post02-Kayla-Simons

Kayla Simons

November 28, 2017

Time Series Analysis and Forecasting on New Zealand Alcohol Consumption

Introduction

In this post, I inted to perform a time series analysis on the alcohol consumption in New Zealand from 2000 to 2012. Through this analysis, I will show how to code a time series analysis. In addition, I will show a forecast of future alcohol sales in New Zealand. I will explain each element of the analysis and forecast, and describe it's meaning. My dataset is a quarterly measure of the total amount of beer, spirits, and wine sold in New Zealand. The data is measured in billion liters sold.

My data comes from http://new.censusatschool.org.nz and can be found here

What is a Time Series Analysis?

A time series is any set of data that is measured periodically in regular intervals. A time series analysis is useful for understanding and forecasting the data.

Loading the Data

```
dat <- read.csv("/Users/krsimons/stat133/stat133-hws-fall17/AlcoholConsumption.csv")
head(dat)
```

```
##     DATE TotalBeer TotalSpirits TotalWine
## 1 2000Q1     2.957        1.097     1.486
## 2 2000Q2     2.823        1.309     1.915
## 3 2000Q3     2.798        1.496     1.844
## 4 2000Q4     3.997        1.810     2.808
## 5 2001Q1     3.037        1.240     1.287
## 6 2001Q2     2.778        1.516     1.861
```

```
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(forecast)
library(tseries)
```

Cleaning the Data

```
#Change DATE to be a numeric vector with quarterly observations
date <- c()
for(i in dat$DATE){
  i = substr(i, 1, 4)
  date <- c(date, i)
}
dat <- dat %>% mutate(DATE = as.numeric(date))
head(dat)
```

```
##   DATE TotalBeer TotalSpirits TotalWine
## 1 2000     2.957        1.097     1.486
## 2 2000     2.823        1.309     1.915
## 3 2000     2.798        1.496     1.844
## 4 2000     3.997        1.810     2.808
## 5 2001     3.037        1.240     1.287
## 6 2001     2.778        1.516     1.861
```

Time Series Modeling

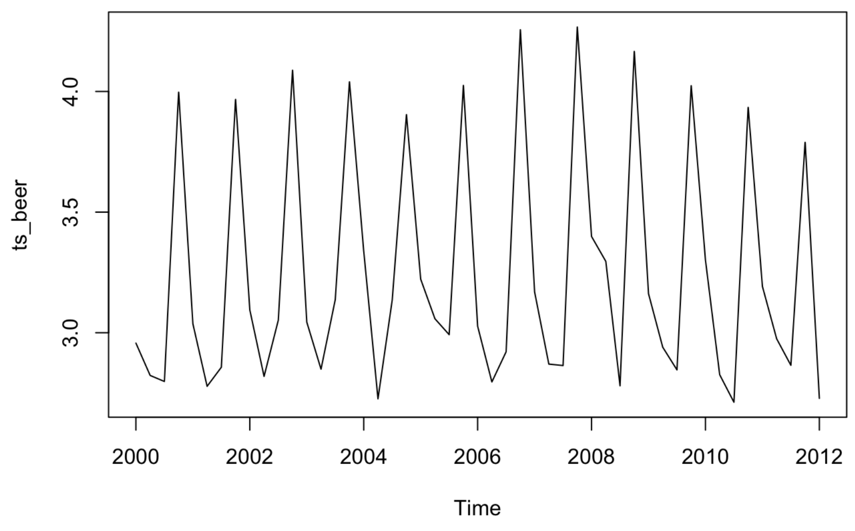
Creating a Time Series

ts() function

The ts() function changes a numeric vector into an R time series object. The function returns a vectors that includes the time indices for each obeservation, sompling frequency, and time between observations. In addition, I made new data frames for beer, spirits, and wine to break up the data and look at one element at a time. This allows for simplification and clearness in writing the code and viewing the plots. Further, having individual data frames is necessary for later analysis.

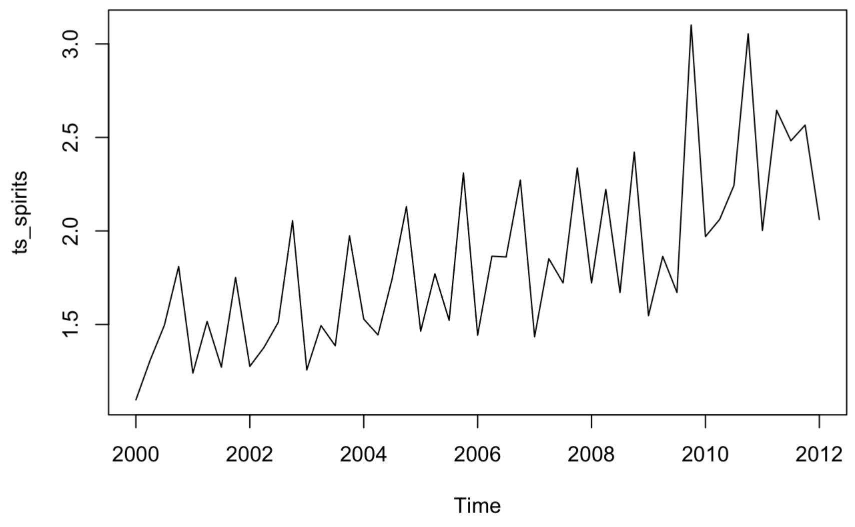
Time series from 2000 to 2012 with quarterly observations for Total Beer

```
beer_dat <- data.frame(dat$TotalBeer)
ts_beer <- ts(beer_dat, start = 2000, end = 2012, frequency = 4)
plot(ts_beer)
```



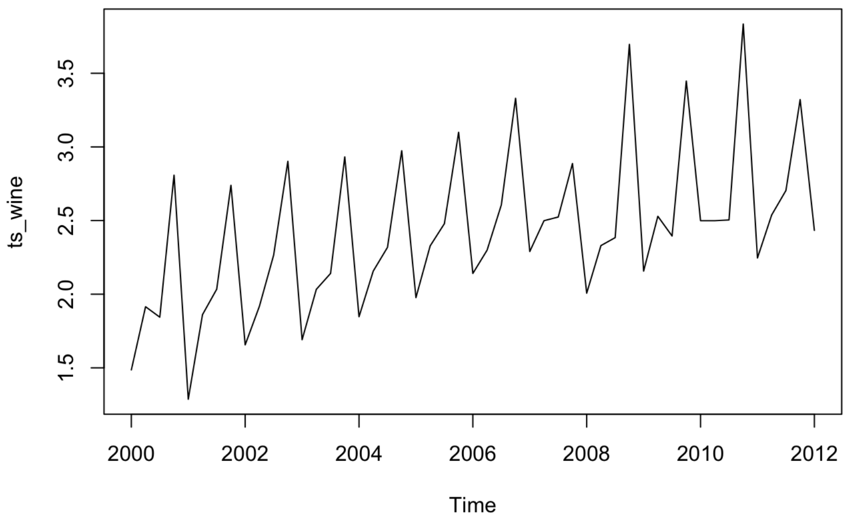
Time series from 2000 to 2012 with quarterly observations for Total Spirits

```
spirits_dat <- data.frame(dat$TotalSpirits)
ts_spirits <- ts(spirits_dat, start = 2000, end = 2012, frequency = 4)
plot(ts_spirits)
```



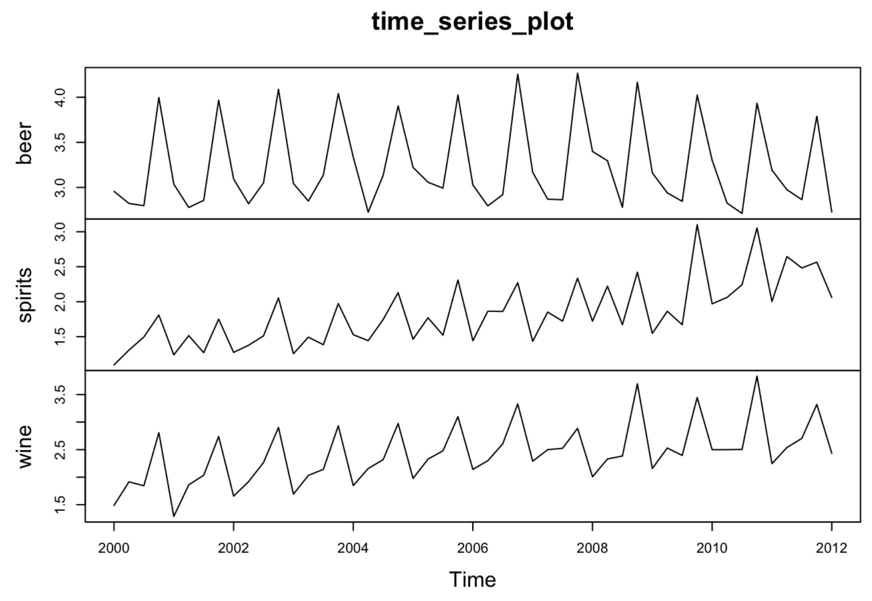
Time series from 2000 to 2012 with quarterly observations for Total Wine

```
wine_dat <- data.frame(dat$TotalWine)
ts_wine <- ts(wine_dat, start = 2000, end = 2012, frequency = 4)
plot(ts_wine)
```



Time series plot for all three categories combined together

```r
time_series_plot <- cbind(beer = ts_beer, spirits = ts_spirits, wine = ts_wine)
time_series_plot <- ts(time_series_plot, start = 2000, frequency = 4)
plot(time_series_plot)
```



Interpreting these plots

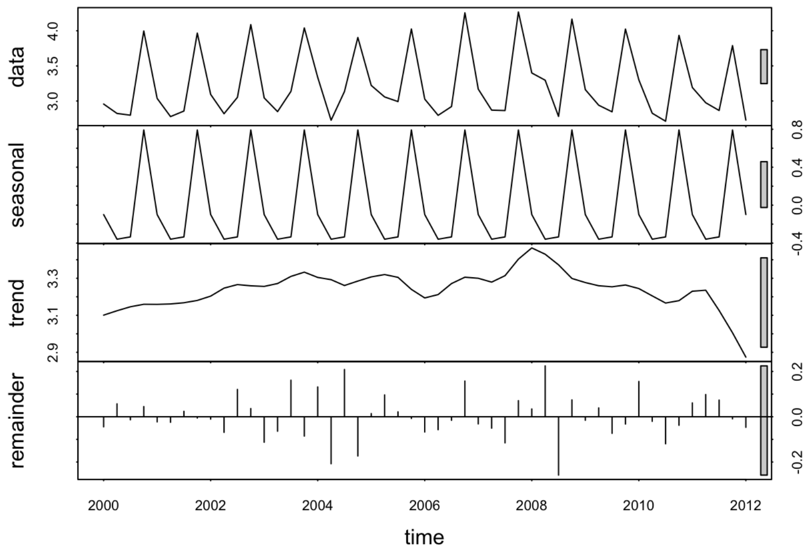
While interpreting these plots, it’s important to look for specific trends in each data set. For example, both wine and spirits appear to be increasing overtime. In addition, beer seems to have decreased slightly in the interim, though not drastically. Further, beer and wine show very clearly show seasonal patterns. The general pattern repeats itself every year. Spirits, however, has a more cyclic movement, where cycles are not repeated a regular intervals, and are not the same shape. There is much more variation in the spirits trend, but it is still possible to find some pattern.

Decomposition

Decomposition breaks up the time series into three components that represent different patterns. The compnents can slightly vary based on the data set, but in this case the time series is made up of a seasonal component, trend-cycle component, and remainder components. The ***seasonal component*** looks at regular fluctations or patterns that repeat themselves on an annual basis (or shorter). The ***trend component*** shows a general graph of long-term changes in the data. The ***cycle component*** is similar to the trend component in that is shows long-term changes. However, the cycle component describes only mildly regular fluctuations, and patterns that last longer than a year. The trend component makes up this data rather than the cycle component. Lastly, the ***remainder component*** or random component show where the data strays from the observed pattern.

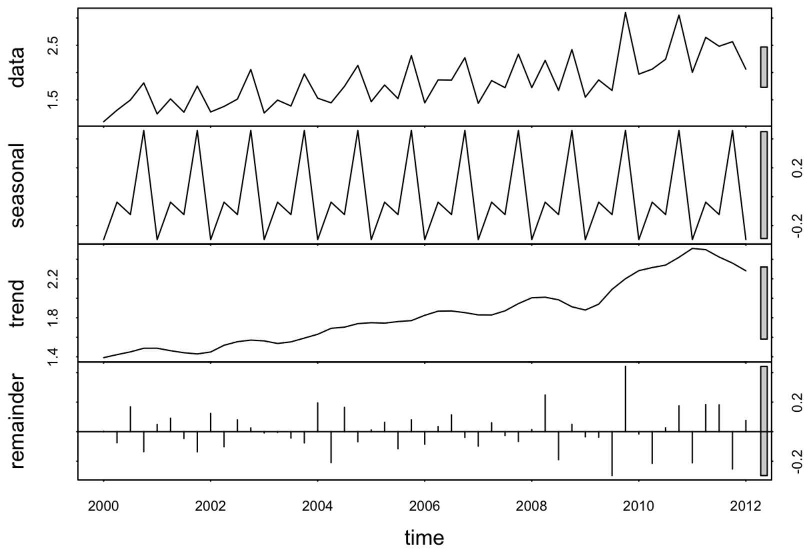
Decomposition for Beer

```r
beer_decomp <- stl(ts_beer, s.window = "period")
plot(beer_decomp)
```



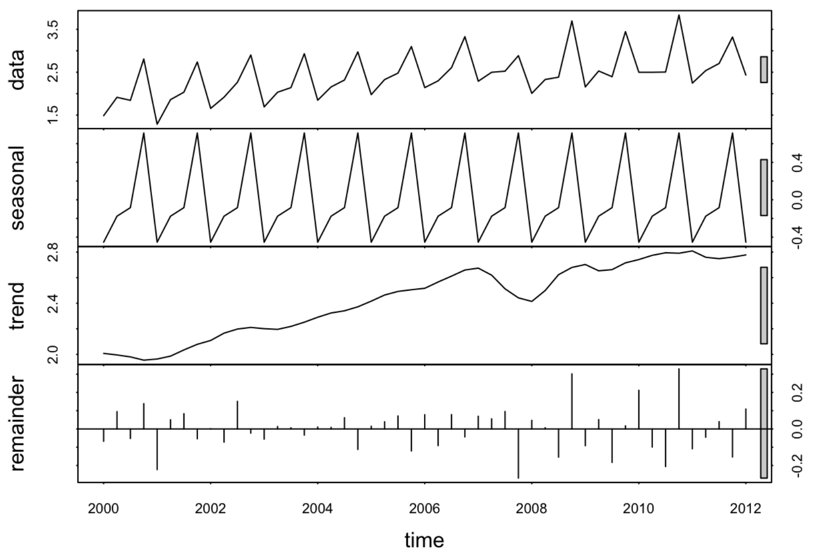
Decomposition for Spirits

```
spirits_decomp <- stl(ts_spirits, s.window = "period")
plot(spirits_decomp)
```



Decomposition for Wine

```
wine_decomp <- stl(ts_wine, s.window = "period")
plot(wine_decomp)
```



Stationary

A stationary time series describes a time series where all the statiscal properties are constant over time. Most time series forecasts perform with the assumption that a time series can be calculated as stationary (otherwise known as "stationarized"). This makes stationarized data extremely important because it allows for future predictions to be based on past data.

Augmented Dickey-Fuller Test

The augmented Dickey-Fuller Test (i.e. adf()) tests if a unit root is present in the time series data. For simplicity, the presence of a unit root is synomous with whether or not a time series stationary. If the p-value is less than 0.05 than we can reject the null hypothesis and see that a unit root is **not** present, and thus the time series is stationary. In the case of this data, the p-vaule for beer, spirits, and wine is greater than 0.05, and thus we find the data to be non-stationary.

KPSS Test for Level Stationary

The Kwiatkowski-Phillips-Schmidt-Shin (KPSS) Test for Level Stationary is the exact opposite of the adf() test. The KPSS test has a null hypothesis that the time series is stationary. In this data set, KPSS test returns a low p-value which rejects the null hypothesis, and thus shows the time series is non-stationary.

```
# Testing if time series for beer is stationary
adf.test(ts_beer)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ts_beer
## Dickey-Fuller = -1.2131, Lag order = 3, p-value = 0.8891
## alternative hypothesis: stationary
```

```
kpss.test(ts_beer)
```

```
## Warning in kpss.test(ts_beer): p-value greater than printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  ts_beer
## KPSS Level = 0.057566, Truncation lag parameter = 1, p-value = 0.1
```

```
#Testing if time series for spirts is stationary
adf.test(ts_spirits)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ts_spirits
## Dickey-Fuller = -2.2179, Lag order = 3, p-value = 0.487
## alternative hypothesis: stationary
```

```
kpss.test(ts_spirits)
```

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

```
##
##  KPSS Test for Level Stationarity
##
## data:  ts_spirits
## KPSS Level = 1.8938, Truncation lag parameter = 1, p-value = 0.01
```

```
#Testing if time series for wine is stationary
adf.test(ts_wine)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ts_wine
## Dickey-Fuller = -2.2303, Lag order = 3, p-value = 0.482
## alternative hypothesis: stationary
```

```
kpss.test(ts_wine)
```

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

```
##
##  KPSS Test for Level Stationarity
##
## data:  ts_wine
## KPSS Level = 1.3175, Truncation lag parameter = 1, p-value = 0.01
```

Creating Stationarized Data

Since the adf() and kpss() tests showed that my time series is not stationary, I need to take the necessary measures to stationarize it. To stationarize the data we need to difference it until the time series is stationary.

Differencing Data

Differencing a time series is subtracting each data point from the previous data point. This is most useful is making data stationary, and typically takes one or two sets of differencing to make the data stationary. In this specific time series, the seasonal component is very prevalent. Because of this, the most effective way to stationarize the data is through seasonal differencing.

Seasonal Differencing

nsdiff() shows us the number of times we need to perform differencing on the data set. In this case, the number is one for all of the data.

```
# Number for seasonal differencing needed
nsdiffs(ts_beer)
```

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

```
nsdiffs(ts_wine)
```

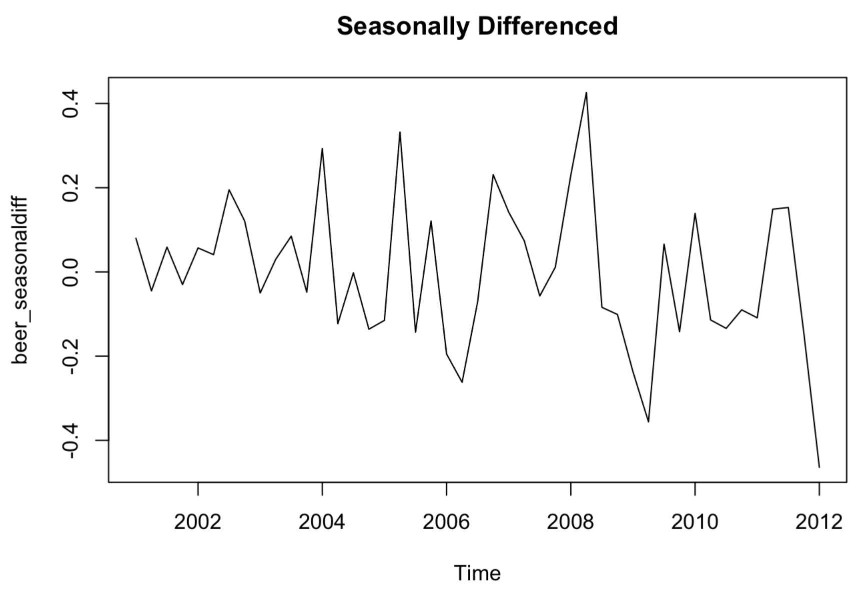
```
## [1] 1
```

```
nsdiffs(ts_spirits)
```

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

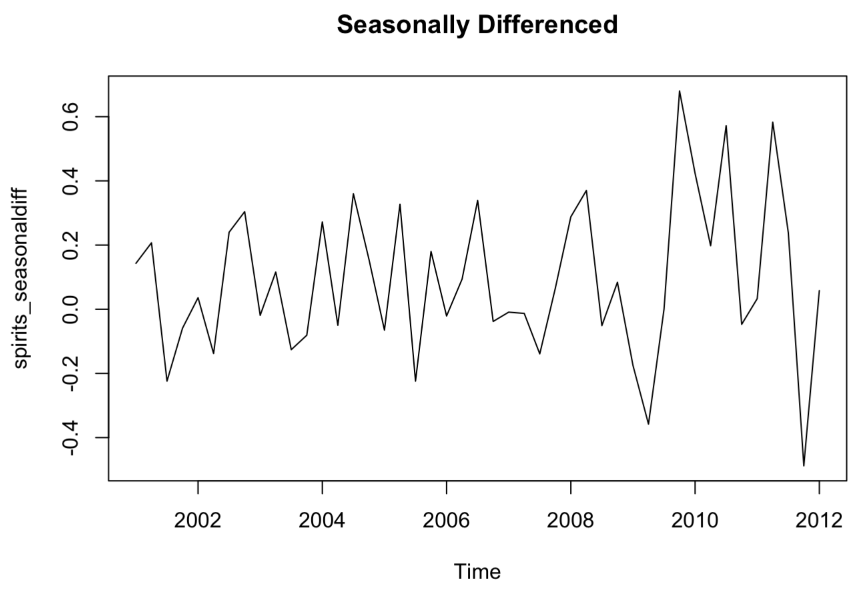
Beer Seasonal Difference

```
beer_seasonaldiff <- diff(ts_beer, lag = frequency(ts_beer), differences = 1)
plot(beer_seasonaldiff, main = "Seasonally Differenced")
```



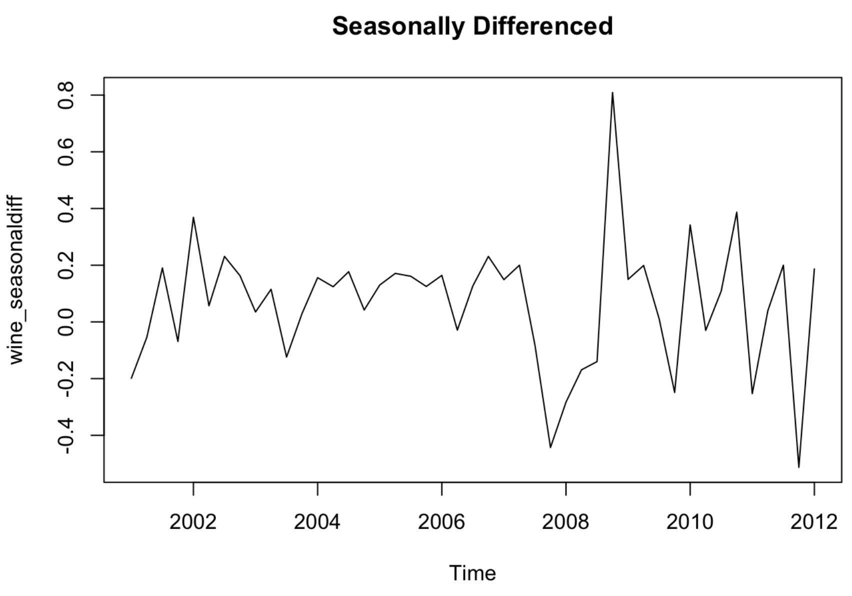
Spirits Seasonal Difference

```
spirits_seasonaldiff <- diff(ts_spirits, lag = frequency(ts_spirits), differences = 1)
plot(spirits_seasonaldiff, main = "Seasonally Differenced")
```



Wine Seasonal Difference

```
wine_seasonaldiff <- diff(ts_wine, lag = frequency(ts_wine), differences = 1)
plot(wine_seasonaldiff, main = "Seasonally Differenced")
```



Forecasting

Exponential Smoothing

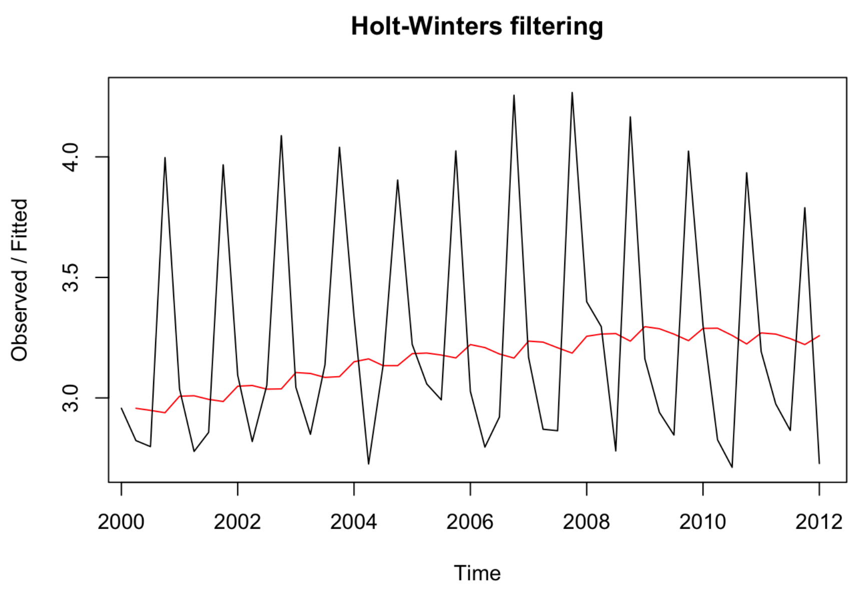
Exponential smoothing helps enable us to predict the future of trends. This can be done using the function HoltWinters(). The HoltWinter function came to existance after Dr. Holt's student expanded on Holt's work from 1957 and created an algorithm for triple exponential smoothing in 1960. For the purpose of simplicity, I am only going to focus on this triple exponential smoothing, and not single and double exponential smoothing. The HoltWinters() function, beta and gamma are set equal to FALSE to estimate the current time point. The red line shows that the analysis produces forecasts future values. An important thing to note about HoltWinters() exponential smoothing is that it focuses more heavily on more recent observations. The amount of weight given to recent observations can be seen through the value of alpha. The closer alpha is to zero, the lesser the weight is placed on recent observations for predicting future forecasts.

Beer Exponential Smoothing

```
beer_hw <- HoltWinters(ts_beer, beta = F, gamma = F)
beer_hw
```

```
## Holt-Winters exponential smoothing without trend and without seasonal component.
##
## Call:
## HoltWinters(x = ts_beer, beta = F, gamma = F)
##
## Smoothing parameters:
##  alpha: 0.06462416
##  beta : FALSE
##  gamma: FALSE
##
## Coefficients:
##       [,1]
## a 3.223824
```

```
plot(beer_hw)
```

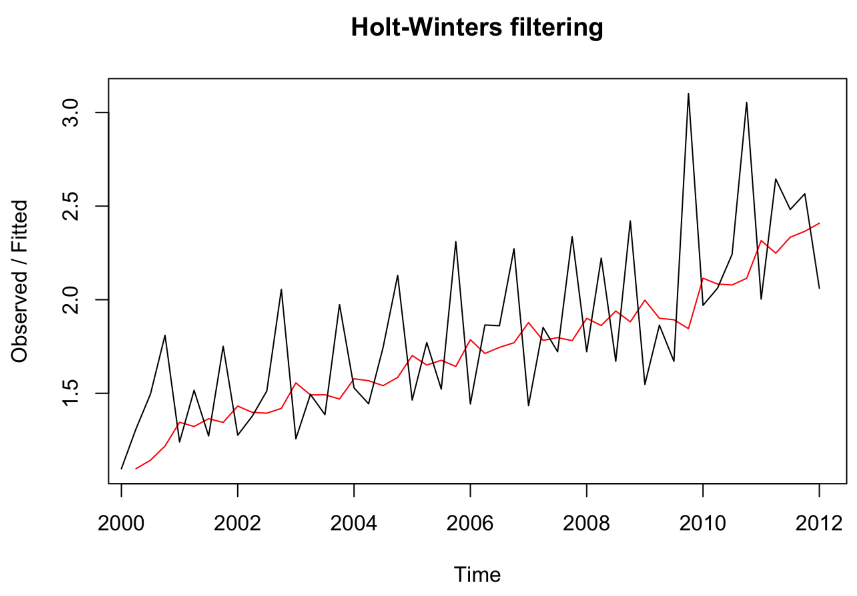


Spirits Exponential Smoothing

```r
spirits_hw <- HoltWinters(ts_spirits, beta = F, gamma = F)
spirits_hw
```

```
## Holt-Winters exponential smoothing without trend and without seasonal component.
##
## Call:
## HoltWinters(x = ts_spirits, beta = F, gamma = F)
##
## Smoothing parameters:
##  alpha: 0.2145627
##  beta : FALSE
##  gamma: FALSE
##
## Coefficients:
##       [,1]
## a 2.333994
```

```r
plot(spirits_hw)
```

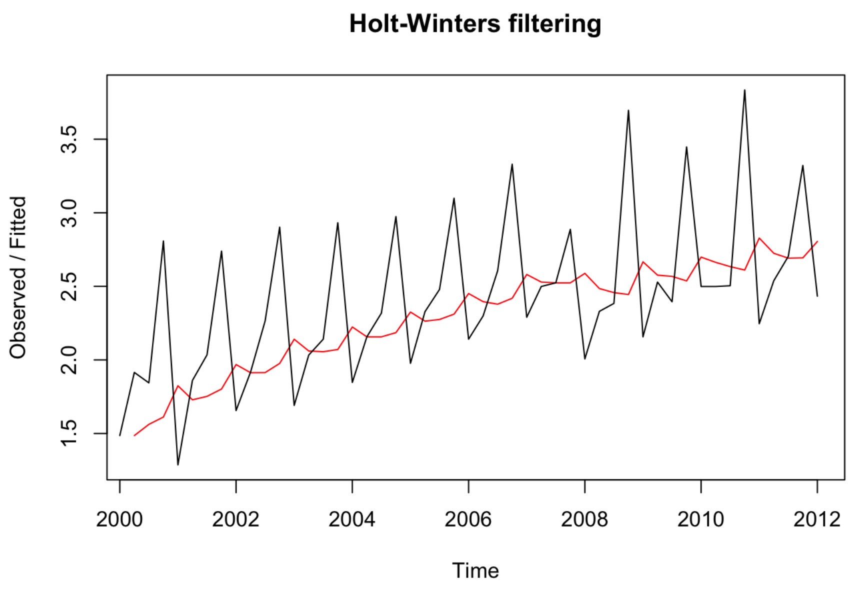


Wine Exponential Smoothing

```r
wine_hw <- HoltWinters(ts_wine, beta = F, gamma = F)
wine_hw
```

```
## Holt-Winters exponential smoothing without trend and without seasonal component.
##
## Call:
## HoltWinters(x = ts_wine, beta = F, gamma = F)
##
## Smoothing parameters:
##  alpha: 0.1775241
##  beta : FALSE
##  gamma: FALSE
##
## Coefficients:
##       [,1]
## a 2.739083
```

```
plot(wine_hw)
```

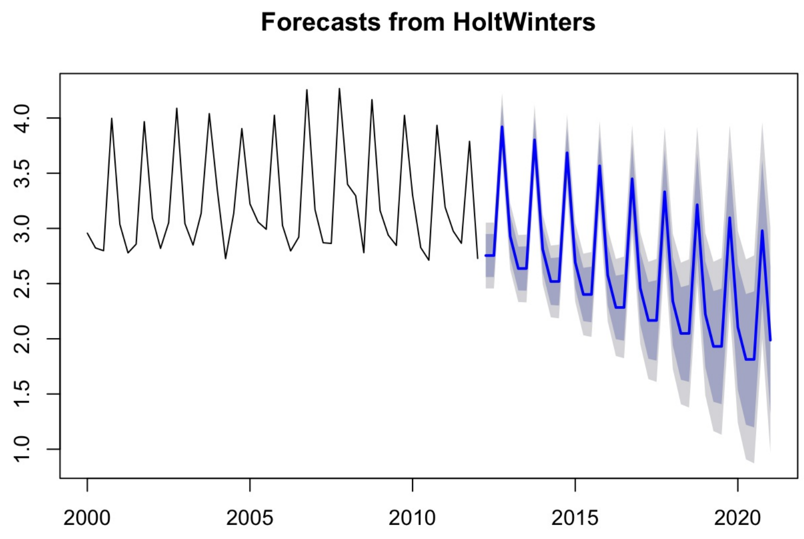


HoltWinters Forecast

Finally, after several layers of analysis, we are able to create a graph with forecasted data. There are multiple different methods to show forecasted data, and HoltWinters() is amongst the most popular. As already mentioned in the HoltWinters() exponential smoothing description, the forecast places a heavier weight on more recent data. The graphs below show the most accurate predications of future forecasts for the 36 months after the time series ends.

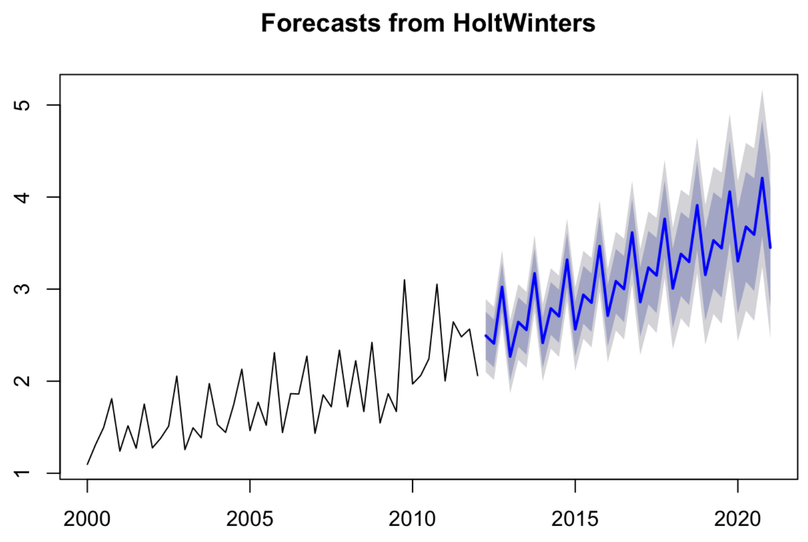
Beer Forecast

```
plot(forecast(HoltWinters(ts_beer), h = 36))
```



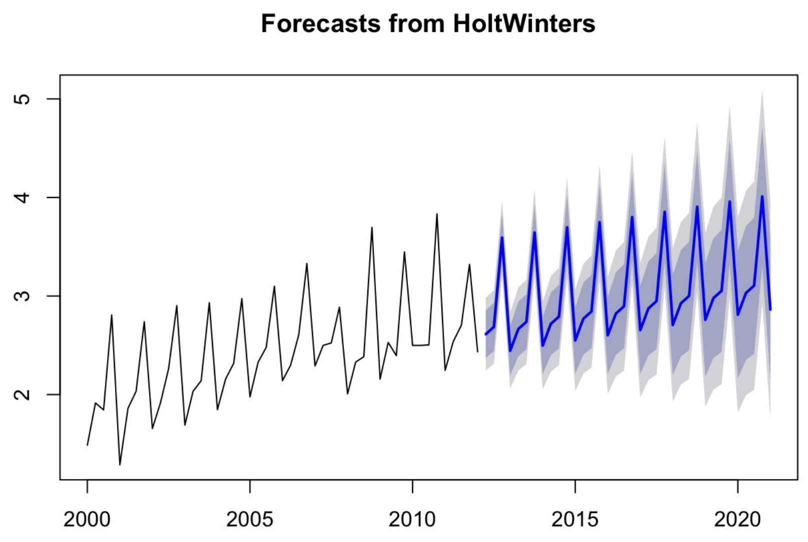
Spirits Forecast

```
plot(forecast(HoltWinters(ts_spirits), h = 36))
```



Wine Forecast

```
plot(forecast(HoltWinters(ts_wine), h = 36))
```



Limitations

My goal to keep this post simple and straightforward created some limitations that are important to note. For example, in the interest of simplicity I did not explain other type of differencing. In addition, I purposefully excluded single and double exponential smoothing because I wanted to focus soley on HoltWinters() aka a triple exponential smoothing. Finally, there are several ways in which one could forecast data. An ARIMA analysis is one of the most popular methods to forecast data. Similar to exponential smoothing, I did not show an ARIMA analysis because I wanted to focus on HoltWinters(). However, even with this limitations, this post is still very comprehensive in how to perform a time series analysis and forecast, while maintaining simplicity.

Findings and Conclusion

Overall, this post shows the steps necessary to perform a time series analysis and forecast through looking at data from sales in New Zealand's alcoholic beverages from 2000-2012. The data shows the beer sales is predicted to drastically decline over the next 36 months, while spirits and wine will greatly increase. Atime series analysis may seem complex, however it is actually very simple to perform if the correct steps are followws. In addition, a time series analysis and forecast can be extremely important and useful to companies making decisions about what to sell. Time series analyses are useful in many different realms of the world, and should always be looked at when making decision about future events. Lastly, I want to reiterate that this post is meant for people with relatively little knoweldge of R, and thus kept very simple. A time series analysis and forecast could be made much more complex if desired.

Resources

Data Set

Time Series Analysis

Time Series and Forecasting

TSA: Start to Finish Examples

Interpret the Key Results for Time Series Plot

Time Series Components

Time Series Analysis

Stationarity and Differencing

Time Series Forecasting using Simple Exponential Smooting Method