WeatherCurrencyAndSales

Caroline Silvestre

30/09/2018

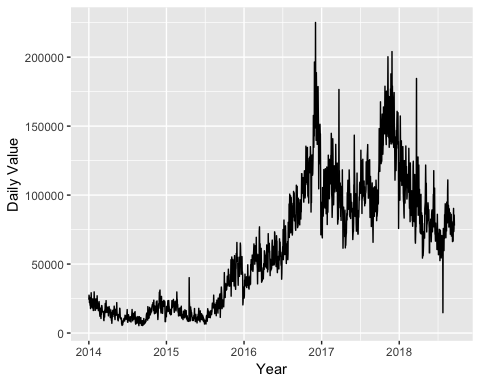
# A program for analysing the relationship between weather patterns, currency values and sales volumes, given a series of daily sale values of a particular product.

#clean up env  
rm(list = ls())  
  
library(readr)  
library(stats)  
library(dplyr)  
library(reshape2)  
library(lubridate)  
library(ggplot2)  
library(forecast)  
library(BBmisc)  
library(zoo)  
  
#reads data from csv and loading sales data with dates and formatting the date field as date in R  
#Note: file must be a csv and the date order must be oldest to newest in the file.  
SalesData <- as.data.frame(read\_csv("SalesReportDaily.csv"))

## Date Daily\_Items\_Sold Total\_Daily\_Value  
## Min. :2014-01-01 Min. : 122 Min. : 5335   
## 1st Qu.:2015-03-06 1st Qu.: 385 1st Qu.: 17310   
## Median :2016-05-08 Median :1122 Median : 56098   
## Mean :2016-05-08 Mean :1191 Mean : 61403   
## 3rd Qu.:2017-07-11 3rd Qu.:1805 3rd Qu.: 95173   
## Max. :2018-09-13 Max. :4270 Max. :225128   
## Avg\_Item\_Value\_GBP Avg\_Trans\_Value\_GBP  
## Min. :29.70 Min. : 73.14   
## 1st Qu.:45.85 1st Qu.:109.26   
## Median :49.72 Median :119.30   
## Mean :49.17 Mean :118.33   
## 3rd Qu.:52.69 3rd Qu.:127.90   
## Max. :62.64 Max. :154.06

# Manipulating data into variables suitable for analysis

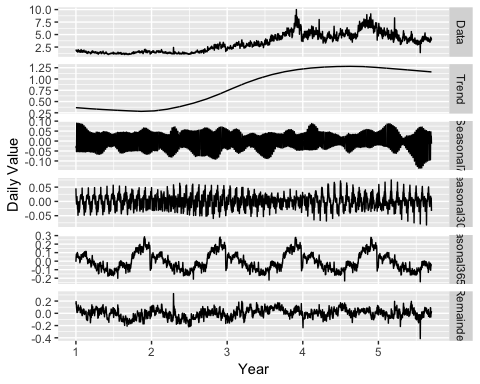
require(lubridate)  
date <- ymd(SalesData$Date)  
SalesData$Date <- date  
SalesData$DoW <- as.factor(weekdays.Date(date))  
SalesData$WeekNum <- recode(SalesData$DoW,'Monday'=1,'Tuesday'=2, 'Wednesday'=3,'Thursday'=4,  
 'Friday'=5,'Saturday'=6,'Sunday'=7)  
### add the number of the month  
require("lubridate")  
SalesData$MonthNum <- month(SalesData$Date)  
SalesData$MonthFactor <- as.factor(recode(SalesData$MonthNum, '1'="Jan", '2'= "Feb", '3'= "Mar", '4'= "Apr", '5'= "May", '6'= "Jun", '7'="Jul",'8' = "Aug", '9'= "Sep", '10' = "Oct", '11'= "Nov", '12' = "Dec"))  
  
#setting the seasonality variables  
weekly <- 7  
monthly <- 30.5  
yearly <- 364.25  
  
#adding a normalized version of daily sales values to the dataframe, using a 1 to 10 range to avoid zero values  
SalesData$ValNorm <- normalize(SalesData$Total\_Daily\_Value , method="range", range=c(1,10))  
  
#need to transform sales data into ts (time series) objects and then into a msts (multi-seasonal time-series). The default is daily.  
#function to transform a vector into a time series object. Need the day of the year as a number from 1 to 365/366.  
#freq is the data interval, for daily data use yearly.  
  
ts.transform <- function(univariateseries, YYYY,DDD, freq){  
 x = ts(univariateseries, start = c(YYYY,DDD) , frequency = freq)  
 return(x)  
}  
  
tsVal <- ts.transform(SalesData$Total\_Daily\_Value, 2014,001, yearly)

 Transforming each column of data into a time series object that R recognises as such

#time series for all variables were created, but not all were used in the current analysis.  
tsItems <- ts.transform(SalesData$Daily\_Items\_Sold, 2014,001, yearly)  
#autoplot.zoo(tsItems) + ylab("Daily Items") + xlab("Year")  
  
tsATV <- ts.transform(SalesData$Avg\_Trans\_Value\_GBP, 2014,001, yearly)  
#autoplot.zoo(tsATV) + ylab("Daily Items") + xlab("Year")  
  
tsValNorm <- ts.transform(SalesData$ValNorm,2014,001, yearly)  
#autoplot(tsValNorm) + ylab("Sales Volume Normalised") + xlab("Year")  
  
#function for transforming a time series object into a msts, which is a multiple-series time series  
require(forecast)  
msts.transform <- function(tsobj, YYYY,DDD,s1=NULL,s2=NULL,s3=NULL){  
 a = msts(tsobj, start = c(2014,001), seasonal.periods=c(s1,s2,s3))  
 return(a)  
}  
  
#function to decompose both single-seasonal and multi-seasonal time-series  
decomp.msts <- function(mstsobj){  
 xy1 = mstl(mstsobj, iterate = 3)  
 return(xy1)  
}  
  
#By default the Full Daily Sales Value will be used for ease of visualisation, with yearly and weekly pattern, but only normlised plots will be generated for better visualisation  
mstsVal <- msts.transform(tsVal, 2014,001, weekly, yearly)  
decVals <- decomp.msts(mstsVal)

Now the Daily Sales Values will be decomposed using Seasonal Decomposition of Time Series by Loess, which finds the seasonal components by smoothing each seasonal subseries (for example, January values, Monday Values, etc). Once those components are found, they are subtracted from the data and the remainder is smoothed to find the trend component.

#generating a decomposition of the normalised Daily sales values for use in the analysis in conjunction with other variables.  
salesNormmsts\_mwy <- msts(tsValNorm, seasonal.periods=c(7,30.5,365.25))  
xnorm\_wmy <- mstl(salesNormmsts\_mwy, lambda = "auto", iterate = 3)  
  
#generating a dataset with decomposed values to use with GAM  
dat\_decomp <- as.data.frame(xnorm\_wmy)  
dat\_decomp$Date <- date  
  
#mergin the decomposition to the original data frame  
SalesData<- merge(SalesData, dat\_decomp, by.x = "Date", by.y = "Date", all.x = TRUE)

Here the Daily Sales time-series was decomposed into five elements: - Trend - Weekly seasonality (7 day period) - Monthly seasonality (30.5 day period) - Yearly seasonality (365.25 day period)  It is possible to see that the strongest component is the Yearly one, showing a large peak in sales in the last quarter of each year. There is also a remarkable upward trend in the data.

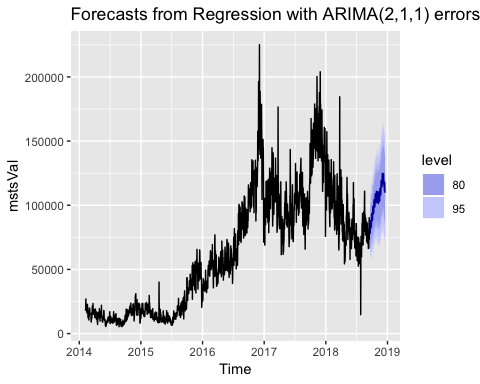
# Notes on forecasting based on sales data and seasonality only

When x is a time-series object, K should be a vector of integers specifying the number of sine and cosine (Fourier) terms to be used with each of the seasonal periods (if more than one) for generating a forecast. The minimum number is 1, and the maximum number is half as many periods of season are in the data. So for yearly periods, the maximum K is 365.25/ 2 (only integer values are allowed). Therefore, the best way to find a suitable number of Fourier terms, it is advisable to do it by computing the lowest AICc. However, due to the large amount of data, the code to make this calculation takes several minutes to run (over 30 minutes). The code below had been added for reference, but it will not be used. This program will not be running a forecast using ARIMA with fourier terms for two reasons: First is that the calculation for large datasets with more than one seasonal element takes too long, and second is that this model does not allow for the introduction of extra variables.

#salesmsts = msts(timeseriesValue, start = c(2014,001), seasonal.periods=c(365.25))  
#bestfit <- list(aicc=Inf)  
#for(K in seq(25)) {  
 # fit <- auto.arima(salesmsts, xreg=fourier(salesmsts, K=K),  
 # seasonal=FALSE)  
 #if(fit[["aicc"]] < bestfit[["aicc"]]) {  
 # bestfit <- fit  
 #bestK <- K  
 #}  
#}  
#fc <- forecast(bestfit,  
 # xreg=fourier(salesmsts, K=bestK, h=6\*365.25))  
  
#autoplot(fc)  
  
#fit[["aicc"]]

# Arima forecast with weekly and yearly seasonalities, for future comparison with GAM models.

#the value of K = 19 was found by running a script for a few hours, based on the lowest AIcc  
#The code below retunrs forecasting using a dynamic regression for 90 days ahead and it takes several minutes to run  
fitwy <- auto.arima(mstsVal, seasonal=FALSE, xreg = fourier(mstsVal, K=c(3,10)))  
fitwy %>% forecast(xreg=fourier(mstsVal,K=c(3,10), h=1\*90)) %>% autoplot(include=5\*336)



summary(fitwy)

## Series: mstsVal   
## Regression with ARIMA(2,1,1) errors   
##   
## Coefficients:  
## ar1 ar2 ma1 S1-7 C1-7 S2-7  
## 0.4267 0.0743 -0.9060 -2603.3502 1949.6165 -1387.9681  
## s.e. 0.0298 0.0276 0.0168 376.4051 376.4324 261.3046  
## C2-7 S3-7 C3-7 S1-364 C1-364 S2-364  
## -1040.6294 2098.7173 -625.7561 -7922.097 13865.397 -5597.822  
## s.e. 261.3943 237.7625 237.7207 3733.828 3755.509 1968.247  
## C2-364 S3-364 C3-364 S4-364 C4-364 S5-364  
## 399.585 -3975.716 -2723.015 -1434.938 -3085.695 -1880.0160  
## s.e. 1938.368 1384.519 1385.371 1121.495 1115.217 971.8072  
## C5-364 S6-364 C6-364 S7-364 C7-364 S8-364  
## 799.4842 -2308.4920 -2072.0894 -598.8565 -2474.5774 -443.8450  
## s.e. 967.4401 877.6466 876.8589 817.4050 813.2323 771.2511  
## C8-364 S9-364 C9-364 S10-364 C10-364  
## 265.2744 1025.5245 -243.2367 1150.8168 -575.8677  
## s.e. 771.1854 739.1654 736.3954 711.2022 710.5021  
##   
## sigma^2 estimated as 98143688: log likelihood=-18209.53  
## AIC=36479.07 AICc=36480.17 BIC=36642.5  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set 199.8866 9819.821 6481.4 -1.678859 14.91504 0.2069251  
## ACF1  
## Training set 0.001958019

Joining historical weather observations values and currency exchange vales by date.

WeatherData <- read\_csv("WeatherDaily.csv")

## Parsed with column specification:  
## cols(  
## DATE = col\_character(),  
## PRCP = col\_double(),  
## TAVG = col\_double()  
## )

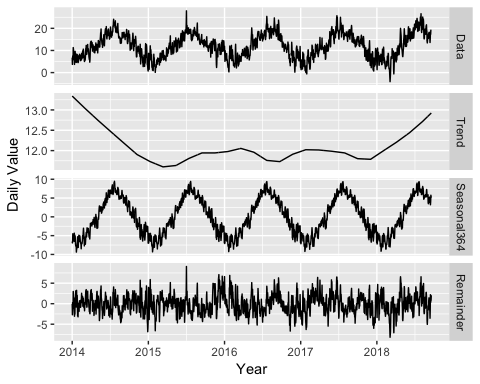
weatherdates <- as.Date(WeatherData$DATE, "%d/%m/%Y")  
#View(weatherdates)  
WeatherData$DATE <- weatherdates  
#remove NA's  
WeatherData$PRCP[is.na(WeatherData$PRCP)] <- 0  
  
#uploading currency data  
CurrencyData <- CurrencyExchangeDaily <- read\_csv("CurrencyExchangeDaily.csv")

## Parsed with column specification:  
## cols(  
## Date = col\_character(),  
## Price = col\_double(),  
## Open = col\_double(),  
## High = col\_double(),  
## Low = col\_double(),  
## `Change %` = col\_character()  
## )

currencydates <- mdy(CurrencyData$Date)  
CurrencyData$Date <- currencydates  
#trading stops on weekends, so weekends will carry the latest value after the merge of datasets  
  
#removing data we don't need  
CurrencyData$Open <- NULL  
CurrencyData$High <- NULL  
CurrencyData$Low <- NULL  
CurrencyData$Change <- NULL  
  
  
#merging weather and sales dataframes  
newdf <- merge(SalesData, WeatherData, by.x = "Date", by.y = "DATE", all.x = TRUE)  
#adding currency data  
newdf <- merge(newdf, CurrencyData, by.x = "Date", by.y = "Date", all.x = TRUE)  
#replacing NA's in currency data with the last value   
newdf$Price <- na.locf(newdf$Price)  
  
#transforming precipitation into factors - 'rain' and 'dry'  
newdf$PrecFact <- newdf$PRCP[newdf$PRCP > 0] <- 1  
newdf$PrecFact <- as.factor(newdf$PRCP)  
  
#running a seasonal decomposition on weather as it is a highly seasonal feature  
tavg.ts <- ts(newdf$TAVG)  
tavg.msts <- msts.transform(tavg.ts,2014,001,yearly)  
tavg.decomp <- decomp.msts(tavg.msts)  
  
  
summary(newdf)

## Date Daily\_Items\_Sold Total\_Daily\_Value  
## Min. :2014-01-01 Min. : 122 Min. : 5335   
## 1st Qu.:2015-03-06 1st Qu.: 385 1st Qu.: 17310   
## Median :2016-05-08 Median :1122 Median : 56098   
## Mean :2016-05-08 Mean :1191 Mean : 61403   
## 3rd Qu.:2017-07-11 3rd Qu.:1805 3rd Qu.: 95173   
## Max. :2018-09-13 Max. :4270 Max. :225128   
##   
## Avg\_Item\_Value\_GBP Avg\_Trans\_Value\_GBP DoW WeekNum   
## Min. :29.70 Min. : 73.14 Friday :245 Min. :1.000   
## 1st Qu.:45.85 1st Qu.:109.26 Monday :245 1st Qu.:2.000   
## Median :49.72 Median :119.30 Saturday :245 Median :4.000   
## Mean :49.17 Mean :118.33 Sunday :245 Mean :3.999   
## 3rd Qu.:52.69 3rd Qu.:127.90 Thursday :246 3rd Qu.:6.000   
## Max. :62.64 Max. :154.06 Tuesday :245 Max. :7.000   
## Wednesday:246   
## MonthNum MonthFactor ValNorm Data   
## Min. : 1.000 Aug :155 Min. : 1.000 Min. : 1.000   
## 1st Qu.: 3.000 Jan :155 1st Qu.: 1.490 1st Qu.: 1.490   
## Median : 6.000 Jul :155 Median : 3.079 Median : 3.079   
## Mean : 6.259 Mar :155 Mean : 3.296 Mean : 3.296   
## 3rd Qu.: 9.000 May :155 3rd Qu.: 4.679 3rd Qu.: 4.679   
## Max. :12.000 Apr :150 Max. :10.000 Max. :10.000   
## (Other):792   
## Trend Seasonal7 Seasonal30.5   
## Min. :0.2821 Min. :-1.362e-01 Min. :-8.251e-02   
## 1st Qu.:0.3524 1st Qu.:-2.177e-02 1st Qu.:-1.654e-02   
## Median :0.9730 Median :-3.098e-03 Median : 1.573e-04   
## Mean :0.8335 Mean :-1.859e-05 Mean : 1.566e-05   
## 3rd Qu.:1.2271 3rd Qu.: 2.618e-02 3rd Qu.: 1.507e-02   
## Max. :1.2805 Max. : 8.974e-02 Max. : 7.506e-02   
##   
## Seasonal365.25 Remainder PRCP TAVG   
## Min. :-0.242018 Min. :-0.4126526 Min. :0.000 Min. :-4.10   
## 1st Qu.:-0.087261 1st Qu.:-0.0475279 1st Qu.:0.000 1st Qu.: 7.90   
## Median :-0.026695 Median : 0.0043379 Median :0.000 Median :12.10   
## Mean :-0.006629 Mean : 0.0003003 Mean :0.446 Mean :12.11   
## 3rd Qu.: 0.065400 3rd Qu.: 0.0516012 3rd Qu.:1.000 3rd Qu.:16.40   
## Max. : 0.286105 Max. : 0.3221262 Max. :1.000 Max. :27.90   
## NA's :4 NA's :4   
## Price Change % PrecFact   
## Min. :1.205 Length:1717 0 :949   
## 1st Qu.:1.311 Class :character 1 :764   
## Median :1.425 Mode :character NA's: 4   
## Mean :1.440   
## 3rd Qu.:1.556   
## Max. :1.717   
##

# Decomposing daily weather data to remove the season element and extract the remainder for use in the analysis.

 Because weather is seasonal as well as sales, decomposing the seasonality from the weather and using the remainder element of decomposition seemed like the best way of retrieving 'unusually' cold or hot days for use in the analysis, instead of actual values. This is because the peak of sales is on the runup to the Chrstmas period, which coincides with the colder months in the nothern hemisphere. If this same analysis were run on the southern hemispehere, there is a change that the results would be completely different.

Temperature <- as.data.frame(tavg.decomp)  
Temperature$Date <- newdf$Date  
#merge datasets again  
newdf <- merge(newdf, Temperature, by.x = "Date", by.y = "Date", all.x = TRUE )  
  
#change days of week and month as factors  
newdf$DoW <- as.factor(newdf$DoW)  
newdf$MonthFactor <- as.factor(newdf$MonthFactor)

The merging of all data that has been extracted, manipulated or kept as original, has resulted in a large dataset. From those resulting variables, different models will be tested for accuracy of forecast.

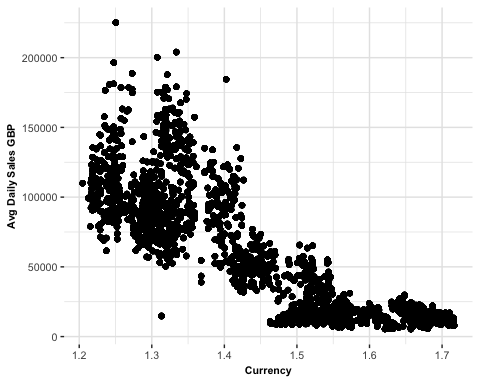
summary(newdf)

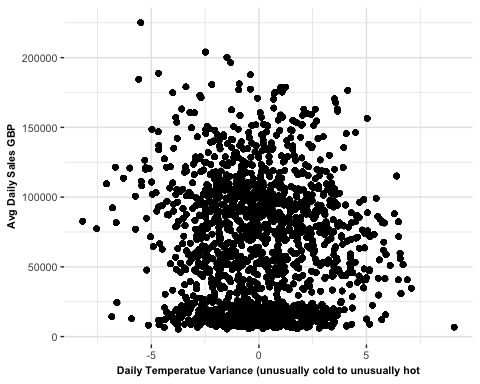
## Date Daily\_Items\_Sold Total\_Daily\_Value  
## Min. :2014-01-01 Min. : 122 Min. : 5335   
## 1st Qu.:2015-03-06 1st Qu.: 385 1st Qu.: 17310   
## Median :2016-05-08 Median :1122 Median : 56098   
## Mean :2016-05-08 Mean :1191 Mean : 61403   
## 3rd Qu.:2017-07-11 3rd Qu.:1805 3rd Qu.: 95173   
## Max. :2018-09-13 Max. :4270 Max. :225128   
##   
## Avg\_Item\_Value\_GBP Avg\_Trans\_Value\_GBP DoW WeekNum   
## Min. :29.70 Min. : 73.14 Friday :245 Min. :1.000   
## 1st Qu.:45.85 1st Qu.:109.26 Monday :245 1st Qu.:2.000   
## Median :49.72 Median :119.30 Saturday :245 Median :4.000   
## Mean :49.17 Mean :118.33 Sunday :245 Mean :3.999   
## 3rd Qu.:52.69 3rd Qu.:127.90 Thursday :246 3rd Qu.:6.000   
## Max. :62.64 Max. :154.06 Tuesday :245 Max. :7.000   
## Wednesday:246   
## MonthNum MonthFactor ValNorm Data.x   
## Min. : 1.000 Aug :155 Min. : 1.000 Min. : 1.000   
## 1st Qu.: 3.000 Jan :155 1st Qu.: 1.490 1st Qu.: 1.490   
## Median : 6.000 Jul :155 Median : 3.079 Median : 3.079   
## Mean : 6.259 Mar :155 Mean : 3.296 Mean : 3.296   
## 3rd Qu.: 9.000 May :155 3rd Qu.: 4.679 3rd Qu.: 4.679   
## Max. :12.000 Apr :150 Max. :10.000 Max. :10.000   
## (Other):792   
## Trend.x Seasonal7 Seasonal30.5   
## Min. :0.2821 Min. :-1.362e-01 Min. :-8.251e-02   
## 1st Qu.:0.3524 1st Qu.:-2.177e-02 1st Qu.:-1.654e-02   
## Median :0.9730 Median :-3.098e-03 Median : 1.573e-04   
## Mean :0.8335 Mean :-1.859e-05 Mean : 1.566e-05   
## 3rd Qu.:1.2271 3rd Qu.: 2.618e-02 3rd Qu.: 1.507e-02   
## Max. :1.2805 Max. : 8.974e-02 Max. : 7.506e-02   
##   
## Seasonal365.25 Remainder.x PRCP TAVG   
## Min. :-0.242018 Min. :-0.4126526 Min. :0.000 Min. :-4.10   
## 1st Qu.:-0.087261 1st Qu.:-0.0475279 1st Qu.:0.000 1st Qu.: 7.90   
## Median :-0.026695 Median : 0.0043379 Median :0.000 Median :12.10   
## Mean :-0.006629 Mean : 0.0003003 Mean :0.446 Mean :12.11   
## 3rd Qu.: 0.065400 3rd Qu.: 0.0516012 3rd Qu.:1.000 3rd Qu.:16.40   
## Max. : 0.286105 Max. : 0.3221262 Max. :1.000 Max. :27.90   
## NA's :4 NA's :4   
## Price Change % PrecFact Data.y   
## Min. :1.205 Length:1717 0 :949 Min. :-4.10   
## 1st Qu.:1.311 Class :character 1 :764 1st Qu.: 7.90   
## Median :1.425 Mode :character NA's: 4 Median :12.10   
## Mean :1.440 Mean :12.11   
## 3rd Qu.:1.556 3rd Qu.:16.40   
## Max. :1.717 Max. :27.90   
## NA's :4   
## Trend.y Seasonal364 Remainder.y   
## Min. :11.60 Min. :-9.37370 Min. :-8.19060   
## 1st Qu.:11.83 1st Qu.:-4.31523 1st Qu.:-1.66520   
## Median :11.97 Median :-0.26674 Median :-0.08504   
## Mean :12.08 Mean : 0.07863 Mean :-0.04054   
## 3rd Qu.:12.17 3rd Qu.: 4.68471 3rd Qu.: 1.44230   
## Max. :13.34 Max. : 9.46271 Max. : 9.07603   
##

# Determining variable correlation using Generalized additive models with integrated smoothness estimation.

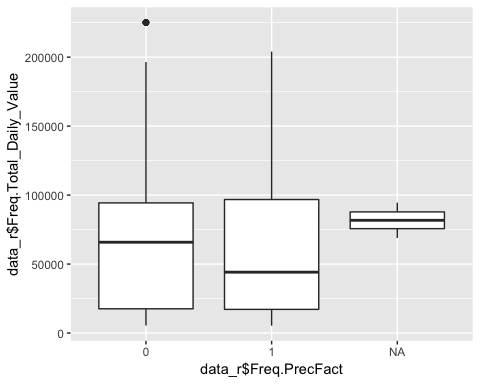
library(mgcv)  
#reading dataframe as data.table  
DT <- as.data.frame.table(newdf)  
  
#Setting variables  
  
n\_value <- unique(DT[, "Freq.Total\_Daily\_Value"])  
n\_date <- unique(DT[, "Freq.Date"])  
n\_weekdays <- unique(DT[, "Freq.WeekNum"])  
n\_dow <- unique(DT[, "Freq.DoW"])  
n\_monthFact <- unique(DT[, "Freq.MonthFactor" ])  
n\_monthn <- unique(DT[, "Freq.MonthNum" ])  
n\_temp <- unique(DT[, "Freq.TAVG"])  
n\_rain <- unique(DT[, "Freq.PRCP"])  
n\_price <- unique(DT[, "Freq.Price"])  
n\_temp\_variation <- unique(DT[, "Freq.Remainder.y"])  
week\_period <- 7  
year\_period <- 365.25  
#back-up  
data\_r <- DT

The plot below highlights the relationship between the price of the British Pound in US Dollars and Daily sales values. It is possible to see an inversed correlation, where the cheper the GBP is in comparison to the USD, the higher the sales value.



The next plot shows the relationship between Sales and how 'unusual' the temperature for a particular day was. This is done by using the remaineder element from the decomposition that was previously done on the daily weather values. The relationship does not seem strong, however it is possible to see that the highest sales numbers tend to lean towards to coldest days. 

#Precipitation close-up - needs ordering  
ggplot(data\_r, aes( data\_r$Freq.PrecFact, data\_r$Freq.Total\_Daily\_Value)) +  
geom\_boxplot()

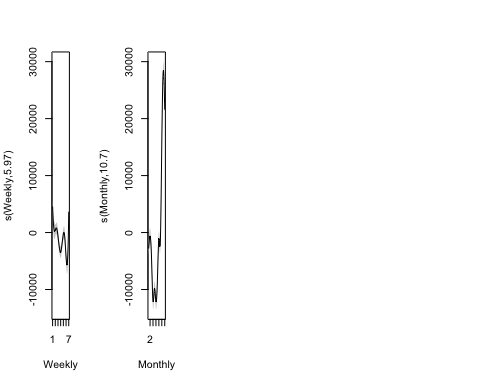


## edf Ref.df F p-value  
## s(Weekly) 5.972148 5.999565 35.70288 1.698202e-43  
## s(Monthly) 10.702601 10.980841 318.01161 0.000000e+00

The table above has shown that the Weekly and the Temperature elements are the weakest. Weekly also have the lowest edf, so it will likely be dropped from the model entirely. Below follows a summary of the model:

summary(gam\_0)

##   
## Family: gaussian   
## Link function: identity   
##   
## Formula:  
## Sales ~ s(Weekly, bs = "ps", k = week\_period) + s(Monthly, bs = "cr",   
## k = 12)  
##   
## Parametric coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 61402.9 208.6 294.4 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Approximate significance of smooth terms:  
## edf Ref.df F p-value   
## s(Weekly) 5.972 6.00 35.7 <2e-16 \*\*\*  
## s(Monthly) 10.703 10.98 318.0 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## R-sq.(adj) = 0.0793 Deviance explained = 7.97%  
## GCV = 1.8684e+09 Scale est. = 1.8676e+09 n = 42925

 - Weekly: From 1 (Monday) to 7 (Sunday): It would seem Saturdays are the lowest expected trading days, while Sunday the highest, although the p-value for this variable was too high (2.578628e-291), so this variable is not significant.

* Monthly: From 1 (Januray) to 12 (December): The highest volumes could be expected in November, the lowers in the summer months. It seems that the month of the year is by far the most important component in prediciting sales volumes in this dataset, although the price of the GBP against the USD do seem highly correlated to the growth trend in the data.
* price: Price of GBP in USD - The highest volumes were seen when the price of GBP was lowest, however there is a natural growth trend present in the data that follows the price growth closely. The key is to determine the level of intereference of either variable, or if this is just a coincidence.
* Temp (Temperature variation): This is a range of how 'off' the moving average the temperature was for a given day. It seems that lower volumes were observed for very unusually cold days, slightly up for slightly colder days, slightly down for slightly hotter days, but the spike in unusually warm days seems a difficult to explain, as indeed the p-value seems too high. Thus this variable will no longer be used in the analysis.

# Rebuilding the model to include tensor product interactions instead of the simple additive smooth functions

#including Weekly, Monthly seasonal elements with the price of the GBP and trend as tensors.  
  
gam\_0\_tensors <- gam(Sales ~ s(Weekly, Monthly)+  
 ti(price,trend),  
 data = matrix\_gam,  
 family = gaussian)  
  
summary(gam\_0\_tensors)

##   
## Family: gaussian   
## Link function: identity   
##   
## Formula:  
## Sales ~ s(Weekly, Monthly) + ti(price, trend)  
##   
## Parametric coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 90855.0 510.9 177.8 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Approximate significance of smooth terms:  
## edf Ref.df F p-value   
## s(Weekly,Monthly) 28.92 29 422.1 <2e-16 \*\*\*  
## ti(price,trend) 16.00 16 18921.2 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## R-sq.(adj) = 0.886 Deviance explained = 88.6%  
## GCV = 2.3221e+08 Scale est. = 2.3197e+08 n = 42925

We can compare with the results from the first model: gam\_0: R-sq.(adj) = 0.919 Deviance explained = 91.9% GCV = 1.6493e+08 Scale est. = 1.6475e+08 n = 42925 Meaning no improvements so far.

The GCV (Generalised Cross-Validation Score) has been reduced, which means less danger of overfitting, while the R-squared has improved (the R-squared indicated how well the model fits the data.)

#comparing the 2 models  
AIC(gam\_0, gam\_0\_tensors)

## df AIC  
## gam\_0 18.67475 1038196.0  
## gam\_0\_tensors 46.92109 948689.4

So far there has not been a significant difference on the information loss levels of either model.

Next the use of tensors will be attempted to try and generate better results. We will remove the weekly component because it has the smallest variance observed and it will speed-up the processing of the gam.

gam\_1\_tensors <- gam(Sales ~ t2(Monthly,trend,price,  
 full = TRUE),  
 data = matrix\_gam,  
 family = gaussian)  
  
summary(gam\_1\_tensors)

##   
## Family: gaussian   
## Link function: identity   
##   
## Formula:  
## Sales ~ t2(Monthly, trend, price, full = TRUE)  
##   
## Parametric coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 61402.91 53.77 1142 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Approximate significance of smooth terms:  
## edf Ref.df F p-value   
## t2(Monthly,trend,price) 109.5 98 1644 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## R-sq.(adj) = 0.939 Deviance explained = 93.9%  
## GCV = 1.2442e+08 Scale est. = 1.241e+08 n = 42925

This model seems to have been by far the most sucessful in comparison to the previous two models.

gam\_0 (Weekly, Monthly, price, Temperature,trend added, simple smoothing functions) R-sq.(adj) = 0.919 Deviance explained = 91.9% GCV = 1.6493e+08 Scale est. = 1.6475e+08 n = 42925

gam\_0\_tensors: s(Weekly, Monthly)+ ti(price,trend) - only price and trend as tensors R-sq.(adj) = 0.886 Deviance explained = 88.6% GCV = 2.3221e+08 Scale est. = 2.3197e+08 n = 42925

gam\_1\_tensors(Monthly,price,trend - all tensors and weekly effect removed) R-sq.(adj) = 0.939 Deviance explained = 93.9% GCV = 1.2442e+08 Scale est. = 1.241e+08 n = 42925

Although this is the best model so far, it is still not clear if the currency price are so highly correlated to the trend in the data by coincidence. So a few further models will be tested to try and clarify.

Since trend is not a tangible variable, it is intead a product of a combination of variables, it would be somewhat unwise to consider the trend element a predictor per se. However, since trend is a product of growth, I have tried a model that predicts the trend using price as a predictor. The result is astonishing: The R-squared for the model was 92.5% and a low GCV of 0.01. The degrees of freedom are 4, indicating a simple quadratic function.

gam\_5\_tensors <- gam(Sales ~ t2(Monthly,price,  
 full = TRUE),  
 data = matrix\_gam,  
 family = gaussian)  
summary(gam\_5\_tensors)

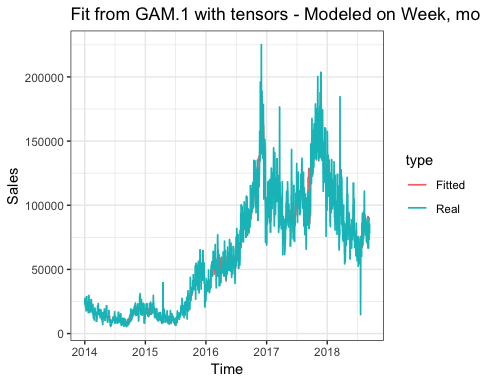
##   
## Family: gaussian   
## Link function: identity   
##   
## Formula:  
## Sales ~ t2(Monthly, price, full = TRUE)  
##   
## Parametric coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 61402.91 78.46 782.6 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Approximate significance of smooth terms:  
## edf Ref.df F p-value   
## t2(Monthly,price) 23.98 24 11954 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## R-sq.(adj) = 0.87 Deviance explained = 87%  
## GCV = 2.6437e+08 Scale est. = 2.6421e+08 n = 42925

The third result to measure is the AIC, an acronym for Akaike information criterion, which is an estimator of how well a model fits the data in comparison to other given models. The AIC measures how much information has been lost in the process of the building of the model, therefore the model with the lowest AIC should be preferred.

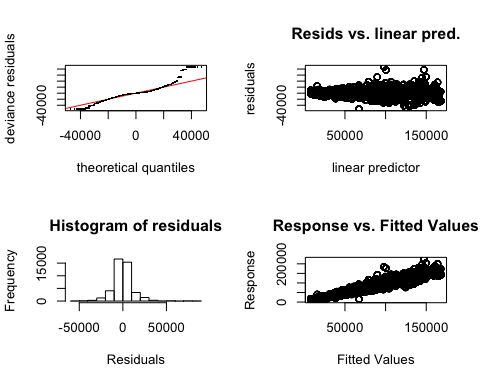
AIC(gam\_0, gam\_0\_tensors, gam\_1\_tensors,gam\_2\_tensors)

## df AIC  
## gam\_0 18.674750 1038196.0  
## gam\_0\_tensors 46.921087 948689.4  
## gam\_1\_tensors 111.453129 921904.2  
## gam\_2\_tensors 5.995821 981852.7

This comparison shows that the gam\_1\_tensors (Month, price, trend, with R-squared at 93%) has the best predictive model.

 Visualisation of the GAM with Monthly, price and trend as predictors

gam.check(gam\_1\_tensors)

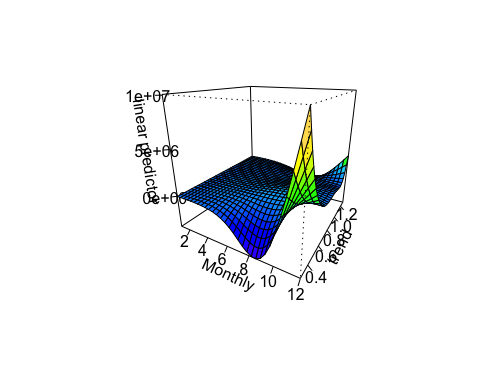


##   
## Method: GCV Optimizer: magic  
## Smoothing parameter selection converged after 86 iterations.  
## The RMS GCV score gradient at convergence was 84.43467 .  
## The Hessian was not positive definite.  
## Model rank = 125 / 125   
##   
## Basis dimension (k) checking results. Low p-value (k-index<1) may  
## indicate that k is too low, especially if edf is close to k'.  
##   
## k' edf k-index p-value   
## t2(Monthly,trend,price) 124 110 0.18 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Of course, the model's reliability comes at the cost of interpretability - and, despite a low GCV score, still a danger of overfitting.

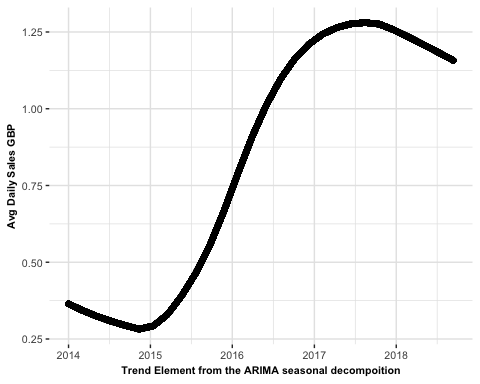
## 3D view of fitted values from the best GAM

vis.gam(gam\_1\_tensors,  
 ticktype = "detailed", color = "topo", theta= 30, phi=20)

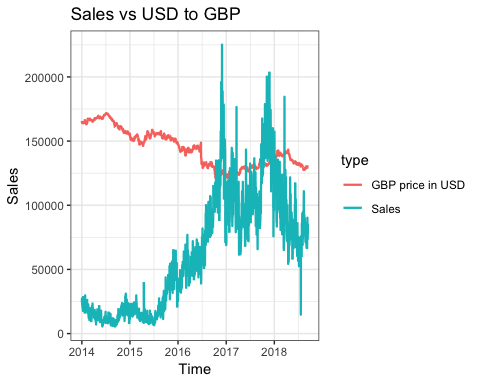
 The 3D graph above shows that the top of sales is when the currency is cheaper, but the peak in sales in July, with the currency price at its highest, at the back of the graph, is difficult to justify. It seems this model is overfitted. This is why other models will be contructed to find a better component explanation.

Simon N. Wood's mcgv library, which contains the gam model and all its calculations, has a function to identify each components variance, although for a tensor product smooths such as this, the author warns, interpretation is not so straight-forward. Below is the code that will return a covariance matrix. Each smoothing parameters could have been estimated + fixed or replicated (it is not clear, but I believe the r below means replicated). On the same scale, components in smoothing parameters are related to variance components.

## Plotting the actual trend returned by the arima decomposition

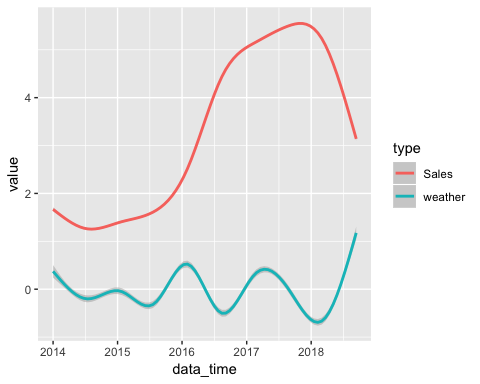


# Plotting sales values against currency prices (price of GBP in USD)



# Plotting sales values and weather

## `geom\_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'

 The plot above indicates that 2016 had a somewhat unusually warmer winter and a colder summer, while 2018 had a unusually cold first quarter (Januray, February, March) followed by an unusually warm summer. Indeed there are news reports about the cold spell which blasted the UK in February 2018, named 'The Beast From The East' by the media, followed by an unusually hot summer. This unusually warm spell seems faintely related to a sales dip over the same period.

Because the initial aim of this project was to investigate a possible effect of weather on sales volumes, we shall run a GAM to predict Sales on temperature (noise only) and evaluate.

gam\_weather <- gam(Sales ~s(Temp, bs="cs"),   
 data = matrix\_gam,  
 family = gaussian)  
summary(gam\_weather)

##   
## Family: gaussian   
## Link function: identity   
##   
## Formula:  
## Sales ~ s(Temp, bs = "cs")  
##   
## Parametric coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 61402.9 215.9 284.4 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Approximate significance of smooth terms:  
## edf Ref.df F p-value   
## s(Temp) 8.927 9 65.36 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## R-sq.(adj) = 0.0134 Deviance explained = 1.36%  
## GCV = 2.0019e+09 Scale est. = 2.0015e+09 n = 42925

The deviance explained by this model, which ignores the principal component of Montly seasonality, is unsurprisingly low, but it is not insignificant. Once of the challenges with this model is that, one can assume, weather variations have different significante over different times of the year. A warm, sunny day that follows a week of rain would, one could assume, produce a more remarked effect than a whole week of sunny weather. This rationale would lead to a much more complex model that is, at the time of writing, beyond my capabilities.

The best that can be done at this time would be to add the Monthly and weekly seasonal components to the model, which are the most significant component of the analysis so far, and hope that a combination of factors that can explain the deviance of the data better will be found.

As Wood et al (2016) recommends in a detailed document on how to use GAMs, it is best to use splines for the components which are known to have constant splines and imilar scales, which Monthly and Weekly components do (as seen by the STL seasonal decomposition). As for the Temperature component, this is not a cyclic component since only the remainder of the decomposition is being used (to identify unusually warm and unusually cold days), therefore not a smooth spline.

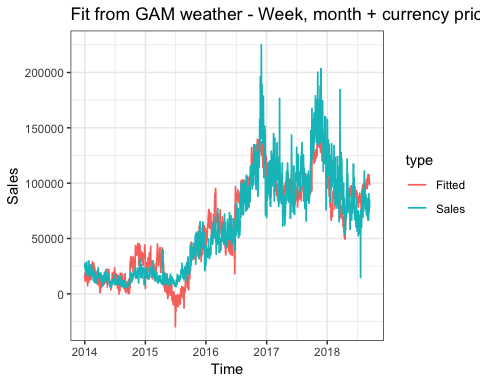
As for the currency price, although the observed data suggests a linear trend, it is known that foreign exchange rates are volatile and that it can be influenced by hundreds of factors, including the prospect of the UK leaving the European Union, the Economic climate at each of the given countries, etc, meaning it is not possible to predict future rates without also predicting the future. Given the above circumstances and Simon Woods advice, yet another GAM model will be build with cyclic variables (Month, Week) as splines and non-cyclical variables as tensors.

gam\_weather <- gam(Sales ~s(Monthly, bs= "ps", k=12) +   
 s(Weekly, bs = "ps", k = week\_period)   
 + ti(Temp, bs="cs") +ti(price),   
 data = matrix\_gam,  
 family = gaussian)  
summary(gam\_weather)

##   
## Family: gaussian   
## Link function: identity   
##   
## Formula:  
## Sales ~ s(Monthly, bs = "ps", k = 12) + s(Weekly, bs = "ps",   
## k = week\_period) + ti(Temp, bs = "cs") + ti(price)  
##   
## Parametric coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 61402.91 84.66 725.3 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Approximate significance of smooth terms:  
## edf Ref.df F p-value   
## s(Monthly) 10.987 11 2269.9 <2e-16 \*\*\*  
## s(Weekly) 5.993 6 214.2 <2e-16 \*\*\*  
## ti(Temp) 3.742 4 213.5 <2e-16 \*\*\*  
## ti(price) 3.998 4 54005.7 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## R-sq.(adj) = 0.848 Deviance explained = 84.8%  
## GCV = 3.0787e+08 Scale est. = 3.0769e+08 n = 42925

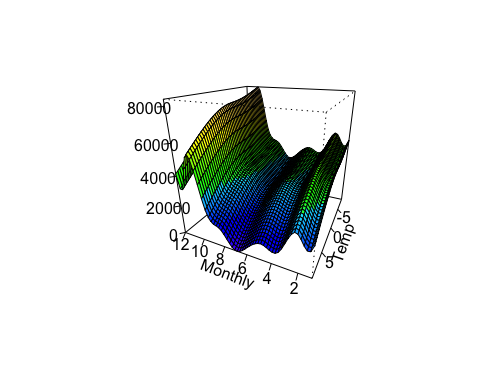
This is not the best model so far, but it is a model that can be interpreted. The trend element has been left out of the equation due to its extreme correlation to the currency price, and also to prevent overfitting. We are now left with a score for each component. The edf (Estimated degress of freedom) for each component are managable (close to a quadratic function for both Temp and price).

# Plotting the latest GAM's fitted values against real values

 This model is not as accurate as the one before, but it is easier to interpret. It explains 84% of the deviance of the data, leaving room for other unkown factors, such as competitor activity, special events such as football's main event, The World Cup, The Olympic Games, and so on.

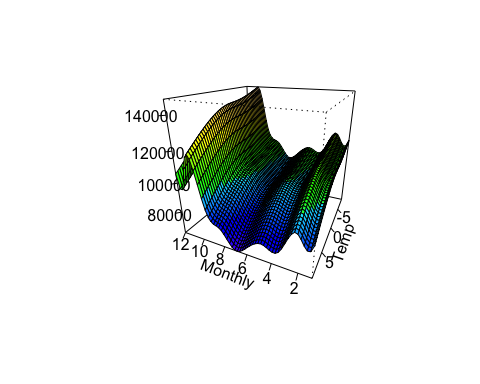
## 3D plot of the GAM above with Month of the year, Temperature Variation and setting the price of GBP to 1.75 against USD (which is the most expensive we have ever seen)

vis.gam(gam\_weather, view=c("Monthly","Temp"), cond=list(price=1.75), n.grid = 50, theta =-155, phi = 22, zlab = "",  
 ticktype = "detailed", color = "topo")



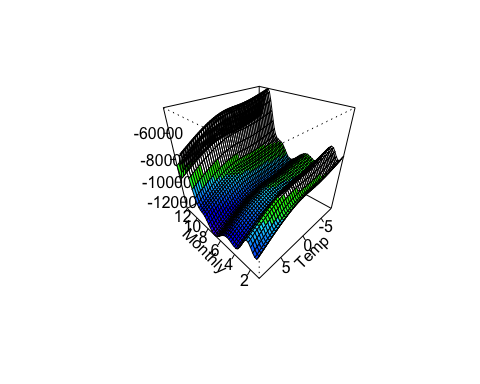
## 3D plot of the same model, but setting the price value of GBP to 1.30 (Current exchange rate as of Sep 2018)

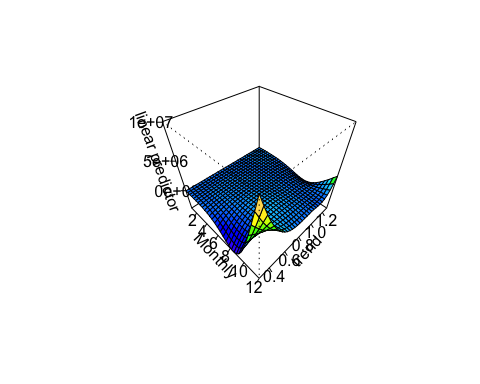
vis.gam(gam\_weather, view=c("Monthly","Temp"), cond=list(price=1.30), n.grid = 50, theta =-155, phi = 22, zlab = "",  
 ticktype = "detailed", color = "topo")



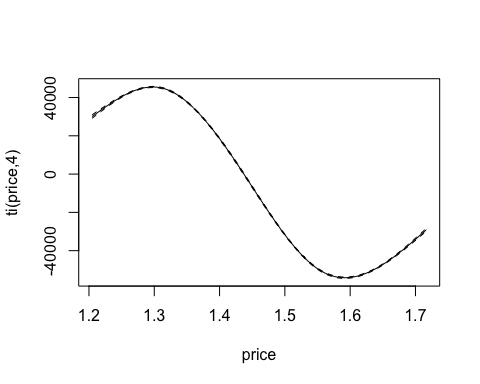
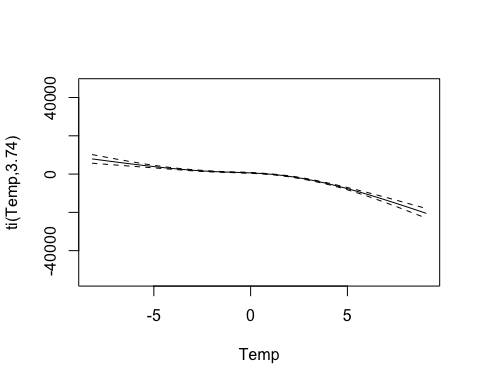
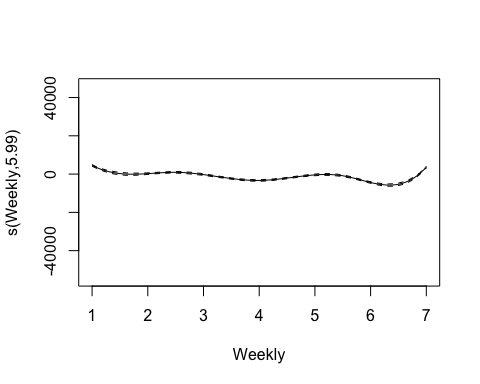
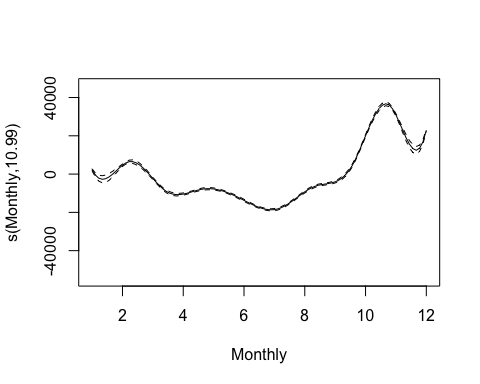
It is worth noticing that the currency exchange rate smooth generated by the model was a spline, which does not make much sense in the real world. It fits the current model, but it gives the wrong predictions for price values that goes outside the current model. For example, the tendency observed is for sales to increase if the value of GBP decreases against the dollar, but if we try fitting values that go below 1.2, it starts predicting negative values. Likewise, if the values are too far above 1.7, it starts predicting very high sales values. This is because the sales values have confounded the model, which has returned an overfitting. For example, if we try setting the price of the GBP to 0.99, the results indicate lower sales than if the price was 1.3O, and if we lower this amount to 0.50 it goes as far as inverting the predicted values to return negative sale value. On the other hand, if we set the price to 2.75, it increases the daily sales value, which would not be expected. ##3D plot for the GAM with the price set to 0.50

vis.gam(gam\_weather, view=c("Monthly","Temp"), cond=list(price=0.5),n.grid = 50, theta =-135, phi = 32, zlab = "",  
 ticktype = "detailed", color = "topo")



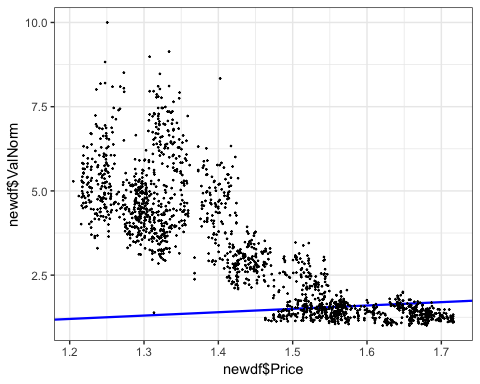


plot(gam\_weather)

 Unlinke traditional regression models, it is not possible to interpret any coefficients or express the estimate curve by a formula.

Finally, I'd like to try a simple linear regression on currency price and sales (sales after decomposition, so only the reminder)

ggplot(data = newdf, aes(newdf$Price, newdf$ValNorm) )+  
 geom\_abline(color = "blue", size = 0.8)+  
 geom\_point(size = 0.2) +  
 theme\_bw()

 It seems the linear model is being confused by a trend found when the curency price was between 1.5 and 1.7, with the data elsewhere rather sparse. There must be a different formula for measuring this type of situation which does not rely on splines (since it has been observed this model can return very strange results). More research would be needed to find this method.

Finally, what this project was able to conclude was that sales are heavily influenced by the yearly seasonality, which can explain the variance of the data by almost 8% of the variance (As given by our initial gam\_0).

The other variables collected did seem to be somewhat correlated to the Sales volumes observed in the data, however the effect of each variable on its own was not found to be measurable with the tools that have been used in this project.

The best predictor of future values was the ARIMA model with seasonality decomposition. It returned the lowest AIC of all models, by far, of 36479.07 and relatively low training set measures:

ME RMSE MAE MPE MAPE MASE ACF1

Training set 199.8866 9819.821 6481.4 -1.678859 14.91504 0.2069251 0.001958019

Run the code below for a comparison of the models' AIC used in this project.

AIC(fitwy, gam\_0, gam\_1\_tensors, gam\_weather)

## df AIC  
## fitwy 30.00000 36479.07  
## gam\_0 18.67475 1038195.99  
## gam\_1\_tensors 111.45313 921904.24  
## gam\_weather 26.71919 960795.49

If there were more time and resources available for this project to be completed, future improvements would certainly include: - Finding a better statistical model to explain variance by different factors. - Support for multiple currencies - User-interface Support for visualization of the predicted value of sales at a certain date by providing certain variables - Add a user interface such as R-Shiny or similar - Build a relational database for the data collected and processed, such as MySQL. - Use python for data manipulation and analysis.