371.6598 309.3927 -0.1967793 5.044069 1.167593
## Test set      1.323734e+02 415.9790 328.4824  1.4437756 4.213286 1.239634
##                    ACF1 Theil's U
## Training set -0.1845823        NA
## Test set      0.2664233 0.8509942
```

```
accuracy(snaive(yvr.train, h=36), yvr.test)
```

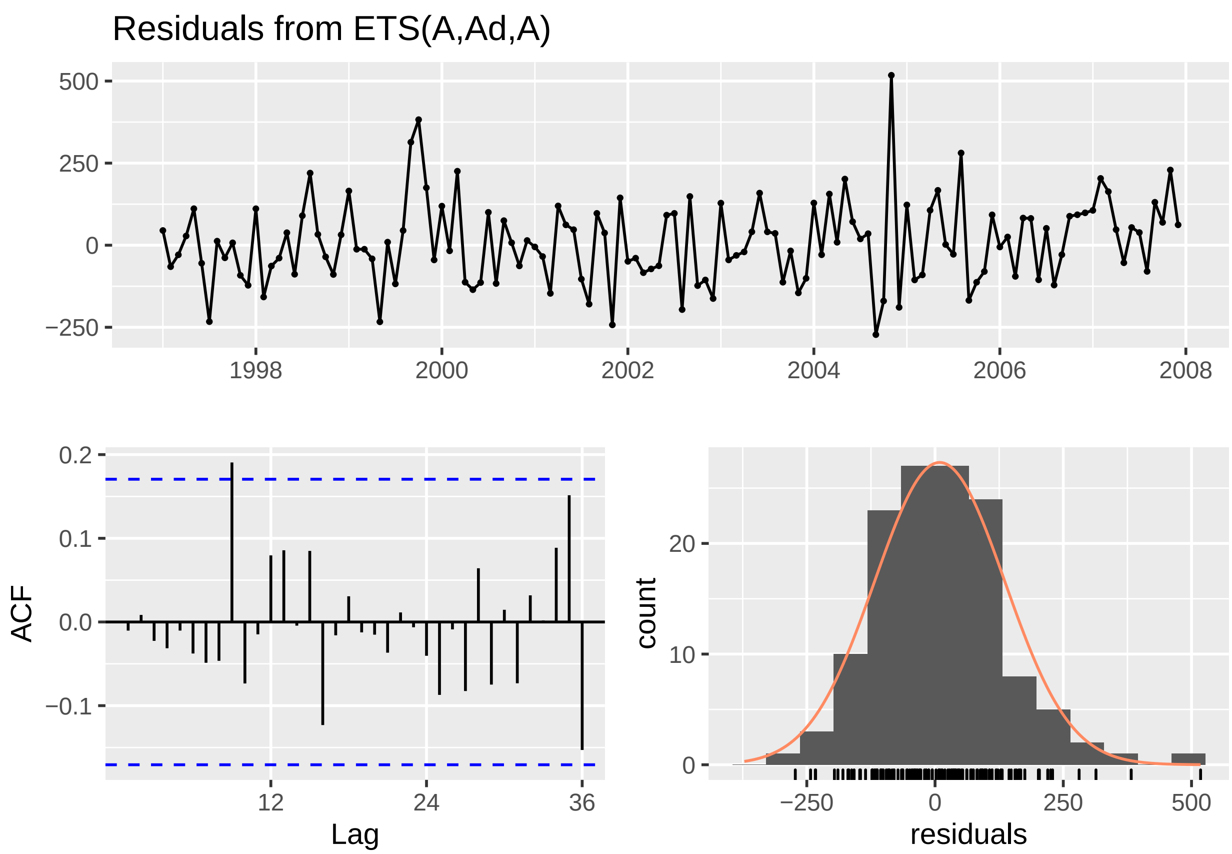
```
##                    ME     RMSE      MAE      MPE     MAPE     MASE
## Training set 127.9833 343.3462 264.9833 1.853620 4.164394 1.000000
## Test set     626.8333 698.6573 626.8333 8.038624 8.038624 2.365558
##                   ACF1 Theil's U
## Training set 0.8206604        NA
## Test set     0.4806562  1.424002
```

The drift method generalizes the best since has the lowest errors on test data, for all metrics.

MASE of drift method: on average, the test error, scaled on training set mean absolute error, is 1.2. It means, on average, the test error is 20 percent higher than the training error.

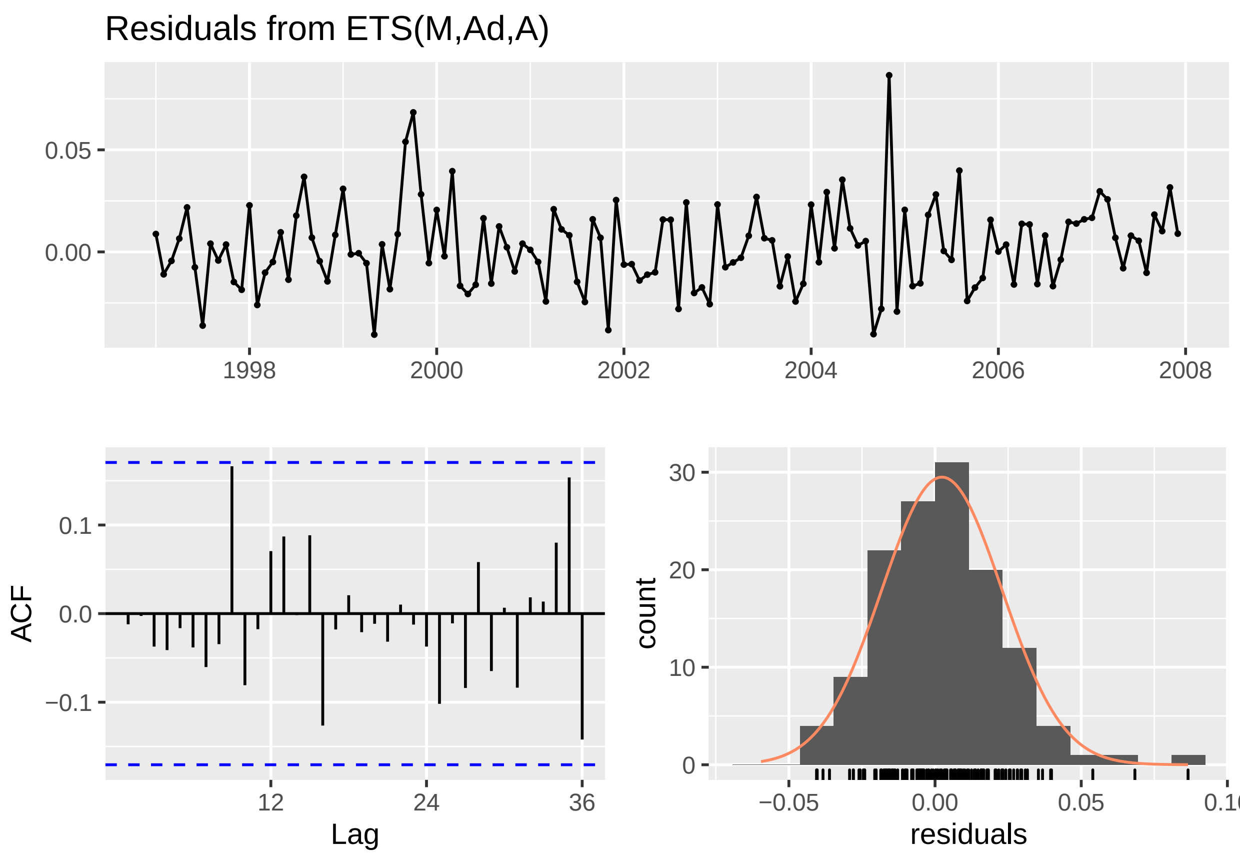
```
fit1.ets = ets(yvr.train, model = 'AAA', damped = TRUE)
fit2.ets = ets(yvr.train, model = 'MAA', damped = TRUE)
fit3.ets = ets(yvr.train, model = 'AAA', damped = NULL)
fit4.ets = ets(yvr.train, model = 'MAA', damped = NULL)


fit5.ets = ets(yvr.train, model = 'MAM', damped=TRUE)
fit6.ets = ets(yvr.train, model = 'MAM', damped=NULL)
fit7.ets = ets(yvr.train)
```

```
checkresiduals(fit1.ets)
```



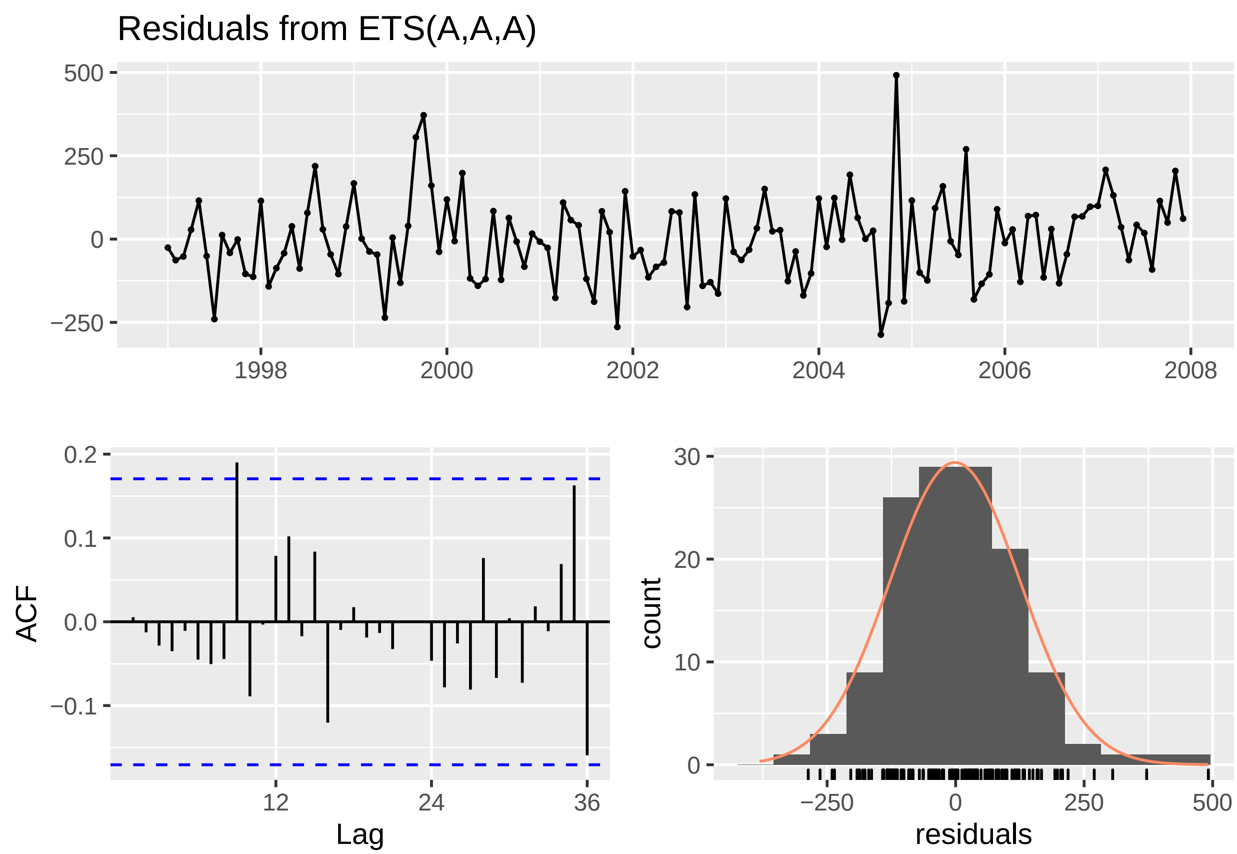

```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,Ad,A)
## Q* = 13.308, df = 7, p-value = 0.06495
##
## Model df: 17.   Total lags used: 24
```

```
checkresiduals(fit2.ets)
```



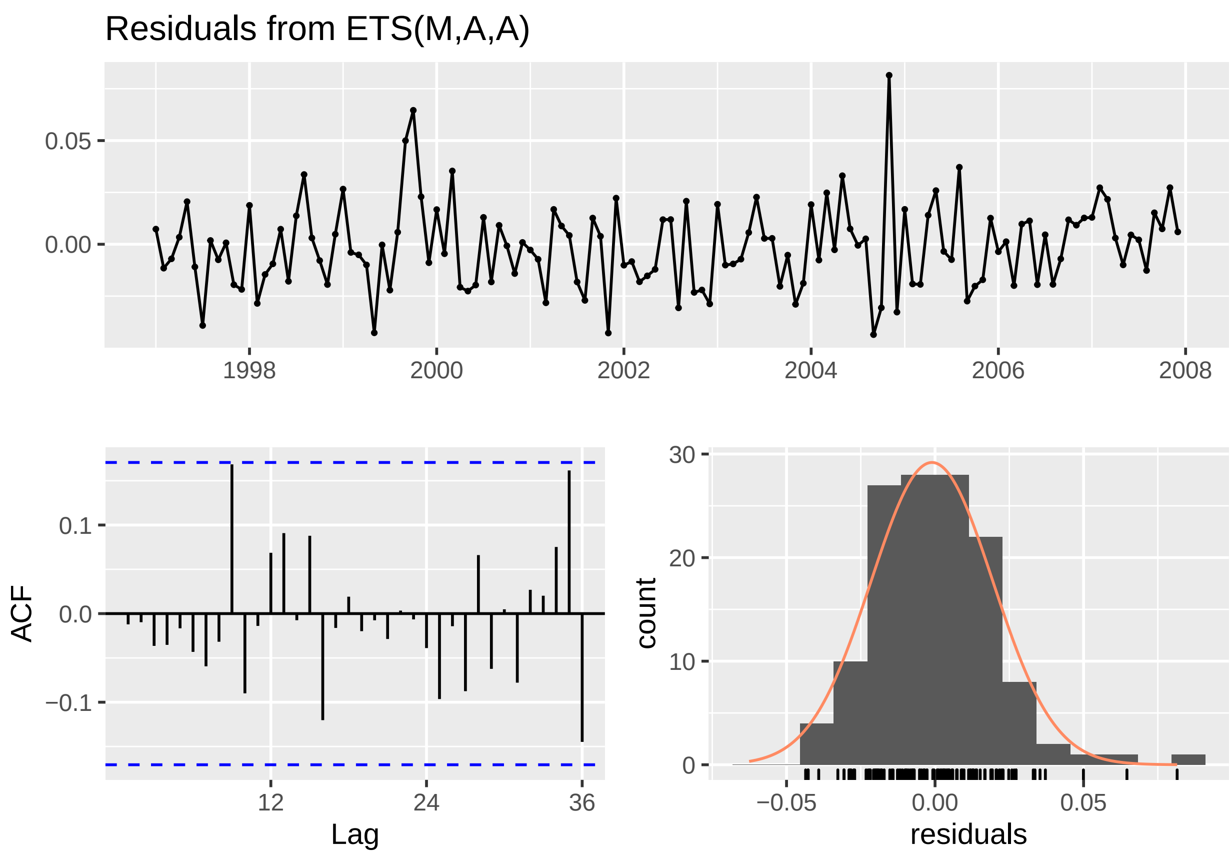

```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,Ad,A)
## Q* = 12.451, df = 7, p-value = 0.08667
##
## Model df: 17.   Total lags used: 24
```

```r
checkresiduals(fit3.ets)
```



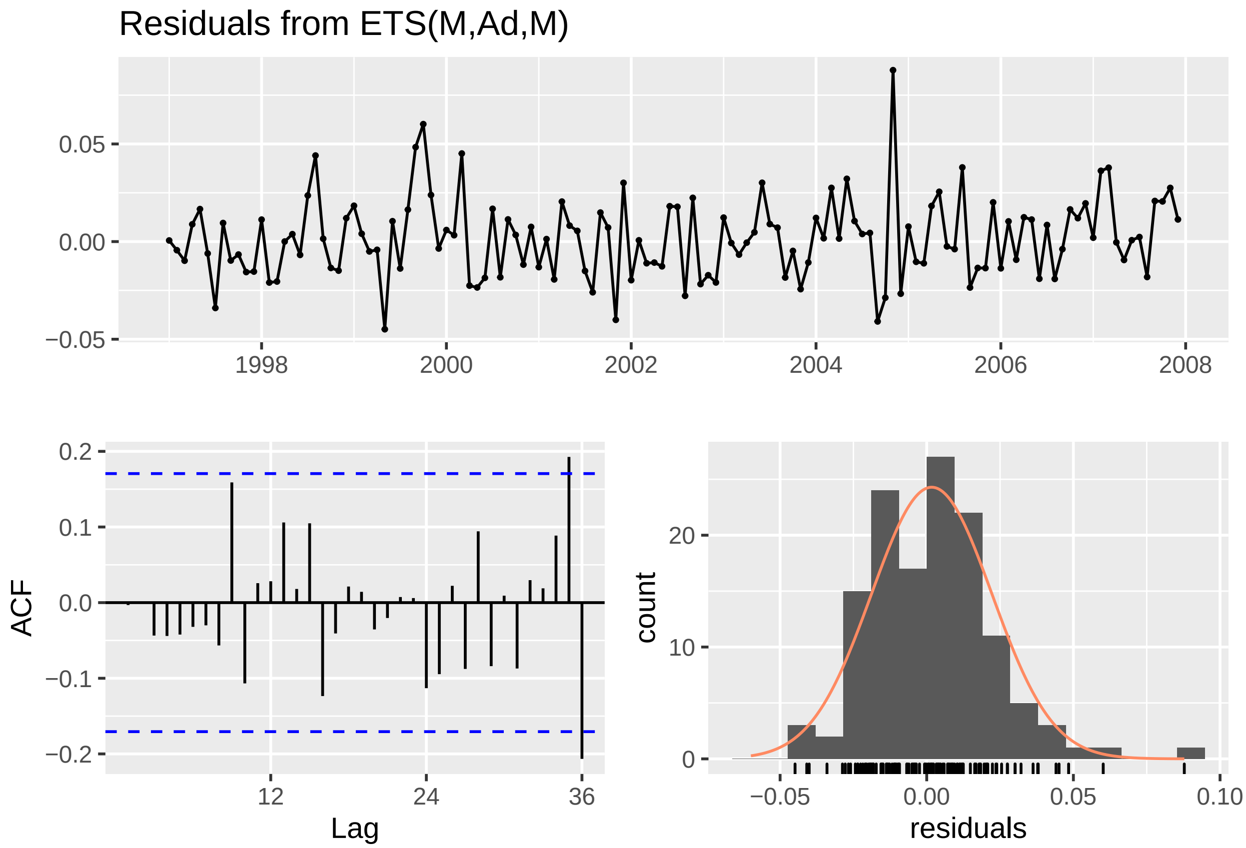

```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,A,A)
## Q* = 14.031, df = 8, p-value = 0.08095
##
## Model df: 16.   Total lags used: 24
```

```r
checkresiduals(fit4.ets)
```



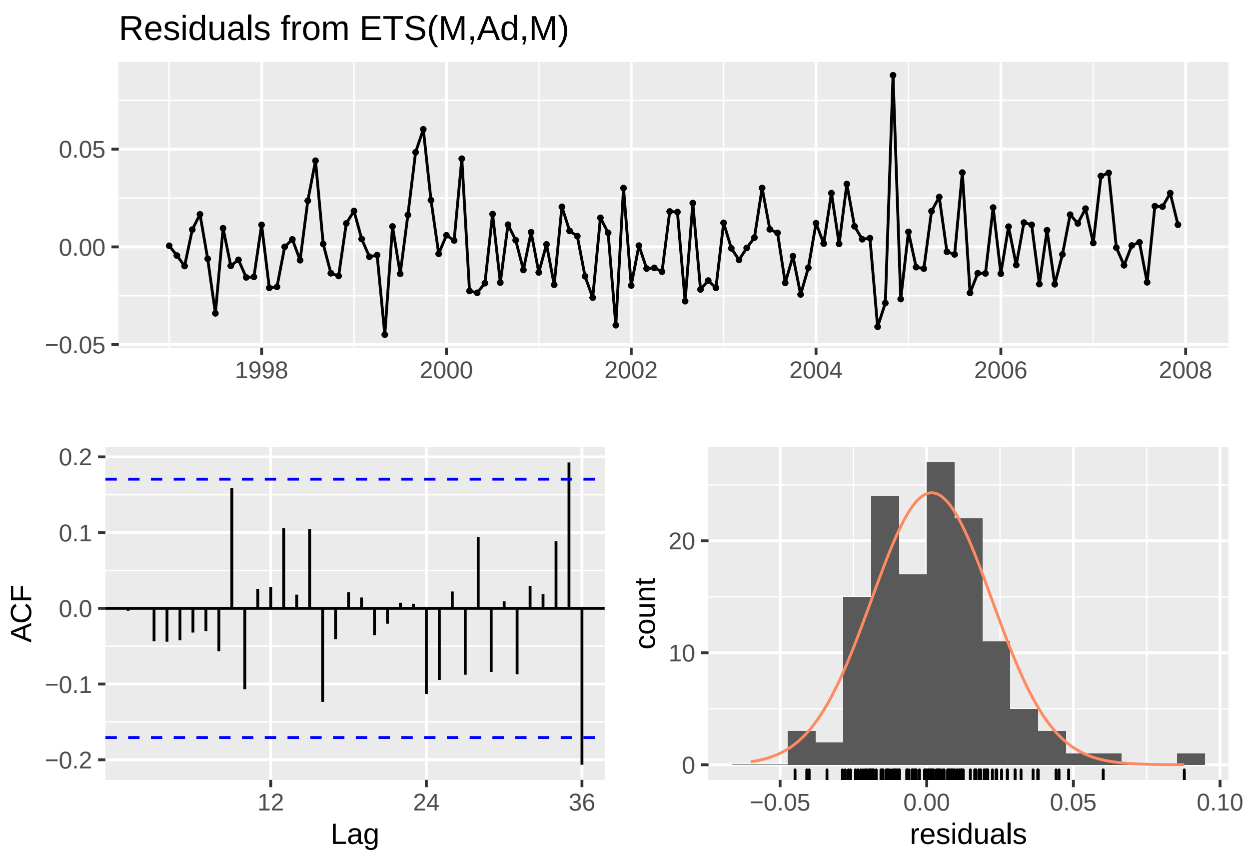

```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,A,A)
## Q* = 12.469, df = 8, p-value = 0.1315
##
## Model df: 16.   Total lags used: 24
```

```
checkresiduals(fit5.ets)
```



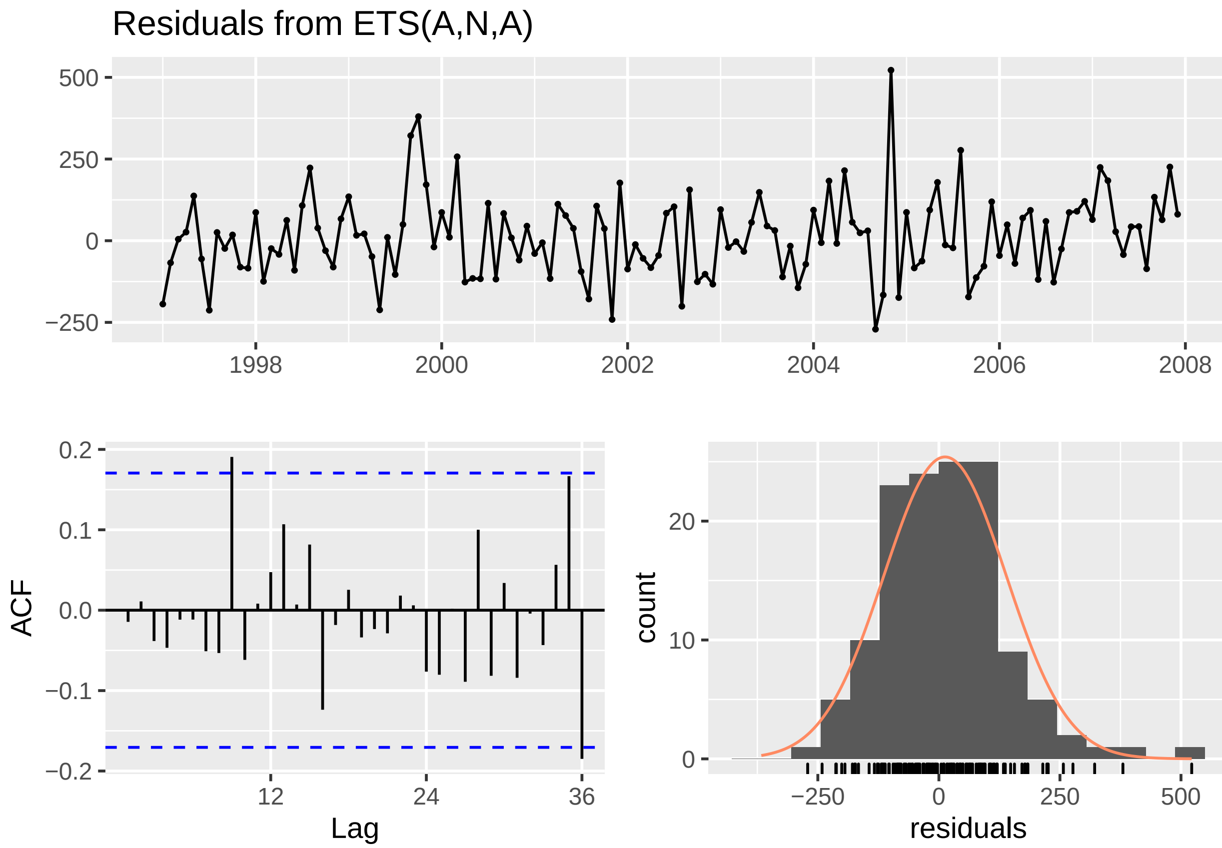

```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,Ad,M)
## Q* = 15.447, df = 7, p-value = 0.03068
##
## Model df: 17.   Total lags used: 24
```

```
checkresiduals(fit6.ets)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,Ad,M)
## Q* = 15.447, df = 7, p-value = 0.03068
##
## Model df: 17.   Total lags used: 24
```

```r
checkresiduals(fit7.ets)
```



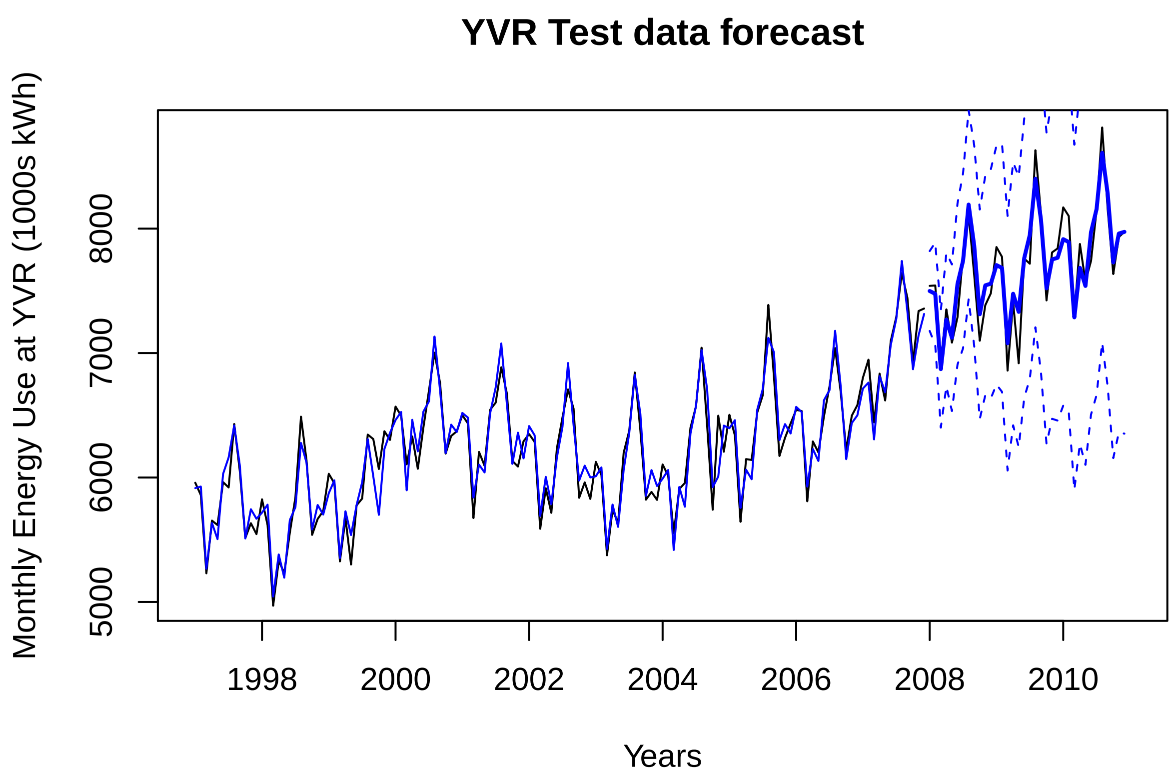

```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,N,A)
## Q* = 14.089, df = 10, p-value = 0.169
##
## Model df: 14.   Total lags used: 24
```

```
summary(fit4.ets)
```

```
## ETS(M,A,A)
##
## Call:
##  ets(y = yvr.train, model = "MAA", damped = NULL)
##
##   Smoothing parameters:
##     alpha = 0.8082
##     beta  = 1e-04
##     gamma = 1e-04
##
##   Initial states:
##     l = 5803.689
##     b = 17.436
##     s = -37.9185 -34.4259 -248.9098 320.0811 666.5463 231.972
##            63.7197 -351.3296 -188.4109 -568.7838 53.8096 93.6498
```

```
##
##   sigma:  0.0218
##
##      AIC     AICc      BIC
## 1957.628 1962.996 2006.636
##
## Training set error measures:
##                     ME     RMSE     MAE        MPE     MAPE      MASE
## Training set -6.611667 126.3696 99.4423 -0.1458104 1.602955 0.3752775
##                     ACF1
## Training set -0.02754951
```

```
# visualize the forecasts
plot(yvr.train, xlim=c(1997, 2011), ylim=c(5000, 8800), main='YVR Test data forecast',
     xlab='Years', ylab='Monthly Energy Use at YVR (1000s kWh)')
lines(yvr.test)
lines(fitted(fit4.ets), col='blue')
lines(forecast(fit4.ets, h=36)$mean, col='blue', lwd=2)
lines(forecast(fit4.ets, h=36)$upper[,2], col='blue', lwd=1, lty='dashed')
lines(forecast(fit4.ets, h=36)$lower[,2], col='blue', lwd=1, lty='dashed')
```

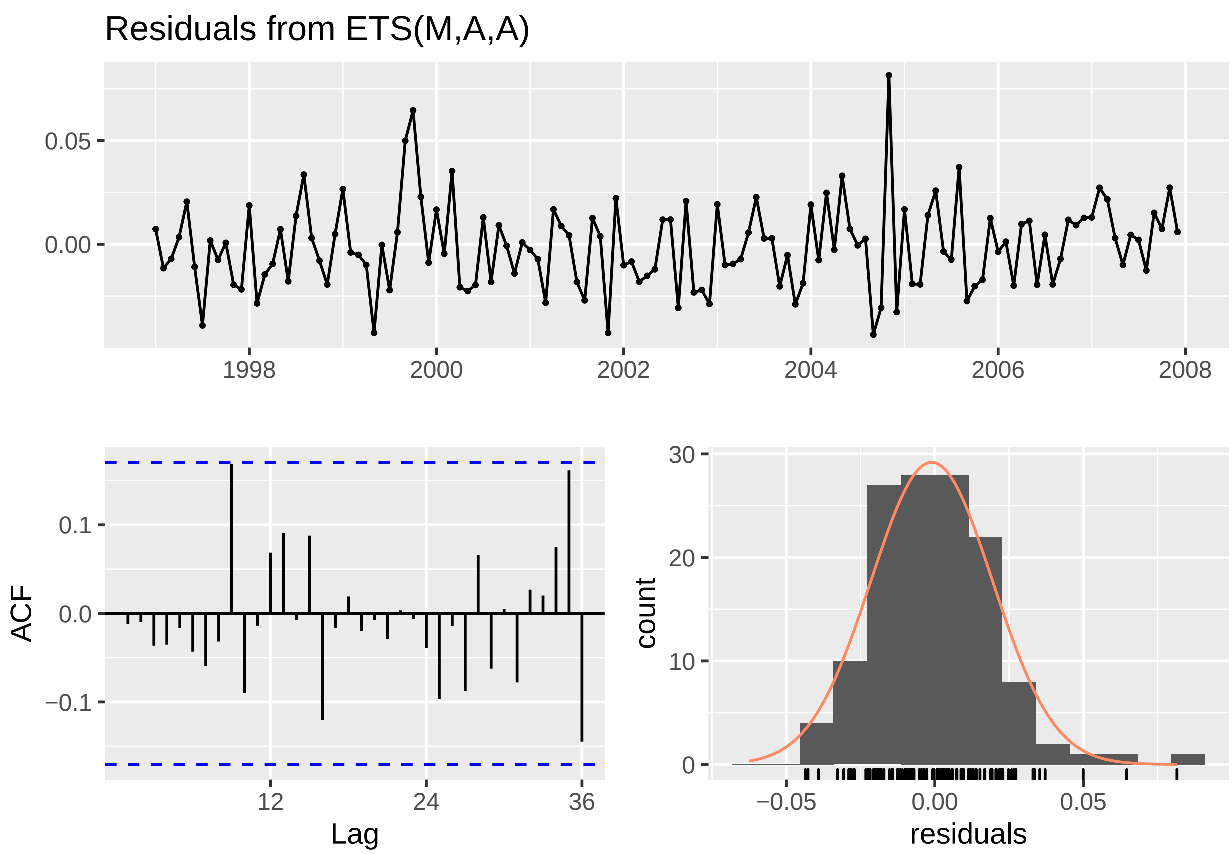



```
# accuracy scores
accuracy(forecast(fit4.ets, h=36), yvr.test)
```

```
##                     ME      RMSE       MAE        MPE      MAPE       MASE
```

```
## Training set  -6.611667 126.3696  99.4423 -0.1458104 1.602955 0.3752775
## Test set     -20.414073 155.9826 123.1789 -0.3335670 1.614019 0.4648554
##                    ACF1 Theil's U
## Training set -0.02754951        NA
## Test set      0.17004622 0.3285668
```

```
# residual diagnostics
checkresiduals(fit4.ets)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,A,A)
## Q* = 12.469, df = 8, p-value = 0.1315
##
## Model df: 16.   Total lags used: 24
```

```
# mean of residuals
mean(fit4.ets$residuals)
```

```
## [1] -0.001043112
```

```
# box ljung test of residual autocorrelations
Box.test(fit1.ets$residuals, type="Ljung", lag=24)
```

```
##
##  Box-Ljung test
##
## data:  fit1.ets$residuals
## X-squared = 13.308, df = 24, p-value = 0.9608
```

Additive Seasonality:

The seasonality seems quite constant across the time series, and did not amplify as the level increases.

Additive Trend without damping:

The trend does not look very exponential. Although the increasing energy use for the airport should become flatter (corresponding to damping), as the operation reaches limit (given the facility does not expand rapidly), we are uncertain should the capacity be reached in the next three years. Therefore we tried both damping and no damping. The version without damping has average residual closer to 0, so we chose no damping.

Multiplicative Errors:

There is no way to tell whether errors are additive or multiplicative from the original time series. Therefore, we tried both and Muliplicative Error has slightly less significant autocorrelations in the residuals.

The ETS model has forecasted the test data much more accurately than either of the basic methods, since all errors from ETS are much lower.

The MASE from the ETS model is less than 1, indicating the average test error is even lower than the training error.

The MAE for the ETS model shows that on average, the forecast on test data is about 123, 000 kilowatt hour off. 123 kwh per month is approximately 4 days of electricity consumption for a typical US family (refer to the bibliographical for the source). Such size of error for an airport can be negligible. Same can be inferred from RMSE.

In all, the ETS model of our choice did a good job on forecasting test data, which can transfer to good generalizability.

Zero MeanThe mean of time plot is approximately zero.

Constant VarianceOverall the variance is equal except for the 2 huge positive spike at around September 1999 just before year 2005. The spikes might be due to chance.

a) mean of residuals = -0.001043112

b)Histogram looks normal

The ACF plot looks perfect. We have no significant autocorrelations.

Box Ljung test for autocorrelations

H0: the first 24 autocorrelations are not significantly different from a white noise process

HA: the first 24 autocorrelations are significantly different from a white noise process

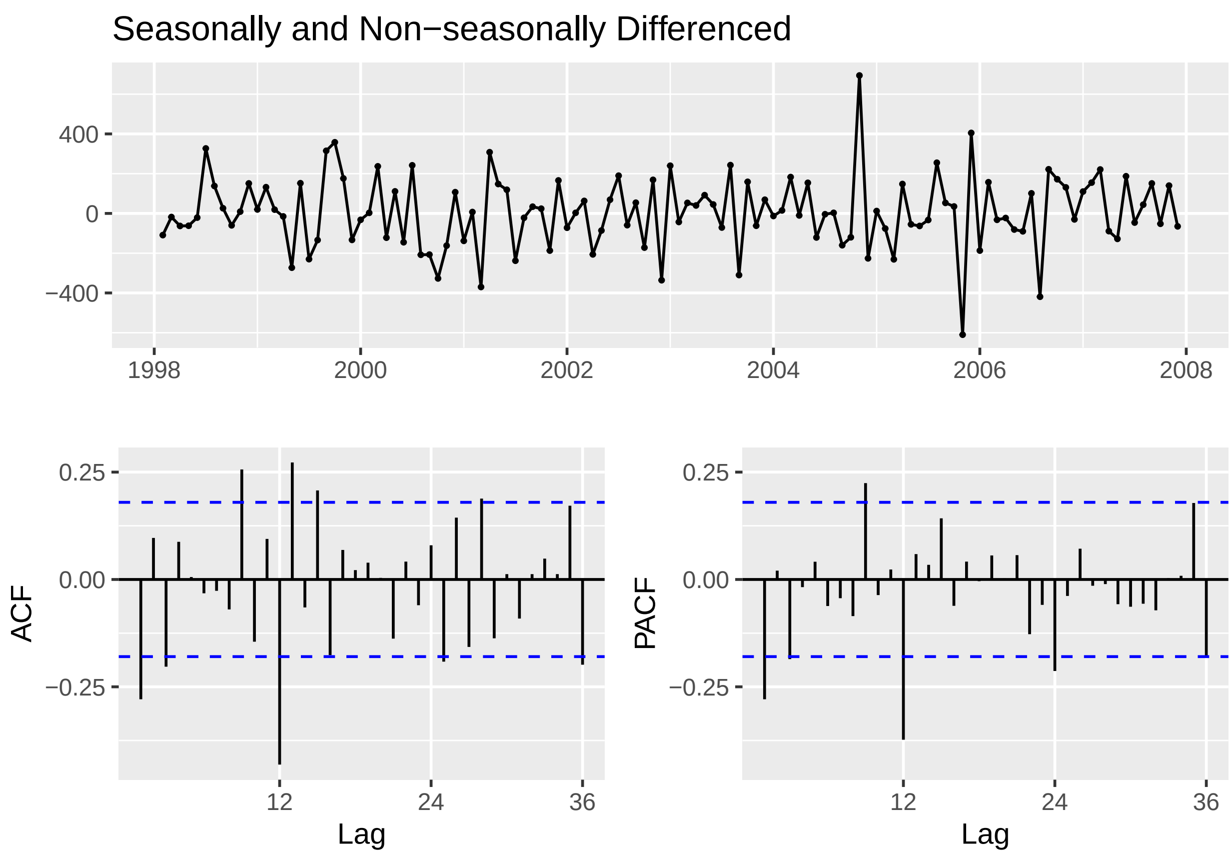
Test statistics = 13.308

p-value = 0.9608

Decision: We fail to reject null because p-value is bigger than 0.05

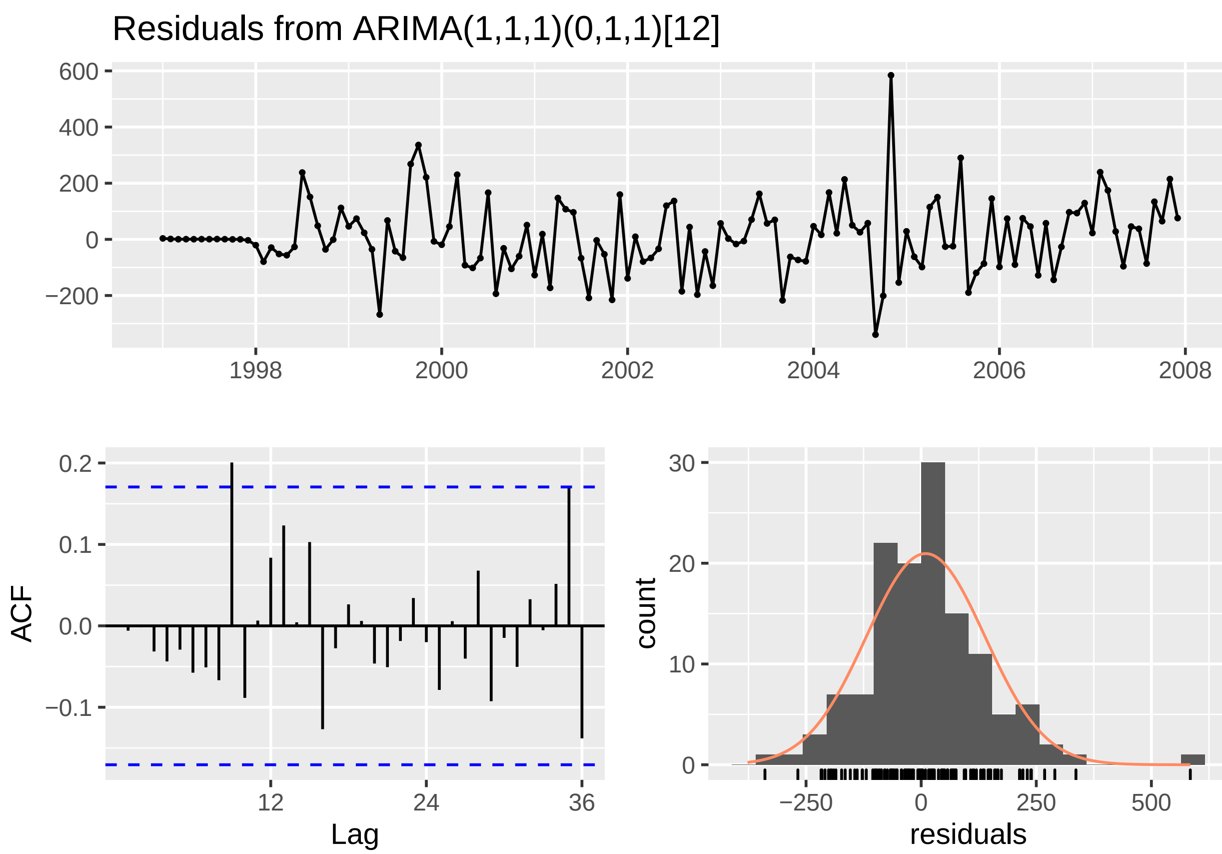
Conclusion: We conclude that the first 24 autocorrelations are not significantly different from a white noise process.

```r
# differencing
yvr.train %>%
  diff(lag=12) %>%
  diff() %>%
  ggtsdisplay(main='Seasonally and Non-seasonally Differenced')
```



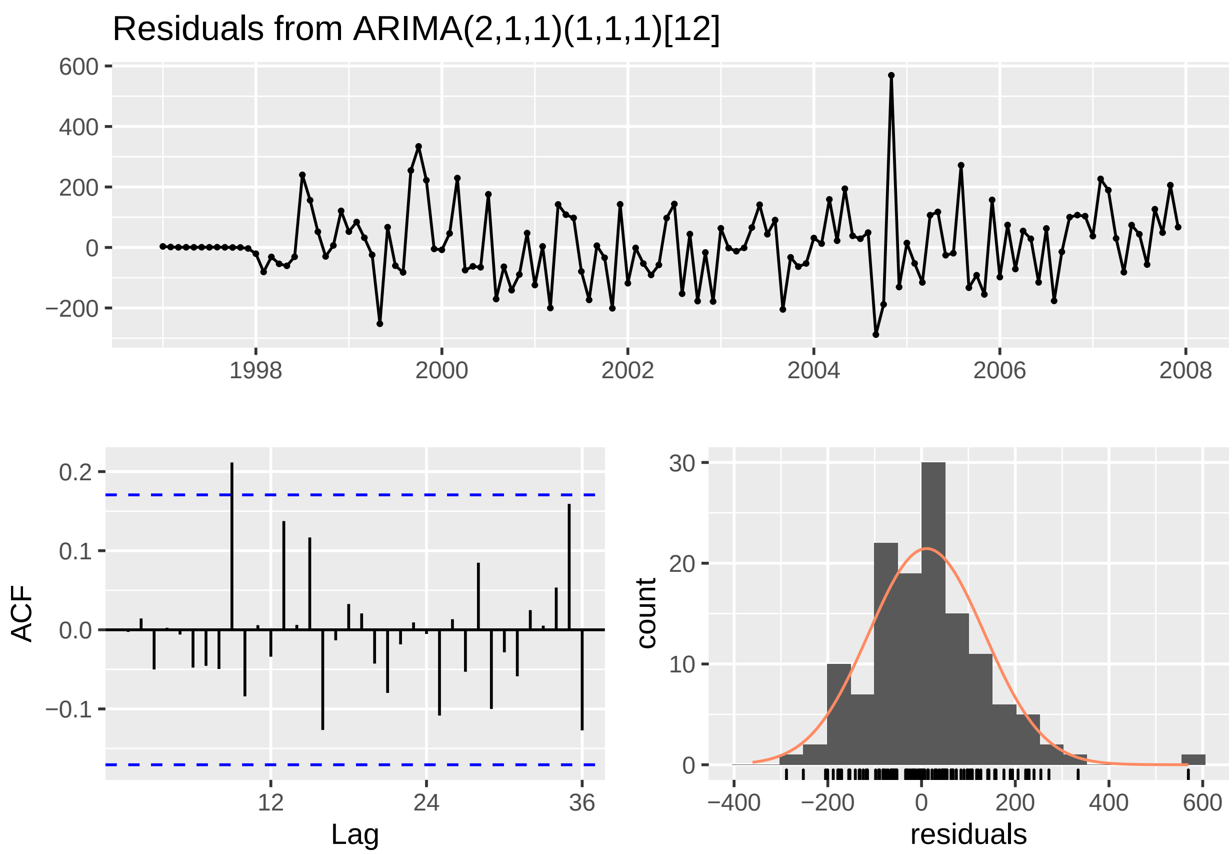

```r
fit1.arima = Arima(yvr.train, order = c(1,1, 1), seasonal = c(0,1,1), include.constant = TRUE)
fit2.arima = Arima(yvr.train, order = c(2,1, 1), seasonal = c(1,1,1), include.constant = TRUE)
fit3.arima = Arima(yvr.train, order = c(0,1, 1), seasonal = c(1,1,1), include.constant = TRUE)
fit4.arima = Arima(yvr.train, order = c(2,1, 1), seasonal = c(0,1,1), include.constant = TRUE)
fit5.arima = auto.arima(yvr.train)
fit6.arima = Arima(yvr.train, order = c(0, 1, 1), seasonal = c(0,1,1), include.constant = TRUE)
fit7.arima = Arima(yvr.train, order = c(2, 1, 2), seasonal = c(1,1,1), include.constant = TRUE)
fit8.arima = Arima(yvr.train, order = c(0, 1, 0), seasonal = c(0,1,1), include.constant = TRUE)
fit9.arima = Arima(yvr.train, order = c(1,1, 0), seasonal = c(0,1,1), include.constant = TRUE)
```

```r
checkresiduals(fit1.arima)
```



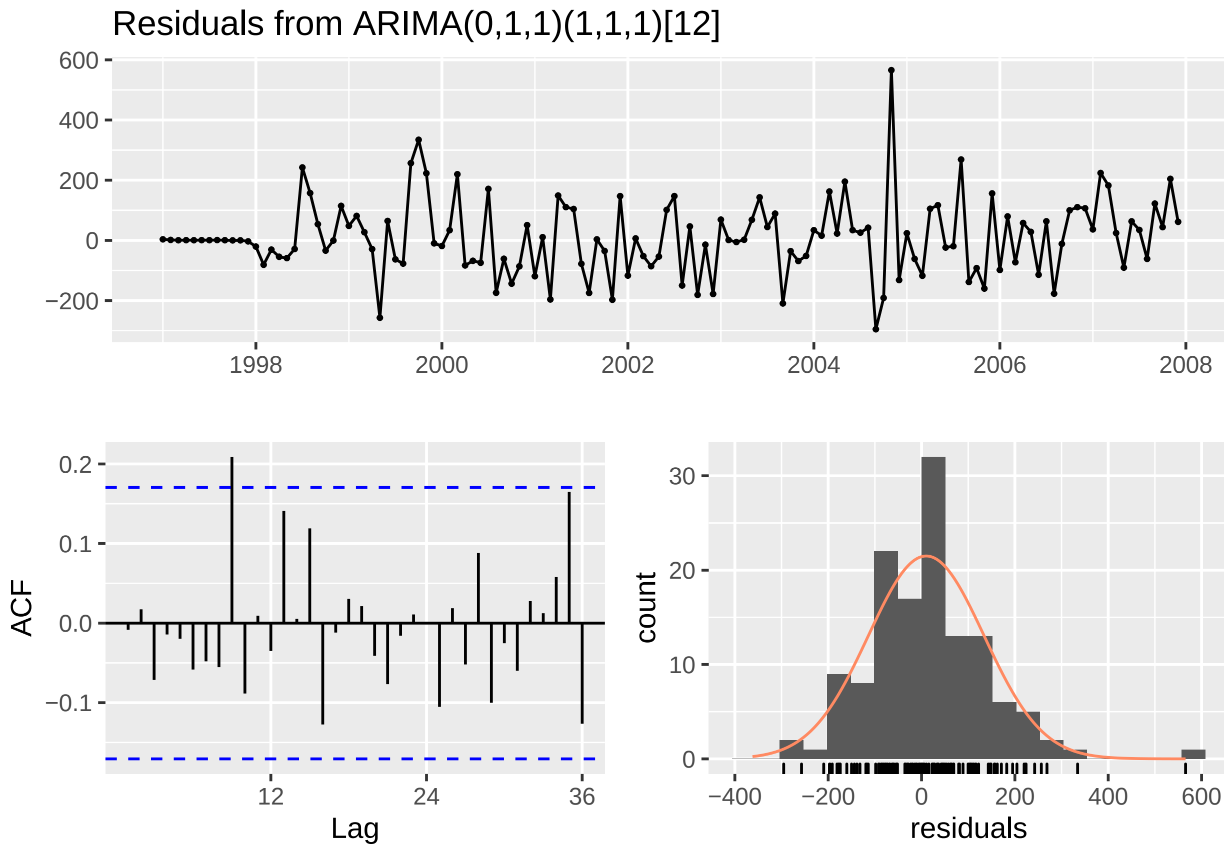

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,1)(0,1,1)[12]
## Q* = 17.561, df = 21, p-value = 0.6766
##
## Model df: 3.   Total lags used: 24
```

```r
checkresiduals(fit2.arima)
```



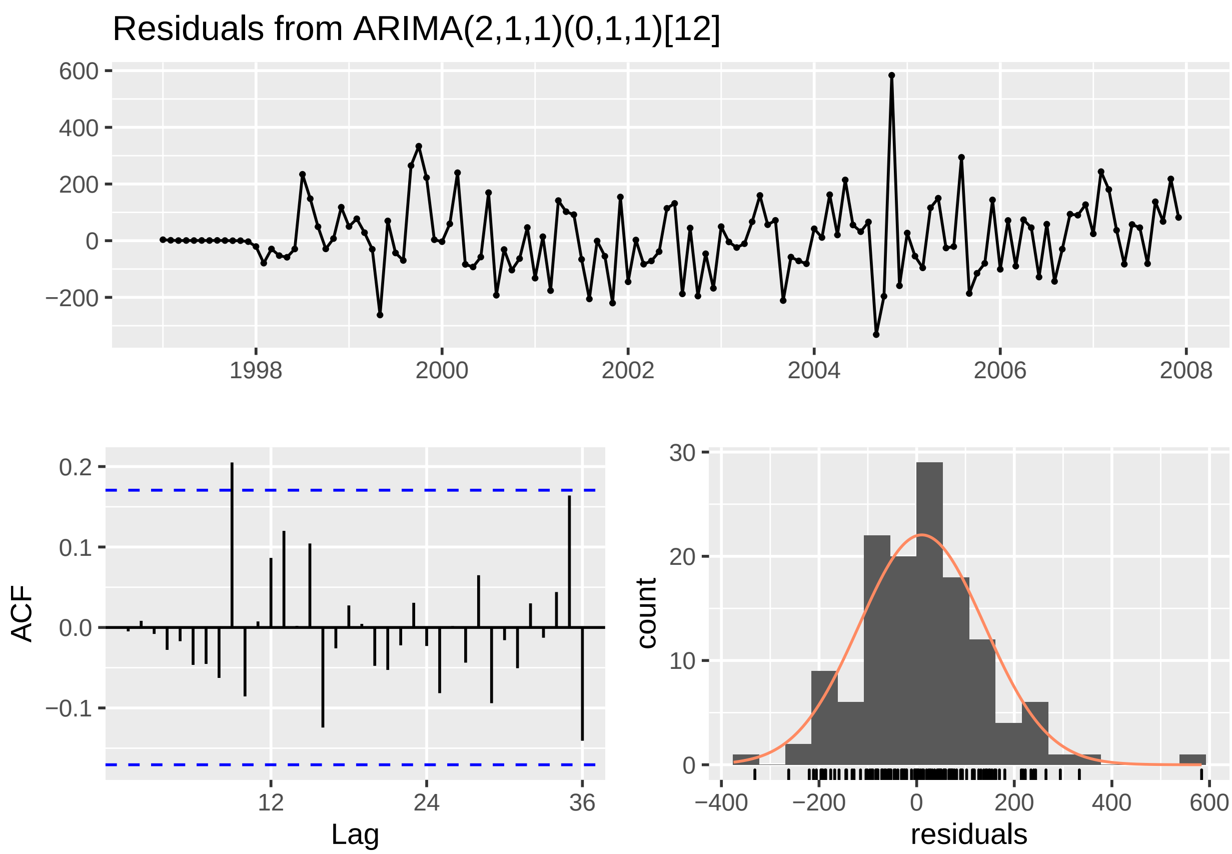

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,1,1)(1,1,1)[12]
## Q* = 17.932, df = 19, p-value = 0.527
##
## Model df: 5.   Total lags used: 24
```

```
checkresiduals(fit3.arima)
```



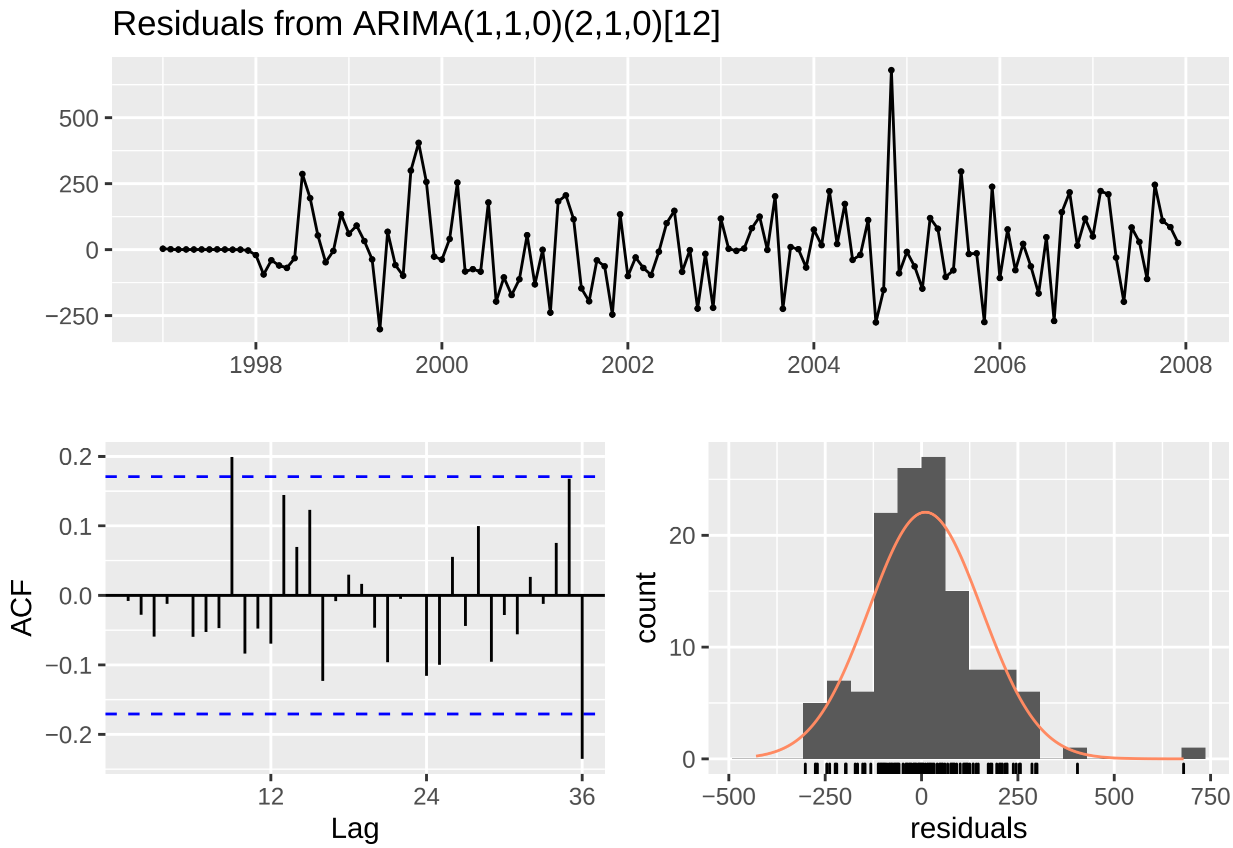

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,1)(1,1,1)[12]
## Q* = 18.752, df = 21, p-value = 0.601
##
## Model df: 3.   Total lags used: 24
```

```
checkresiduals(fit4.arima)
```



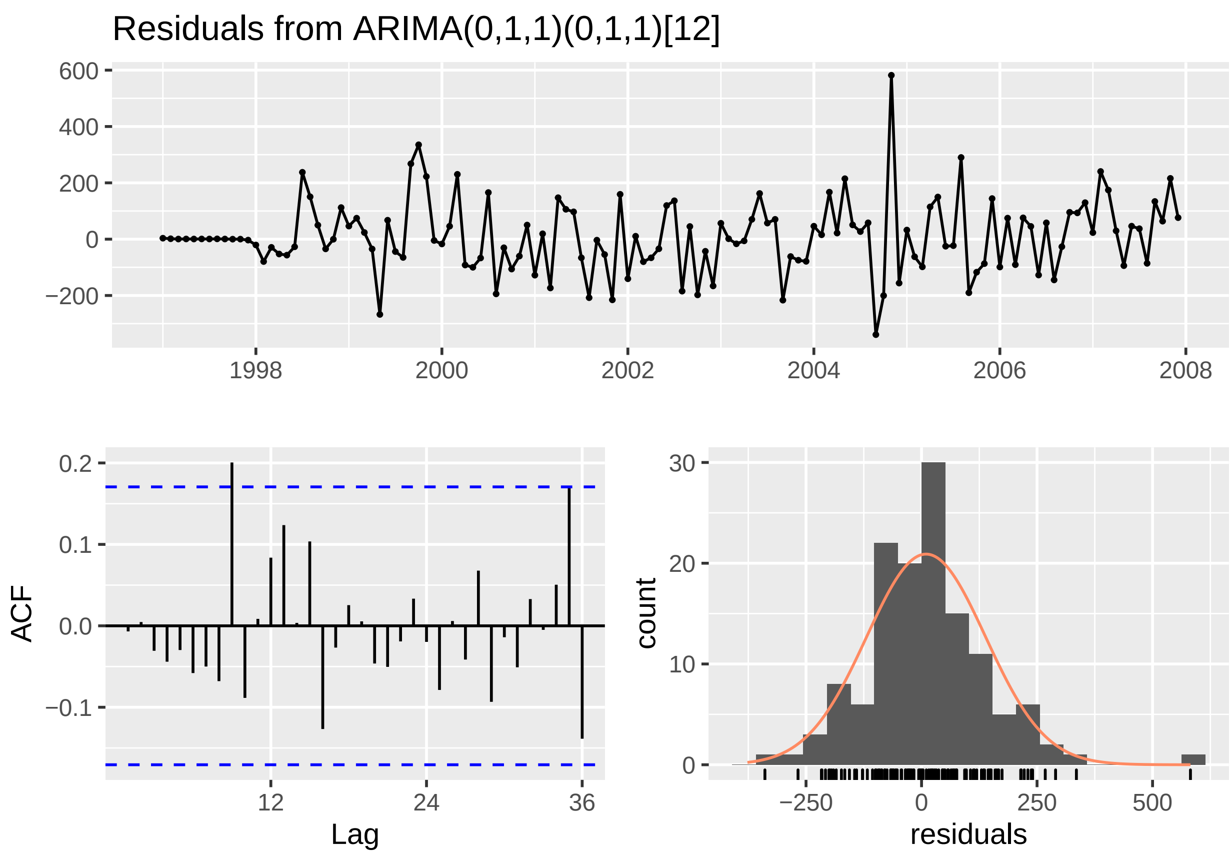

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,1,1)(0,1,1)[12]
## Q* = 17.033, df = 20, p-value = 0.6508
##
## Model df: 4.   Total lags used: 24
```

```
checkresiduals(fit5.arima)
```



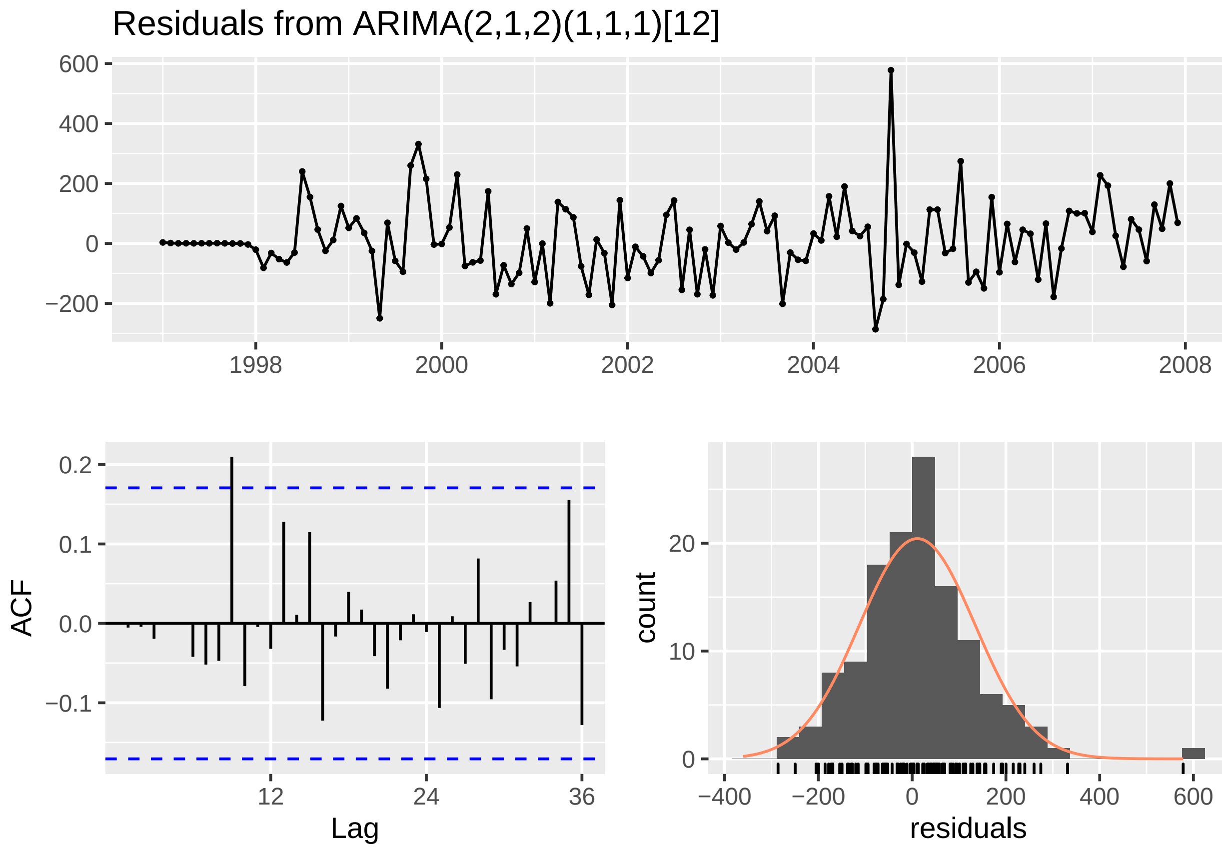

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,0)(2,1,0)[12]
## Q* = 22.208, df = 21, p-value = 0.3876
##
## Model df: 3.   Total lags used: 24
```

```
checkresiduals(fit6.arima)
```



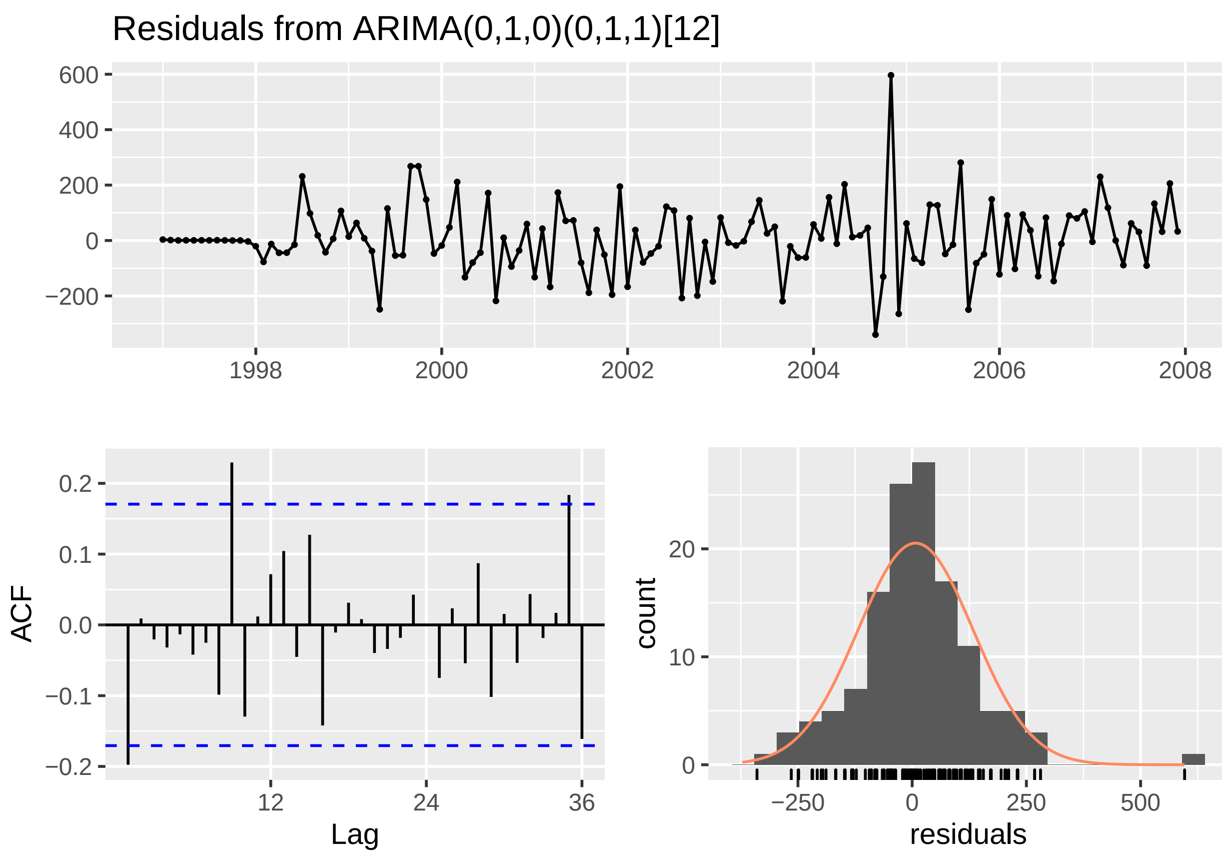

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,1)(0,1,1)[12]
## Q* = 17.582, df = 22, p-value = 0.7305
##
## Model df: 2.   Total lags used: 24
```

```
checkresiduals(fit7.arima)
```



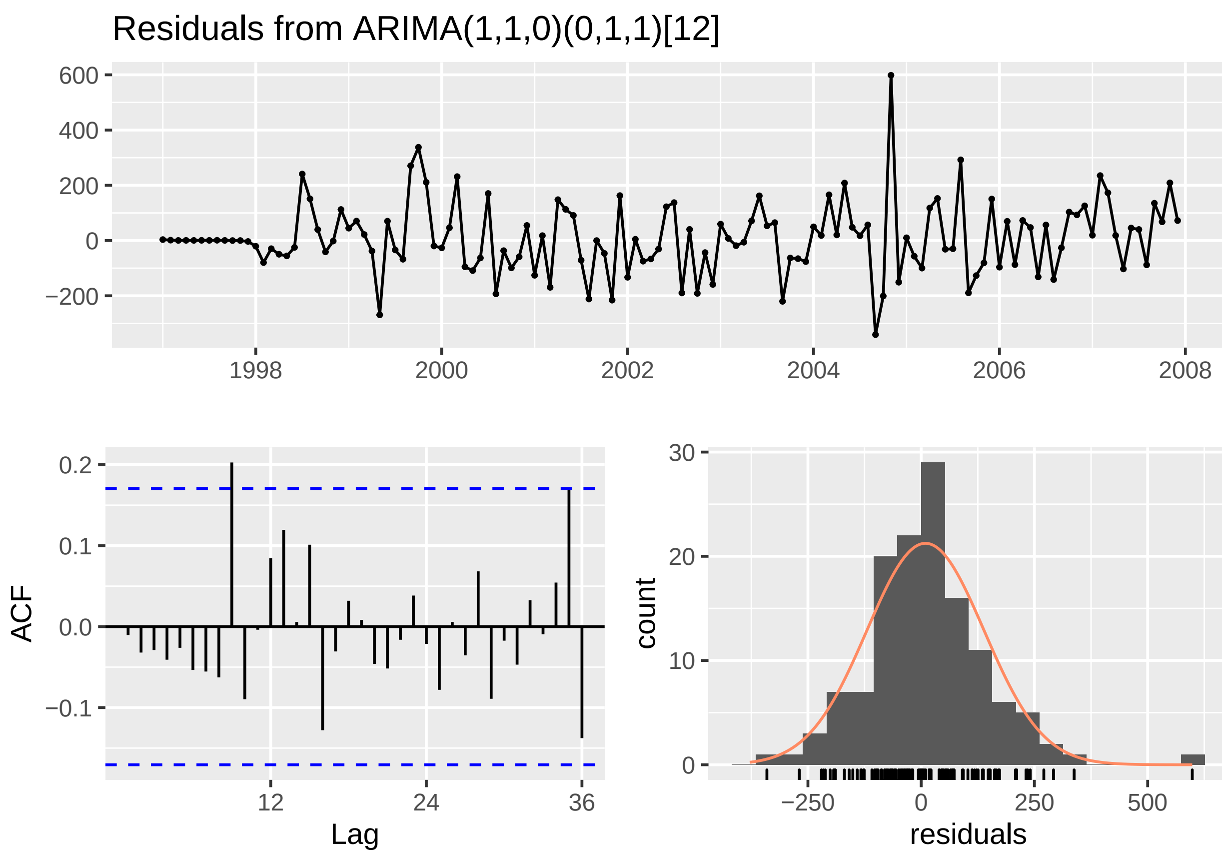

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,1,2)(1,1,1)[12]
## Q* = 16.895, df = 18, p-value = 0.5303
##
## Model df: 6.   Total lags used: 24
```

```r
checkresiduals(fit8.arima)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,0)(0,1,1)[12]
## Q* = 26.433, df = 23, p-value = 0.2808
##
## Model df: 1.   Total lags used: 24
```

```
checkresiduals(fit9.arima)
```



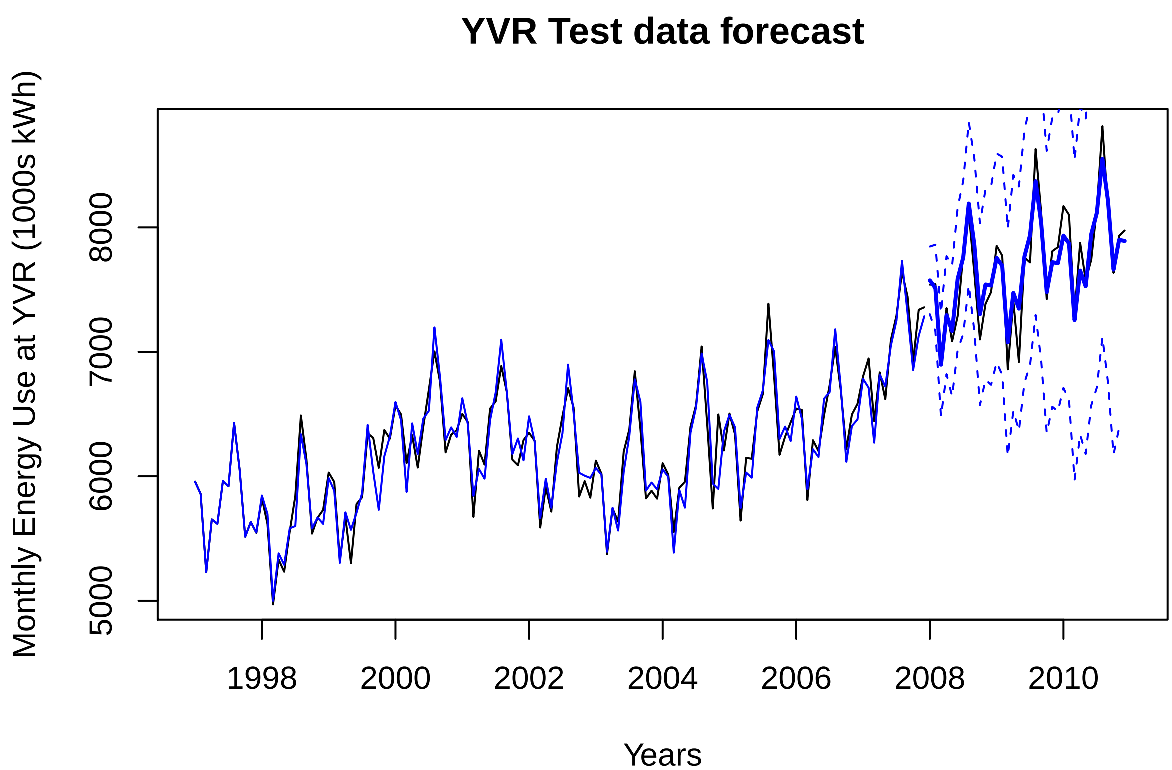

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,0)(0,1,1)[12]
## Q* = 17.722, df = 22, p-value = 0.7223
##
## Model df: 2.   Total lags used: 24
```

```
# check parameters
summary(fit9.arima)
```

```
## Series: yvr.train
## ARIMA(1,1,0)(0,1,1)[12]
##
## Coefficients:
##           ar1     sma1
##       -0.1972  -0.8927
## s.e.   0.0905   0.1794
##
## sigma^2 estimated as 18824:  log likelihood=-762.54
## AIC=1531.08   AICc=1531.29   BIC=1539.42
##
## Training set error measures:
##                    ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 9.495075 129.171 93.95688 0.1257726 1.494625 0.3545766
```

```
##                        ACF1
## Training set -0.01035696
```

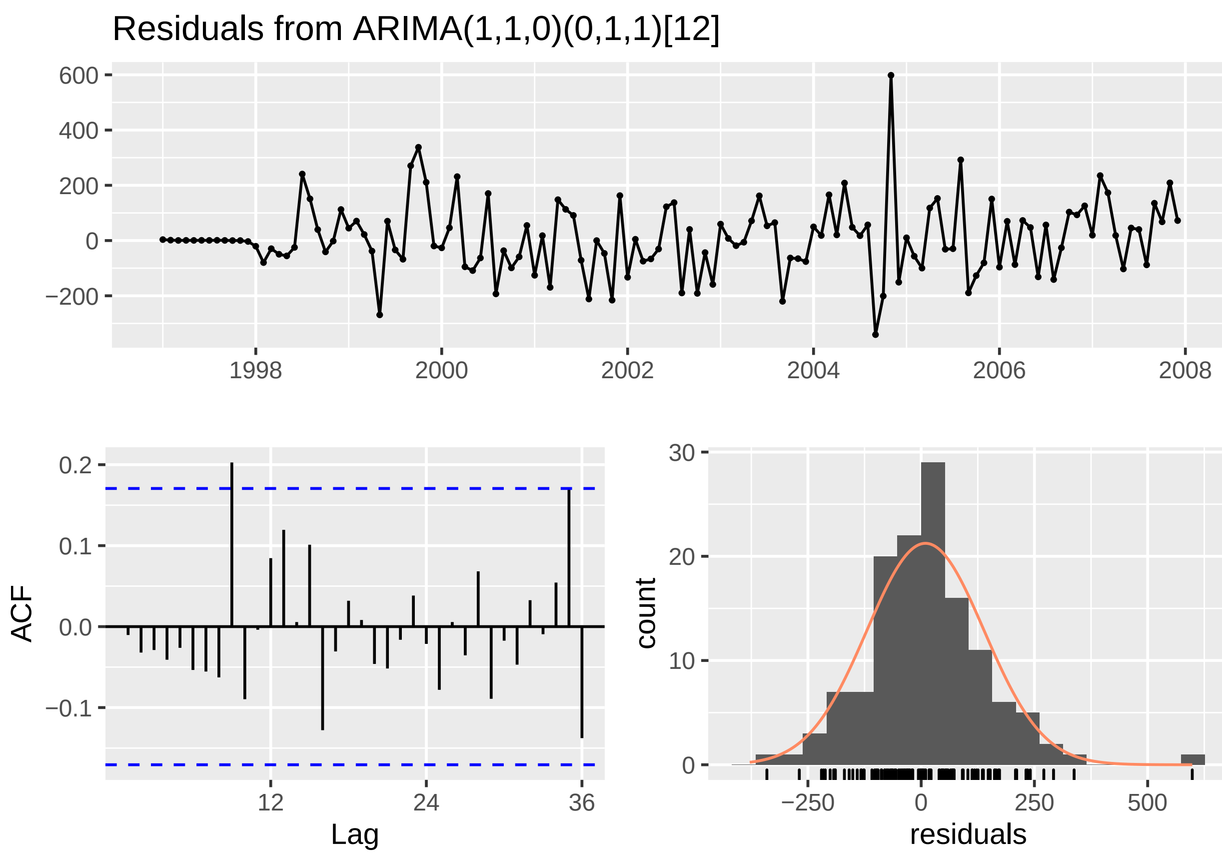
```r
# visualize the forecasts
plot(yvr.train, xlim=c(1997, 2011), ylim=c(5000, 8800), main='YVR Test data forecast',
     xlab='Years', ylab='Monthly Energy Use at YVR (1000s kWh)')
lines(yvr.test)
lines(fitted(fit9.arima), col='blue')
lines(forecast(fit9.arima, h=36)$mean, col='blue', lwd=2)
lines(forecast(fit9.arima, h=36)$upper[,2], col='blue', lwd=1, lty='dashed')
lines(forecast(fit9.arima, h=36)$lower[,2], col='blue', lwd=1, lty='dashed')
```




```r
# accuracy measures
accuracy(forecast(fit9.arima, h=36), yvr.test)
```

```
##                     ME     RMSE       MAE        MPE     MAPE      MASE
## Training set  9.495075 129.1710  93.95688  0.1257726 1.494625 0.3545766
## Test set     -9.243577 159.9366 125.09305 -0.2000004 1.633898 0.4720790
##                     ACF1 Theil's U
## Training set -0.01035696        NA
## Test set      0.22421431 0.3373746
```

```r
# residual diagnostics
checkresiduals(fit9.arima)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,0)(0,1,1)[12]
## Q* = 17.722, df = 22, p-value = 0.7223
##
## Model df: 2.   Total lags used: 24
```

```
# mean of residuals
mean(fit9.arima$residuals)
```

```
## [1] 9.495075
```

```
# box ljung test of residual autocorrelations
Box.test(fit9.arima$residuals, type="Ljung", lag=24)
```

```
##
##  Box-Ljung test
##
## data:  fit9.arima$residuals
## X-squared = 17.722, df = 24, p-value = 0.8163
```

SAR =1:

The PACF looks like it has exponential decay at lag 12, 24, and 36 (though for lag 36 it might not be significant). There is a significant lag 12 for ACF.

AR=1:

The PACF has a significant lag 1. Though ACF also has a significant lag 1, the over pattern is more complex. While suspecting it to be AR(1), I tried MA(1) as well. The residual plots of the two models are quite the same. Nevertheless, I still picked AR(1) as the ACF and PACF plots are more like it.

The ARIMA model has forecasted the test data much more accurately than either of the basic methods, since all errors from ARIMA are much lower. The MASE from the ARIMA model is less than 1, indicating the average test error is even lower than the training error. The MAE for the ARIMA model shows that on average, the forecast on test data is about 125, 000 kilowatt hour off. 125 kwh per month is approximately 4 days of electricity consumption for a typical US family (refer to the bibliographical for the source). Such size of error for an airport can be negligible. Same can be inferred from RMSE. In all, the ARIMA model of our choice did a good job on forecasting test data, which can transfer to good generalizability.

a) Mean of residuals = 9.495075. The slightly positive mean could be due to the positive spike of residual just before Year 2005.
b) Residuals are a little positively skewed, mainly due to the existence of high residuals (of about 620, which could be outliers). Apart from the slight skewness, the histogram looks normal overall.

Overall no significant Autocorrelations from the ACF plotThere is a significant spike at lag 9 which could be due to chance. Overall there is no significant autocorrelations.

Box Ljung test for autocorrelations

H0: the first 24 autocorrelations are not significantly different from a white noise process

HA: the first 24 autocorrelations are significantly different from a white noise process

Test statistics = 17.722

p-value = 0.8163

Decision: We fail to reject null because p-value is bigger than 0.05

Conclusion: We conclude that the first 24 autocorrelations are not significantly different from a white noise process.

Comparisons:

Goodness of fitT

he ETS model has better RMSE of training data, while the ARIMA model has better MAE of training data. Generally the goodness of fit does not say anything about forecasting ability. Therefore, here we are only checking training errors but not using them for decision making.

Generalizability

Overall the two models have very close test errors. The ETS model of our choice have slightly better test errors (for all metrics) than the ARIMA model, while such differences are negligible to an airport.

Residual Diagnostics

First, the average residuals from ETS is -0.001043112, which is closer to 0 than the 9.495075 from ARIMA. Second, the residual distribution of the ETS model seems less skewed than that of the ARIMA model. Third, the autocorrelations of the residuals from ETS are less significant than those from the ARIMA (even if the significance spike in ARIMA might be due to chance). In all the residuals from ETS are closer to white noise than ARIMA, meaning the ETS model has captured more information than the ARIMA does.

Model Complexity

The two models have hyperparameters simple enough that run time and computational requirements will not be issues. Therefore, the two models have approximately the same level of complexity.

Conclusion

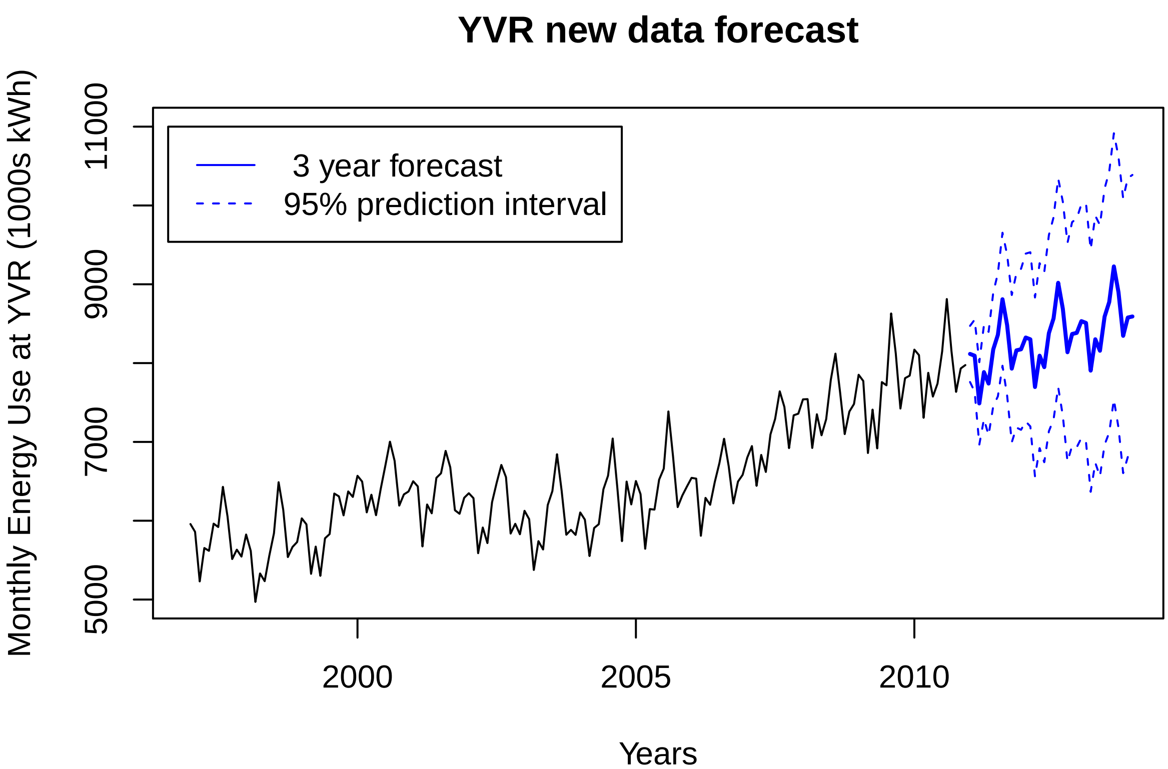
Overall I would choose the ETS model because it has lower test errors and more promising residuals.

```r
# point forecasts and prediction intervals
options(digits = 2)
forecast(ets(yvr, model=fit4.ets, use.initial.values = TRUE), h=36)
```

```
##          Point Forecast Lo 80 Hi 80 Lo 95 Hi 95
## Jan 2011           8116  7883  8349  7760  8472
## Feb 2011           8094  7795  8392  7636  8551
## Mar 2011           7488  7146  7830  6965  8011
## Apr 2011           7886  7496  8276  7290  8483
## May 2011           7740  7312  8169  7085  8396
## Jun 2011           8173  7702  8644  7453  8893
## Jul 2011           8358  7848  8869  7578  9139
## Aug 2011           8810  8259  9362  7967  9654
## Sep 2011           8481  7897  9066  7587  9375
## Oct 2011           7930  7319  8540  6995  8864
## Nov 2011           8161  7521  8802  7182  9141
## Dec 2011           8175  7507  8843  7153  9197
## Jan 2012           8324  7628  9021  7260  9389
## Feb 2012           8302  7579  9024  7197  9406
## Mar 2012           7696  6954  8439  6561  8832
## Apr 2012           8094  7327  8861  6921  9268
## May 2012           7949  7160  8737  6742  9155
## Jun 2012           8381  7567  9195  7136  9626
## Jul 2012           8567  7728  9405  7284  9849
## Aug 2012           9019  8153  9884  7695 10343
## Sep 2012           8689  7801  9578  7331 10048
## Oct 2012           8138  7231  9045  6751  9525
## Nov 2012           8370  7441  9298  6950  9789
## Dec 2012           8383  7435  9332  6933  9834
## Jan 2013           8533  7563  9502  7050 10016
## Feb 2013           8510  7520  9500  6996 10024
## Mar 2013           7905  6899  8910  6367  9442
## Apr 2013           8302  7277  9327  6735  9870
## May 2013           8157  7115  9199  6563  9751
## Jun 2013           8589  7527  9651  6964 10214
## Jul 2013           8775  7692  9857  7119 10430
## Aug 2013           9227  8122 10331  7537 10916
## Sep 2013           8897  7774 10021  7180 10615
## Oct 2013           8346  7207  9485  6604 10088
## Nov 2013           8578  7421  9735  6808 10347
## Dec 2013           8592  7417  9766  6795 10388
```

```r
plot(yvr,
     xlim=c(1997, 2013.8),
     ylim=c(5000, 11000),
     main='YVR new data forecast',
     xlab='Years', ylab='Monthly Energy Use at YVR (1000s kWh)')
```

```
lines(forecast(ets(yvr, model=fit4.ets, use.initial.values = TRUE), h=36)$mean, col='blue', lwd=2)
lines(forecast(ets(yvr, model=fit4.ets, use.initial.values = TRUE), h=36)$upper[,2], col='blue', lwd=1,
lines(forecast(ets(yvr, model=fit4.ets, use.initial.values = TRUE), h=36)$lower[,2], col='blue', lwd=1,
legend(1996.6, 11000, legend = c(' 3 year forecast', '95% prediction interval'), lty = c('solid', 'dash
```



Model Limitations

Limitation 1: Data LeakingThe first limitation regards to the model building process rather than the model itself. Back from the beginning of the project, we visualized the time plot of the entire data. Therefore we have already had a basic idea of the trend of the test data (upward), which would lead us to choose model with additive or multiplicative trend. Additionally, we did model selection based on test set, which overfits the test data. Such would increase the variance and therefore increases generalization error. (Our good performance on test data would not imply good performance on unseen future data)

Recommendation:Split data into training set, validation set, and test set. Use validation set for model selection to avoid overfitting test set.

Limitation 2: Only using past observations of the same variableThe caveat of the time series models is that they only use past observations of the same variable. Anything outside of seasonality, trend, and cycle cannot be explained by time series models.

Recommendation:Use exploratory models (details will be explained in the next question)

a) Idea 1:

Exploratory Model (Focus on Predictability)We can use all other variables available at hand (i.e. total area, passenger…) to predict month energy use. Since we focus on predictability, we should select high-performing black box models (eg: XGBoost, Random Forest..). Feature selections should be done to exclude unimportant features. After that, to actually make forecast, we first build separate time series models to forecast the important explanatory variables you chose. Then we predict monthly energy use based on the forecasted explanatory variables. Such approach would explain patterns that are caused other than time

b) Idea 2: Exploratory Model (Focus on Interpretability)Idea 2 has similar process to that in Idea 1. Should we focus on Interpretability of the model, we can only choose highly interpretable models (such as linear regression and decision trees). In the end we should be able to interpret coefficients on the linear regression or evaluate each split in a decision tree.

c) Idea 3: Try ETS Models with multiplicative trend and additive seasonalitySince ETS model forbids us to use combinations with additive seasonality and multiplicative trend, we need to first do seasonal adjustment so that the data has no