Forecasting the Canadian Retail Sales

In this document we develop a simple forecasting model of Canadian Retail Sales. The data describes unadjusted and seasonally adjusted Monthly Retail Trades in Canada from **January 2000** to **July 2015**.

Definition: Retail trade is defined in the International Standard Industrial Classification (ISIC) as the re-sale (sale without transformation) of new and used goods to the general public, for personal or household consumption or utilisation.

By researching the Statistics Canada help page, we note that that seasonally adjusted data is built using the X-12-ARIMA method.

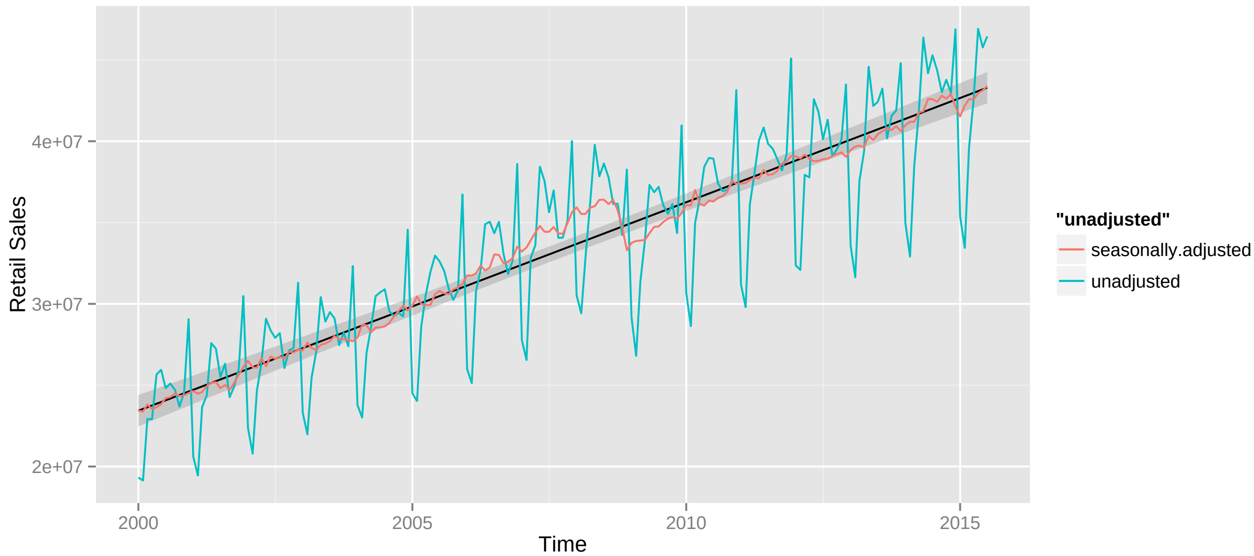
Formatting the data:

First the data provided is not ideal for an analysis using R so we need to reformat it.

```
base.data <- read.csv('./data/cansim__Jan-2000_Jul-2015.csv', header=FALSE)

data <-
  base.data %>%
  select(-V1, -V2, -V3) %>% # Remove the description columns
  slice(5:7) %>%            # Only select row containing data
  t %>% data.frame %>%      # Transpose and cast to data.frame
  rename(adjustments=X1) %>%
  rename(unadjusted=X2) %>%
  rename(seasonally.adjusted=X3) %>%
  mutate(seasonally.adjusted=as.numeric(as.character(seasonally.adjusted))) %>%
  mutate(unadjusted=as.numeric(as.character(unadjusted))) %>%
  mutate(adjustments=as.Date(
                      as.character(x=paste("01-", adjustments, sep="")),
                      format="%d-%b-%Y")
       )# Converts the adjustments. You need to paste a day to be able to convert such format to a da
```

First look at the data:

```
g.data.with.regression <-
  ggplot(data, aes(x=adjustments, y=unadjusted, color='unadjusted')) +
  geom_smooth(method='lm', color='black') + geom_line() +
  geom_line(data=data, aes(x=adjustments, y=seasonally.adjusted, color='seasonally.adjusted')) +
  xlab('Time') + ylab('Retail Sales')

g.data.with.regression
```



From here, we can note a few observations:

- The retail sales are strongly seasonal.
- The time plot shows a sudden change, particulary in 2009.
- The variability of the seasonal components seems constant.

We want to build a model that can forecast unadjusted retail sales data, let's create timeseries objects for that data.

```
unadjusted.ts <- ts(data$unadjusted, frequency=12)
```

The unadjusted retail sales data seem to be a good candidate for an ARIMA model

ARIMA models have three parts, the autoregression part (AR), the integration part (I) and the moving average part (MA). The main assumption behind the AR part is that observed values depend on some linear combination of previous values up to a maximum lag in conjonction with random error term. The MA part is based on the fact that the observed value is a random error term plus some linear combination of previous random error terms up to a maximum lag.

In order to analyse a time series to estimate an ARIMA model, the common procedure requires the timeseries to be stationary.

Stationarity:

It is clear from the observations made in the previous section that the Retail Sales data are not stationary.

But let's test that assumption using the **Augmented Dickey-Fuller Test**, the **Box test** and the **KPSS test**.

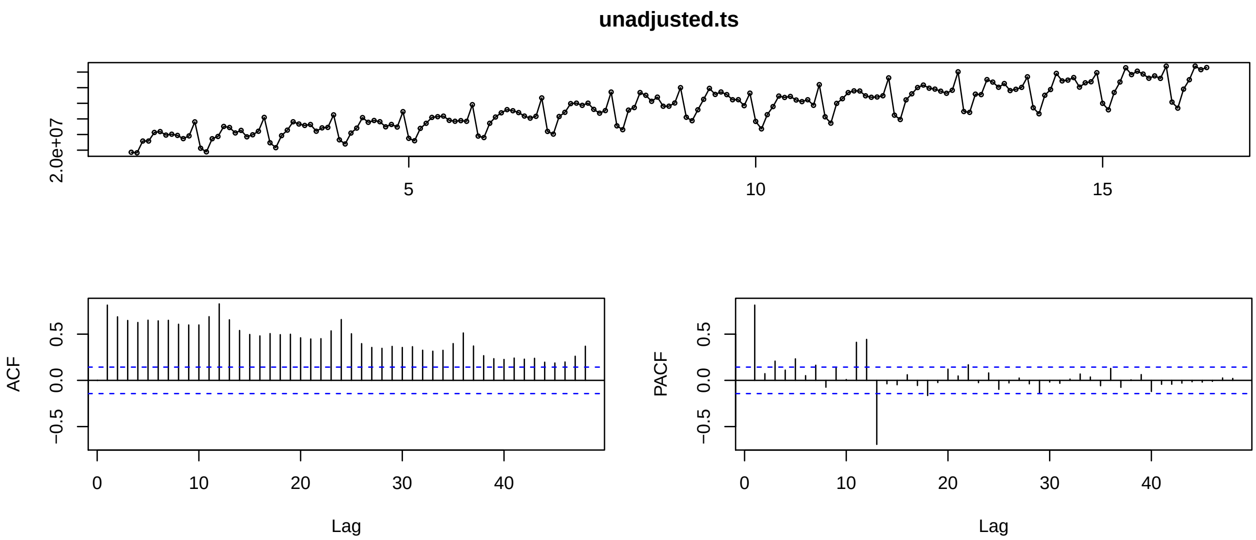
```
results.unadjusted <- list(
  adf=adf.test(unadjusted.ts)$p.value < 0.05,
  box=Box.test(unadjusted.ts, lag=12, type='Ljung-Box')$p.value < 0.05,
  kpss=kpss.test(unadjusted.ts)$p.value > 0.05
)
results.unadjusted
```

```
## $adf
## [1] TRUE
##
## $box
## [1] TRUE
##
## $kpss
## [1] FALSE
```

From those test, we note that the three tests don't give the same results of stationarity.

If we have a look at the Autocorrelation function (ACF) and the Partial autocorrelation function (PACF) of our data. The ACF shows large autocorrelations that diminish very slowly at large lags. This is usually the signature of a non-stationary time series.

```
tsdisplay(unadjusted.ts, lag=48)
```



Note that the seasonal lags of the PACF and ACF show:

- exponential decay in the seasonal lags of the ACF
- a single significant spike at lag 12 in the PACF.

Thus, we can safely assume the data is not stationary which means we need to transform the data such that our process is stationary on mean and variance. This can be done via a series of differenciations, log transformation or linear regressions.

Transforming the unadjusted retail sales data:

The evolution of the retail sales data seems to follow a linear trend, let's model it and remove the trend to the data.

We model the trend using a linear regression.

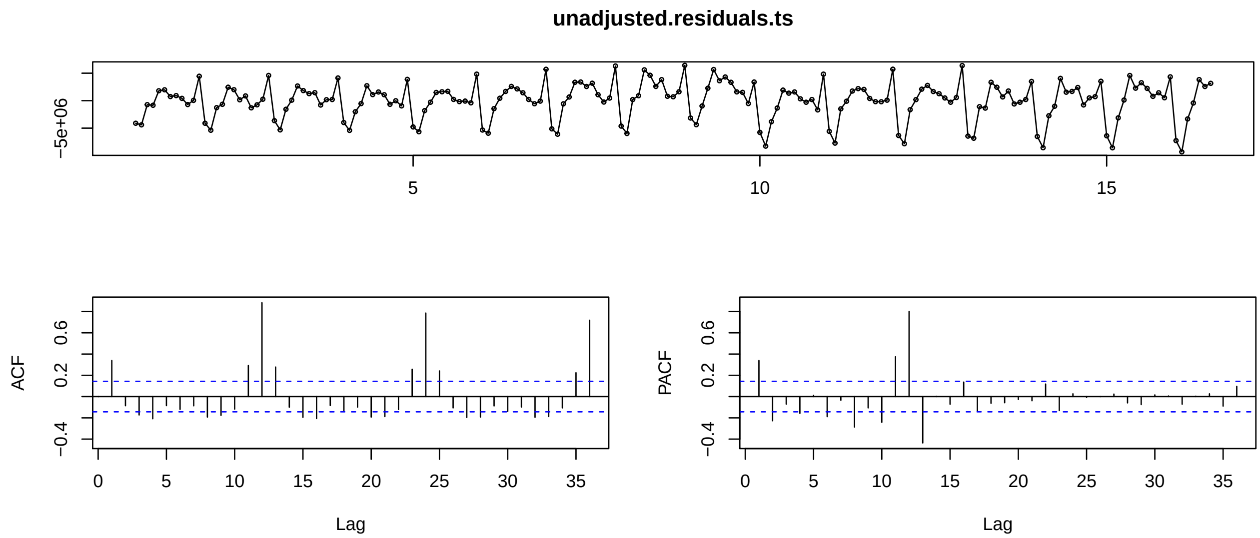
```
unadjusted.regression <- lm(data$unadjusted~data$adjustments)
```

Then we remove the trend from the data by simply getting the residuals of the regression

```
unadjusted.residuals <- residuals(unadjusted.regression)
```

We now have the following ACF and PACF:

```
unadjusted.residuals.ts <- ts(unadjusted.residuals, frequency=12)
tsdisplay(unadjusted.residuals.ts)
```



From that plot, we remark that a strong seasonal component available with a 12 lag period (see lags 12, 24, 36, etc…)

The goal now is to identify presence of AR and MA components in the residuals. Because the seasonal pattern is strong and stable, we want to use an order of seasonal differencing in the model. This, we seasonallly differentiate the residuals with a lag of 12 months.

```
diff.residuals.ts <- diff(unadjusted.residuals.ts, 1, lag=12)
```

Now, let's test that the transformed data trend stationary or if the data requires more levels of differentiation.

```
list(
  adf=adf.test(unadjusted.ts)$p.value < 0.05,
  box=Box.test(unadjusted.ts, lag=12, type='Ljung-Box')$p.value < 0.05,
  kpss=kpss.test(unadjusted.ts)$p.value > 0.05
)
```

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

```
## Warning in kpss.test(unadjusted.ts): p-value smaller than printed p-value
```

```
## $adf
## [1] TRUE
##
## $box
## [1] TRUE
##
## $kpss
## [1] FALSE
```

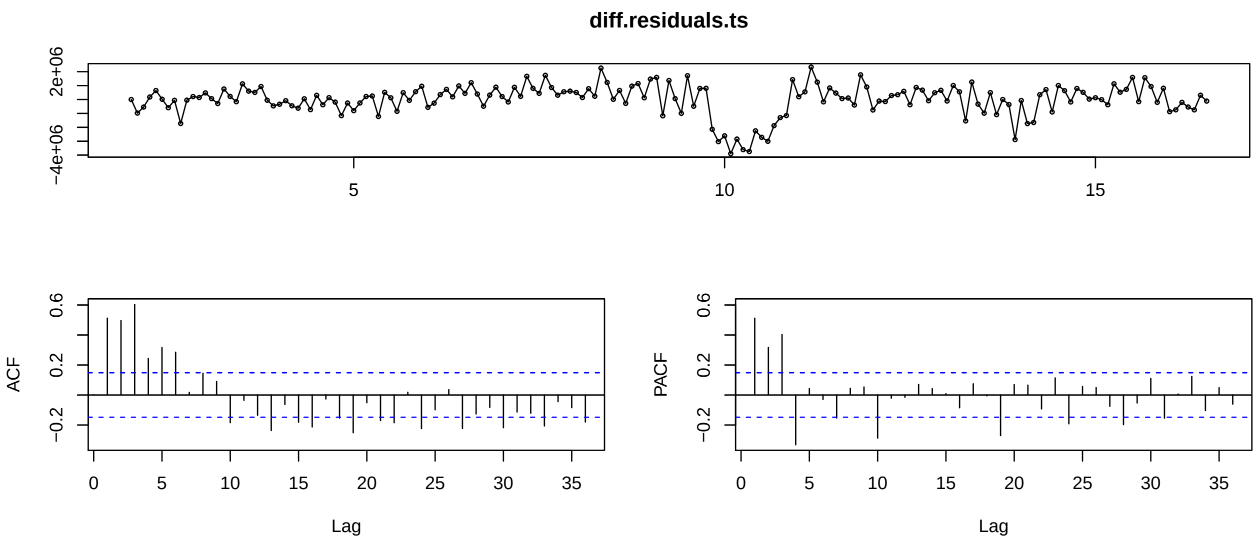
The three standard tests for stationarity seem to indicate we don't need to differenciate the data more.

We can then start to anaylise the transformed data in order to estimate a model.

Esitmating an ARIMA model:

The estimation of an ARIMA model is done by looking at the autocorrelograms.

```
tsdisplay(diff.residuals.ts)
```



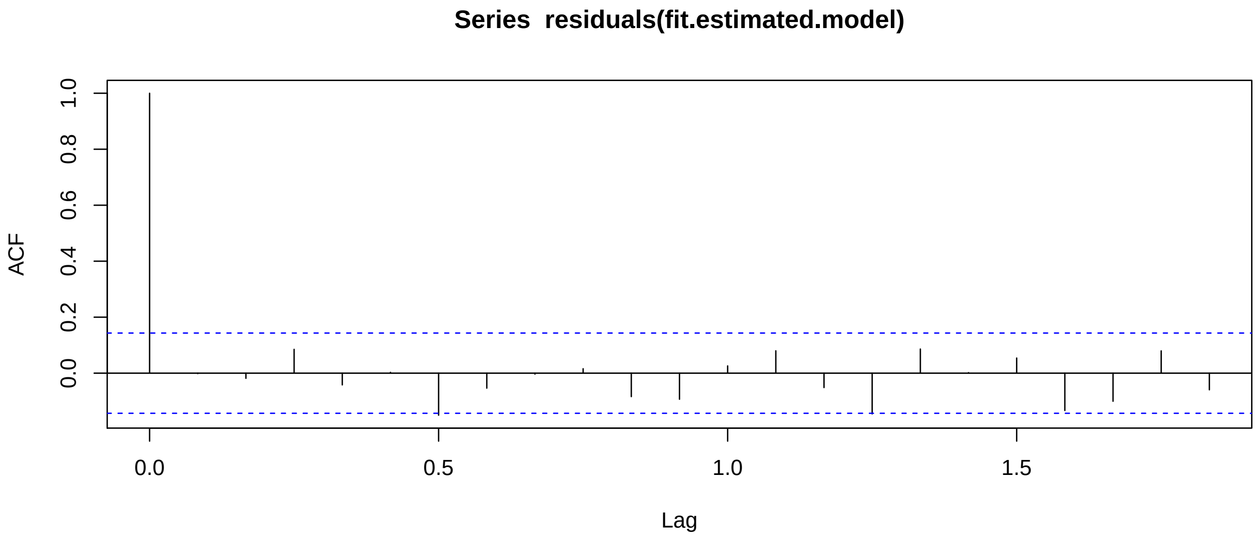
We see that the partial autocorrelation at `lag 1` is positive and exceeds the significance bounds (~0.5), while the partial autocorrelation at `lag 4` is negative and also exceeds the significance bounds (~-0.3). The partial autocorrelations tail off to zero after lag 4. Finally, there are three significant spikes in the PACF suggesting a possible AR(3) term. The pattern in the ACF is not indicative of any simple model.

We can then try to fit our estimated model on the unadjusted retail sales data:

```
fit.estimated.model <-
  Arima(unadjusted.ts,
        order=c(3, 0, 4),
        seasonal=list(order=c(2, 1, 2), period=12)
  )

fit.estimated.model
```

```
## Series: unadjusted.ts
## ARIMA(3,0,4)(2,1,2)[12]
##
## Coefficients:
##           ar1     ar2     ar3     ma1     ma2      ma3      ma4    sar1     sar2     sma1    sma2
##       -0.1625  0.1382  0.9695  0.6629  0.4717  -0.4008  -0.0428  1.3375  -0.4343  -1.9024  0.9963
## s.e.   0.0152  0.0170  0.0211  0.0804  0.0939   0.0985   0.0983  0.1051   0.1299   0.1821  0.1589
##
## sigma^2 estimated as 3.904e+11:  log likelihood=-2600.89
## AIC=5225.79   AICc=5227.71   BIC=5263.77
```

```r
plot(acf(residuals(fit.estimated.model)))
```



Let's test whether there is significant evidence of non-zero correlation with a Ljung-Box test.

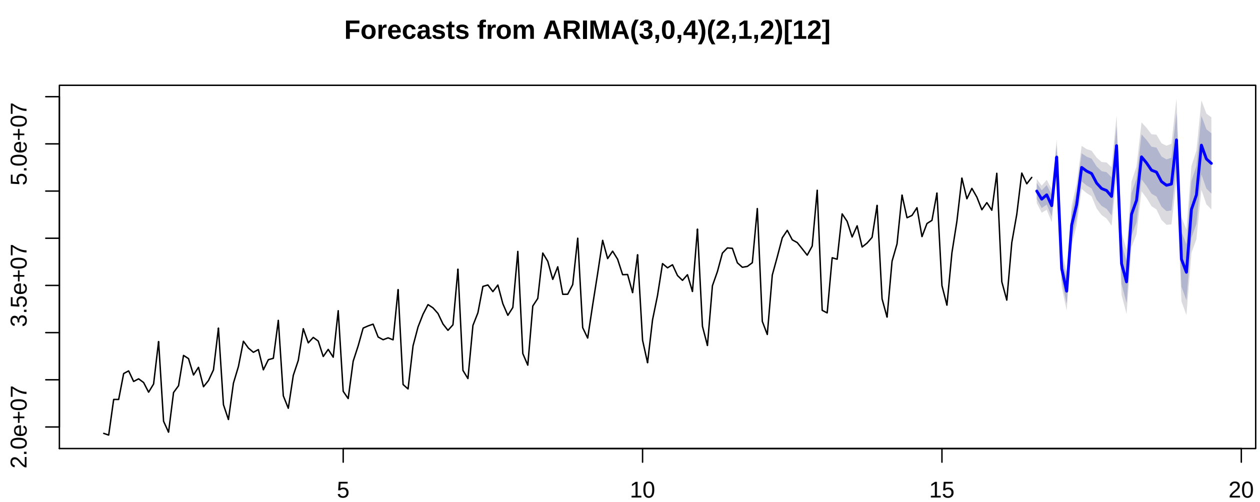
```r
Box.test(residuals(fit.estimated.model), lag=20, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  residuals(fit.estimated.model)
## X-squared = 24.3068, df = 20, p-value = 0.2292
```

The p-value is 0.2292. There is no evidence of non-zero autocorrelation in the residuals.

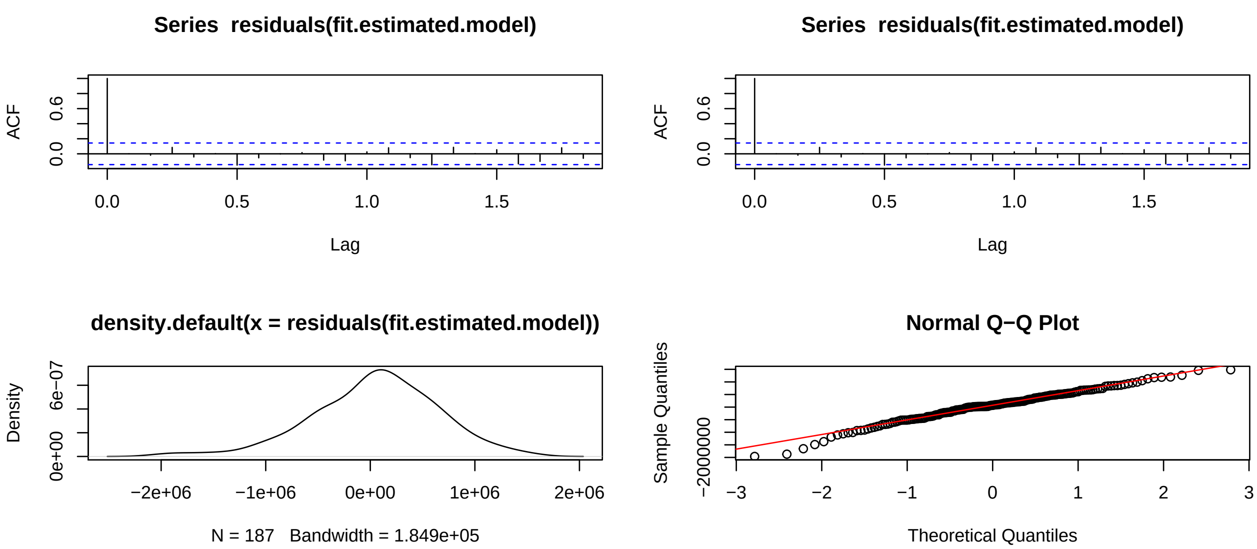
The following plot shows forecasting results of the estimated model:

```r
plot(forecast(fit.estimated.model, h=36))
```



To understand how well the model performs, we need to look at the distribution of the residuals for the fitted model:

```r
par(mfrow=c(2,2))
plot(acf(residuals(fit.estimated.model)))
plot(density(residuals(fit.estimated.model)))
qqnorm(residuals(fit.estimated.model))
qqline(residuals(fit.estimated.model), col = 2)
```



The successive forecast errors doesn't seem to be auto-correlated. The residuals seem to be normally distributed but has a notable deviations from the straight line are signature of a light left tailed distribution (the density plot confirms this intuition)

We perform a Shapiro test to verify the normality of the distribution.

```r
shapiro.test(residuals(fit.estimated.model))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(fit.estimated.model)
## W = 0.9838, p-value = 0.02912
```

the p-value is < 0.05 so the test rejects the null hypothesis that the data are normal. The fact that ACF/PACF plots does not show correlation in the errors and their distribution is not normal shows that the model is not performing very well and thus can be improved upon.

Ideas to improve the model using ARIMA methods:

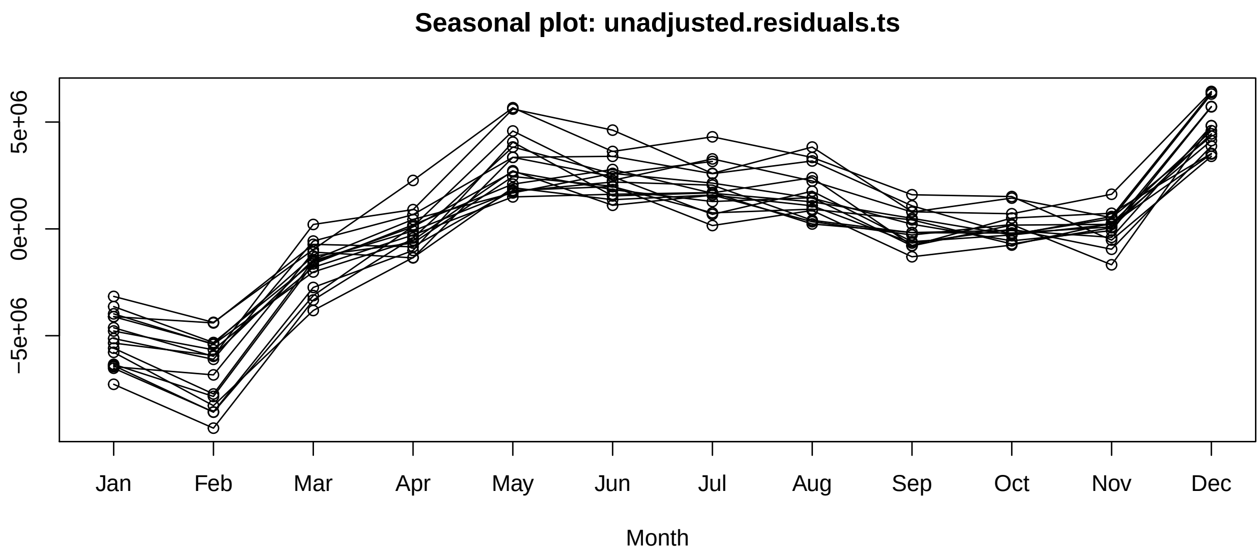
Manually estimating the parameters of the ARIMA model can be a good approach and give satisfying results, but they are quite hard to identify and extremelly subjective. Using automatic methods should overcome that issue.

Automatic methods search over the space of possible models by minimizing the Akaike's Information Criterion (AIC) - which measures the goodness of fit for a particular model by balancing the error of the fit against the number of parameters in the model. Other criterions can be used such that the Biases Corrected AIC or the Bayesian Information Criterion.

An other step would be to introduce variables based on subjective understanding of the data. In the case of retail sales, it seems natural to assume that the retail sales would significantly increase in the Christmas or Easter holidays periods for example. Obviously, domain knowlodge and research is key here and more sofisticated variables can be introduced. ARIMA models allow the introduction of such variables.

Let's assume the month has an effect on the retail sales. The following plot can give a rough idea of the monthly variations:

```
seasonplot(unadjusted.residuals.ts)
```



In R a function `seasonaldummy` generates those variables for us but it can be easily done manually

```
months <- data.frame(months(data$adjustments))
colnames(months) <- c('month')

holidays.regressor.df <-
  months %>%
  mutate(has.holidays=ifelse(month == 'December' | month == 'May', 1, 0))
```

```
order <- c(1, 1, 4)
seasonal.order <- c(0, 0, 2)

fit.arima <-
Arima(unadjusted.ts,
    order=order,
    seasonal=list(order=seasonal.order, period=12)
  )

fit.arima.christmas.easter <-
Arima(unadjusted.ts,
    order=order,
    seasonal=list(order=seasonal.order, period=12),
    xreg=holidays.regressor.df$has.holidays
  )

fit.arima.seasonal.dummy <-
Arima(unadjusted.ts,
    order=order,
```

```r
    seasonal=list(order=seasonal.order, period=12),
    xreg=seasonaldummy(unadjusted.ts)
  )



accuracy(fit.arima)
```

```
##                    ME    RMSE     MAE       MPE     MAPE      MASE        ACF1
## Training set 264055.5 1434415 1064952 0.5604921 3.296428 0.7037006 -0.02590868
```

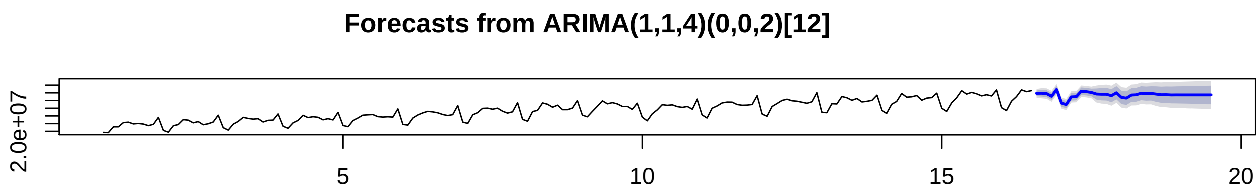
```r
accuracy(fit.arima.christmas.easter)
```

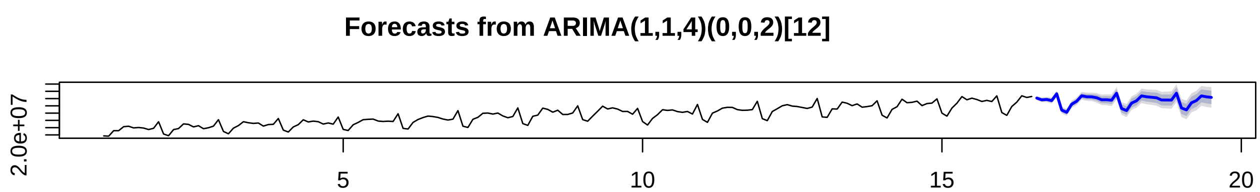
```
##                    ME    RMSE      MAE       MPE     MAPE      MASE        ACF1
## Training set 219205.5 1204026 963870.4 0.5302437 2.992015 0.6369077 -0.01331867
```

```r
accuracy(fit.arima.seasonal.dummy)
```

```
##                    ME     RMSE      MAE       MPE     MAPE      MASE        ACF1
## Training set 104146.5 715132.9 567765.8 0.3014853 1.738695 0.3751692 -0.02917636
```

```r
par(mfrow=c(2,1))
plot(forecast(fit.arima, h=36))
plot(forecast(fit.arima.seasonal.dummy, xreg=seasonaldummyf(unadjusted.ts, h=36)))
```





Cross Validation:

In order to choose the best approach to model our data, we can not only based our decision on how well the model fits historical data. The accuracy of a forecasting should be determined by considering how well a model performs on new data that were not used when fitting the model.

To do so, we are going to use a variation of the cross validation adapted to the context of time series forecasting, it is equivalent to evaluate forecasting models with a rolling origin. And look at different evaluation metrics at different "horizons". The main question when evaluating a forecasting model being: how well does the model forecasts after 1 months, 2 months and so on?

Model's evaluation will be based on the two following metrics:

- The Mean Absolute Percentage Error (MAPE) which is a scale-independent measure of the forecasting errors.
- And, the Mean Absolute Scaled Error (MASE). MASE has the property that it is greater than one if the forecast is worse than the average naïve forecast.

To compute MAPE and MASE, we define two functions that computes the MAPE and the Mean Average Error (MAE)

```
mape <- function(actual, forecast) {
  return(abs((actual-forecast)/actual))
}

mae <- function(actual, forecast) {
  return(abs(actual-forecast))
}
```

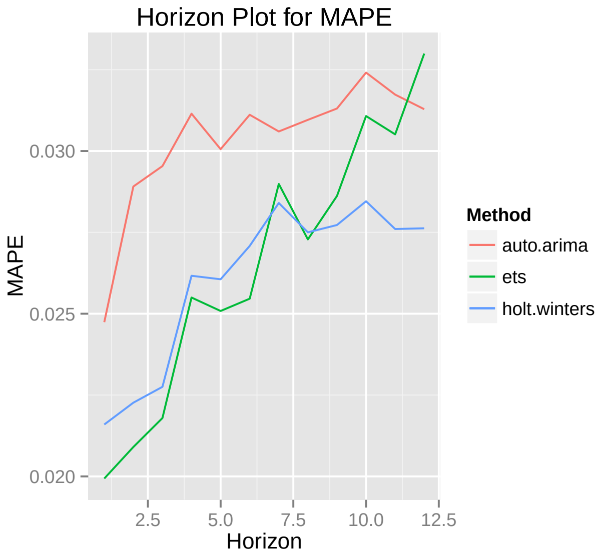
The following procedure computes MASE and MAPE in the context of time series cross-validation:

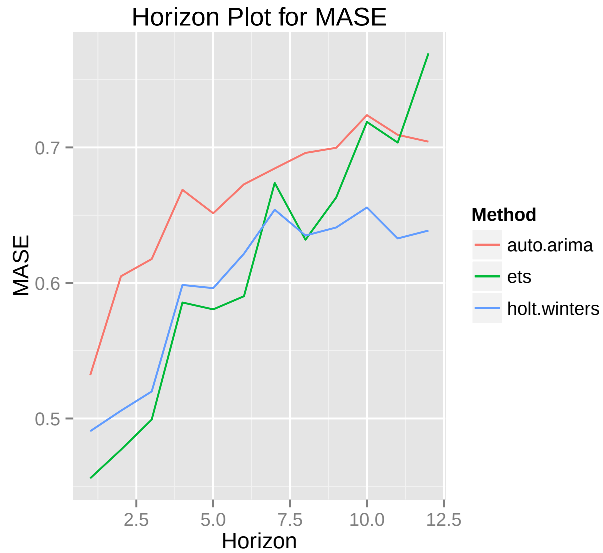
```
k <- 12
n <- length(unadjusted.ts)

frequency <- tsp(unadjusted.ts)[3]
start <- 3
end <- tsp(unadjusted.ts)[2] - k/12

default.matrix <- matrix(NA,ceiling(end-start)*12,12)

mape.auto.arima <- mase.auto.arima <- mae.auto.arima <- default.matrix
mape.ets <- mase.ets <- mae.ets <- default.matrix
mape.holt.winters <- mase.holt.winters <- mae.holt.winters <- default.matrix

count <- 0

for(i in seq(start, end, by=1/12)) {
  train <- window(unadjusted.ts, end=i)
  test <- window(unadjusted.ts, start=i+1/12, end=i+k/12)

  # auto.arima:
  fit.auto.arima <- auto.arima(train)
  forecast.auto.arima <- forecast(fit.auto.arima, h=12)

  # ETS
  fit.ets <- ets(train)
  forecast.ets <- forecast(fit.ets, h=12)

  # Holt-Winters
  fit.holt.winters <- HoltWinters(train, seasonal='additive')
  forecast.holt.winters <- forecast(fit.holt.winters, h=12)

  mape.auto.arima[count, 1:length(test)] <- mape(test, forecast.auto.arima[['mean']])
  mape.ets[count, 1:length(test)] <- mape(test, forecast.ets[['mean']])
  mape.holt.winters[count, 1:length(test)] <- mape(test, forecast.holt.winters[['mean']])
```

```
  mae.auto.arima[count, 1:length(test)] <- mae(test, forecast.auto.arima[['mean']])
  mae.ets[count, 1:length(test)] <- mae(test, forecast.ets[['mean']])
  mae.holt.winters[count, 1:length(test)] <- mae(test, forecast.holt.winters[['mean']])

  count <- count + 1
}

mase.auto.arima <- mae.auto.arima / mean(abs(diff(unadjusted.ts, lag=12)))
mase.ets <- mae.ets / mean(abs(diff(unadjusted.ts, lag=12)))
mase.holt.winters <- mae.holt.winters / mean(abs(diff(unadjusted.ts, lag=12)))
```

Now, we can plot the results:

```
mape.horizon <- data.frame(
  index=seq(1, dim(default.matrix)[2]),
  auto.arima=colMeans(mape.auto.arima, na.rm = TRUE),
  ets=colMeans(mape.ets, na.rm = TRUE),
  holt.winters=colMeans(mape.holt.winters, na.rm = TRUE)
)

mase.horizon <- data.frame(
  index=seq(1, dim(default.matrix)[2]),
  auto.arima=colMeans(mase.auto.arima, na.rm = TRUE),
  ets=colMeans(mase.ets, na.rm = TRUE),
  holt.winters=colMeans(mase.holt.winters, na.rm = TRUE)
)

g.mape <-
  ggplot(mape.horizon, aes(x=index)) +
  geom_line(aes(y=auto.arima, color='auto.arima')) +
  geom_line(aes(y=ets, color='ets')) +
  geom_line(aes(y=holt.winters, color='holt.winters')) +
  labs(title="Horizon Plot for MAPE", x="Horizon", y="MAPE", color="Method")


g.mase <-
  ggplot(mase.horizon, aes(x=index)) +
  geom_line(aes(y=auto.arima, color='auto.arima')) +
  geom_line(aes(y=ets, color='ets')) +
  geom_line(aes(y=holt.winters, color='holt.winters')) +
  labs(title="Horizon Plot for MASE", x="Horizon", y="MASE", color="Method")

grid.arrange(g.mape, g.mase, ncol=2)
```





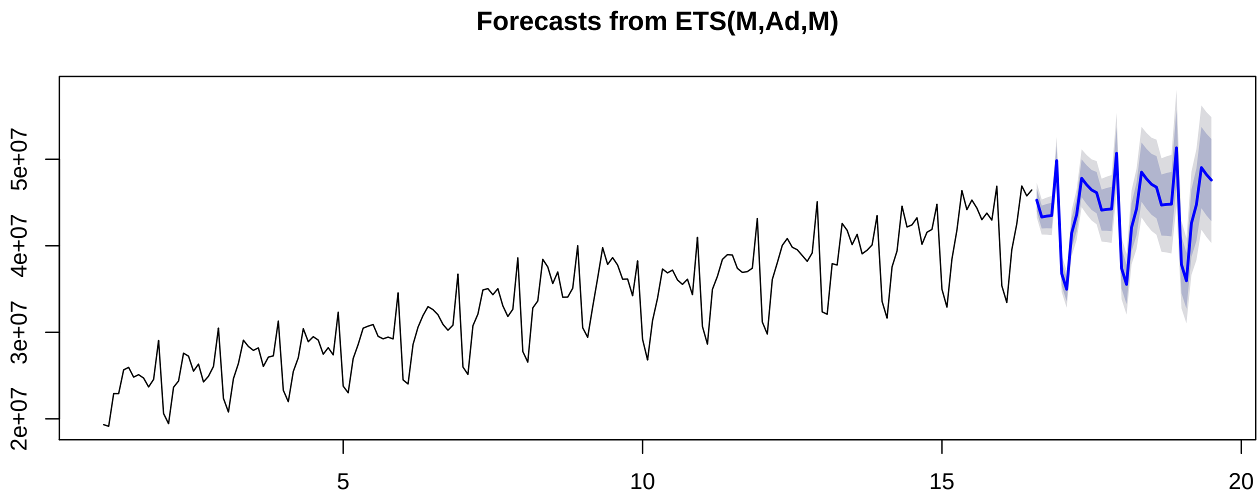
The results compare forecasting results for 3 different families of forecasting methods:

- auto.ARIMA
- ETS, an automatic procedure that find the best forecasting method between all Exponential Smoothing Models: Simple exponential smoothing, Holt's method, Exponential trend method and Holt-Winters. One advantage of the exponential smoothing models is that they can be non-linear. So time series that exhibit non-linear characteristics may be better modelled using exponential smoothing state space models.
- Holt-Winters additive methods (we focus on the additive methodology because the fluctuations around the trend look constant over time). A special case of Exponential Smoothing particulary (Triple exponential smoothing) efficient to model seasonal time series. The smoothing parameters are infered by decomposing the time series in terms of a trend and seasonal component using moving averages.

It seems clear that Holt-Winters Method performs the best overall. But ETS methods performs better for short horizons when Holt-Winters is more performant for larger horizons. So choosing between the two methods depends on the question we want to answer with our forecasting.

Here we want to know how next month's unadjusted Canadian retail sales number, ETS seems more appropriate

```
fit.ets <- ets(unadjusted.ts)
forecast.ets <- forecast(fit.ets, h=36)
plot(forecast.ets)
```



For August 2015, this gives us:

```
forecast.aug.2015 <- 45282203
aug.2015 <- 44975310
error <- 100 * abs(forecast.aug.2015 - aug.2015) / aug.2015
error
```

```
## [1] 0.6823588
```

Conclusion:

In order to forecast the canadian retail sales we first started by estimating an ARIMA model manually and saw that by fine tuning exogeneous regressors one could improve its forecasting capabilities. Then, we saw that in order to elect a good forecasting model one should evaluate the perfomance of the model not on historical data but on data that weren't used to fit the model. A cross-validation procedure, allowed us to choose a model based on several criterions.