JWashburn_Homework12

Jeff Washburn

2/18/2019

1. Brief Financial Data

1a. EuStockMarkets

Natively in R, you have access to sample data sets of prominent stocks over time. We'll be using EuStockMarkets for this question. Type help(EuStockMarkets) to learn more. From these data, pull specifically the DAX index. For all questions in this assignment, you're welcome to normalize (or don't!) how you see fit, but, if you choose to, please document what you're doing and why for the grader. It's not necessary for the purpose of this assignment.

```
data("EuStockMarkets")
str(EuStockMarkets)
```

```
##  Time-Series [1:1860, 1:4] from 1991 to 1999: 1629 1614 1607 1621 1618 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:4] "DAX" "SMI" "CAC" "FTSE"
```

```
is.ts(EuStockMarkets)
```

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

```
head(time(EuStockMarkets))
```

```
## [1] 1991.496 1991.500 1991.504 1991.508 1991.512 1991.515
```

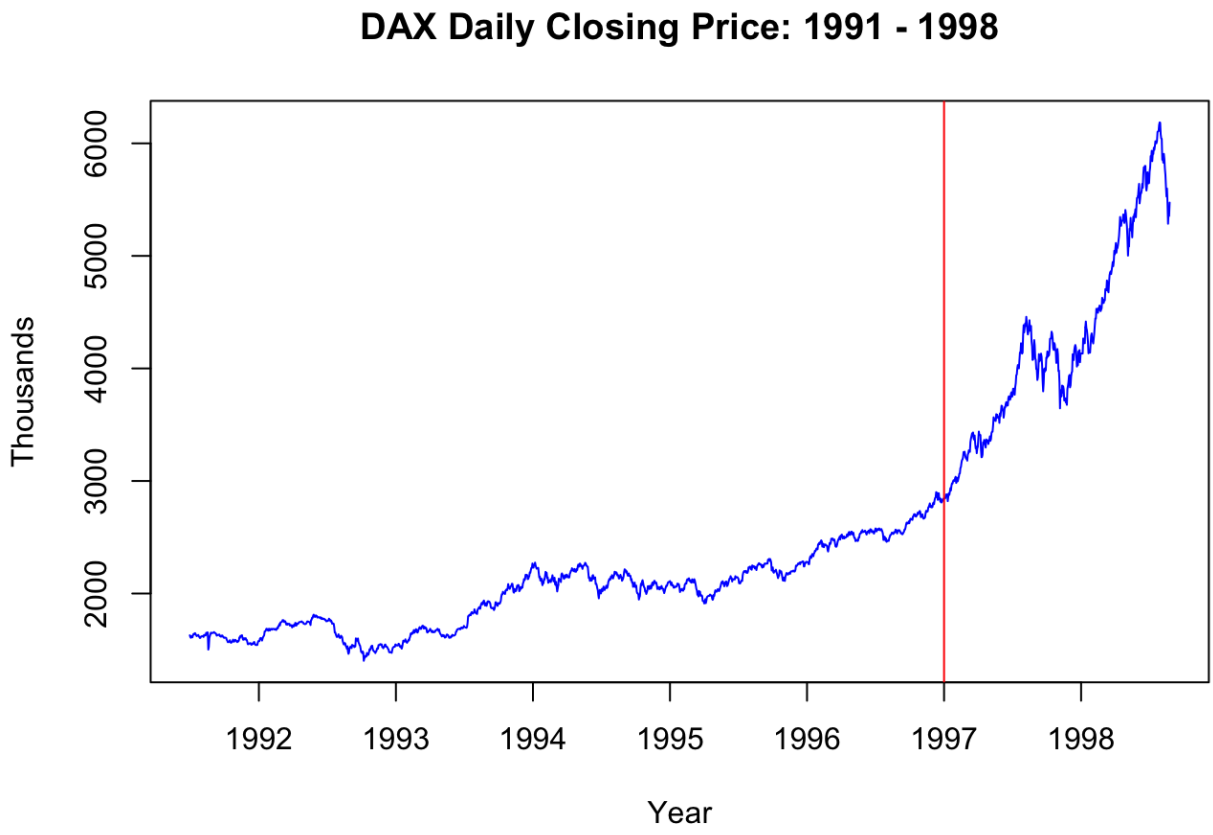
```
tail(time(EuStockMarkets))
```

```
## [1] 1998.627 1998.631 1998.635 1998.638 1998.642 1998.646
```

1b. Plot the data

These are annual European Stock Data from 1990 onward. Create a rudimentary plot of the data. Make the line blue. Give an informative title. Label the axes accurately. In 1997, an event happened you want to indicate; add a vertical red line to your plot which divides pre-1997 and post-1997 information

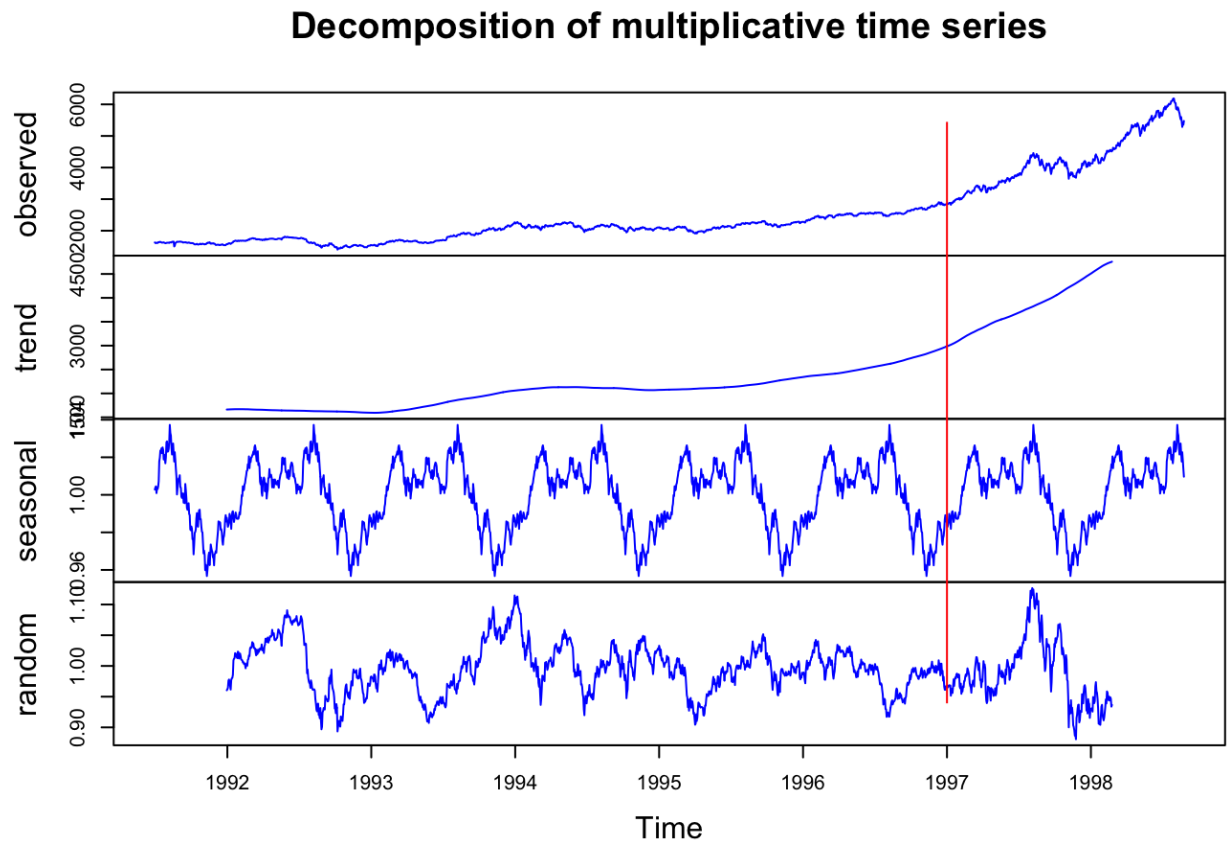
```
#plot(EuStockMarkets[])
plot(EuStockMarkets[,"DAX"], main="DAX Daily Closing Price: 1991 - 1998", xlab=
  "Year", ylab="Thousands", col="blue")
abline(v=1997, col="red")
```



1c. Decomponse time series

Decompose the time series into its components (i.e., trend, seasonality, random). Keep in mind that this is a multiplicative model you want. Create a plot of all decomposed components. As before, make all lines blue and have a vertical divider at the year 1997.

```
dax.decompose = decompose(EuStockMarkets[,"DAX"], type="mult")
plot(dax.decompose, col="blue")
abline(v=1997, col="red")
```



2. Temperature Data

2a. maxtemp dataset

Using the maxtemp dataset granted by loading fpp2, there are maximum annual temperature data in Celsius. For more information, use help(maxtemp). To see what you're looking at, execute the command in 'Examples' in the help document.

```
library(fpp2)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: forecast
```

```
## Loading required package: fma
```

```
## Loading required package: expsmooth
```

```
data("maxtemp")
str(maxtemp)
```

```
##  Time-Series [1:46] from 1971 to 2016: 34.6 39.3 40.5 36.8 39.7 40.5 41.5 3
8.2 41.4 41.5 ...
##  - attr(*, "names")= chr [1:46] "1971" "1972" "1973" "1974" ...
```

```
is.ts(maxtemp)
```

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

```
head(time(maxtemp))
```

```
## Time Series:
## Start = 1971
## End = 1976
## Frequency = 1
## [1] 1971 1972 1973 1974 1975 1976
```

```
tail(time(maxtemp))
```

```
## Time Series:
## Start = 2011
## End = 2016
## Frequency = 1
## [1] 2011 2012 2013 2014 2015 2016
```

2b. Eliminate unwanted data

We are only concerned with information after 1990. Please eliminate unwanted information or subset information we care about.

```
maxtemp.after1990 = window(maxtemp, start=1990)
head(time(maxtemp.after1990))
```

```
## Time Series:
## Start = 1990
## End = 1995
## Frequency = 1
## [1] 1990 1991 1992 1993 1994 1995
```

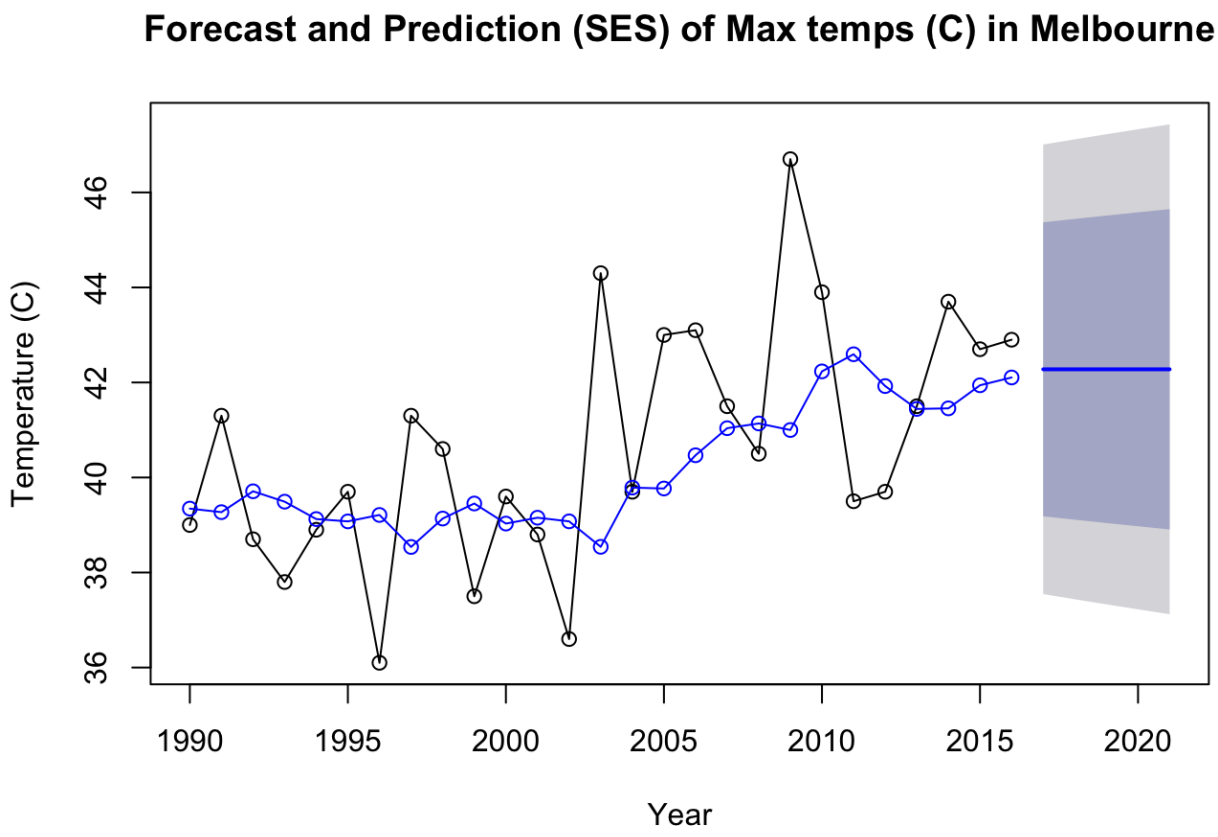
```
tail(time(maxtemp.after1990))
```

```
## Time Series:
## Start = 2011
## End = 2016
## Frequency = 1
## [1] 2011 2012 2013 2014 2015 2016
```

2c. Utilize SES to predict

Utilize SES to predict the next five years of maximum temperatures in Melbourne. Plot this information, including the prior information and the forecast. Add the predicted value line across 1990-present as a separate line, preferably blue. So, to review, you should have your fit, the predicted value line overlaying it, and a forecast through 2021, all on one axis. Find the AICc of this fitted model. You will use that information later.

```
maxtemp.after1990.fit <- ses(maxtemp.after1990, h=5)
plot(maxtemp.after1990.fit, plot.conf = FALSE, ylab="Temperature (C)", xlab="Ye
  ar", main="Forecast and Prediction (SES) of Max temps (C) in Melbourne", fcol
  ="blue", type="o")
lines(fitted(maxtemp.after1990.fit), col="blue", type="o")
```



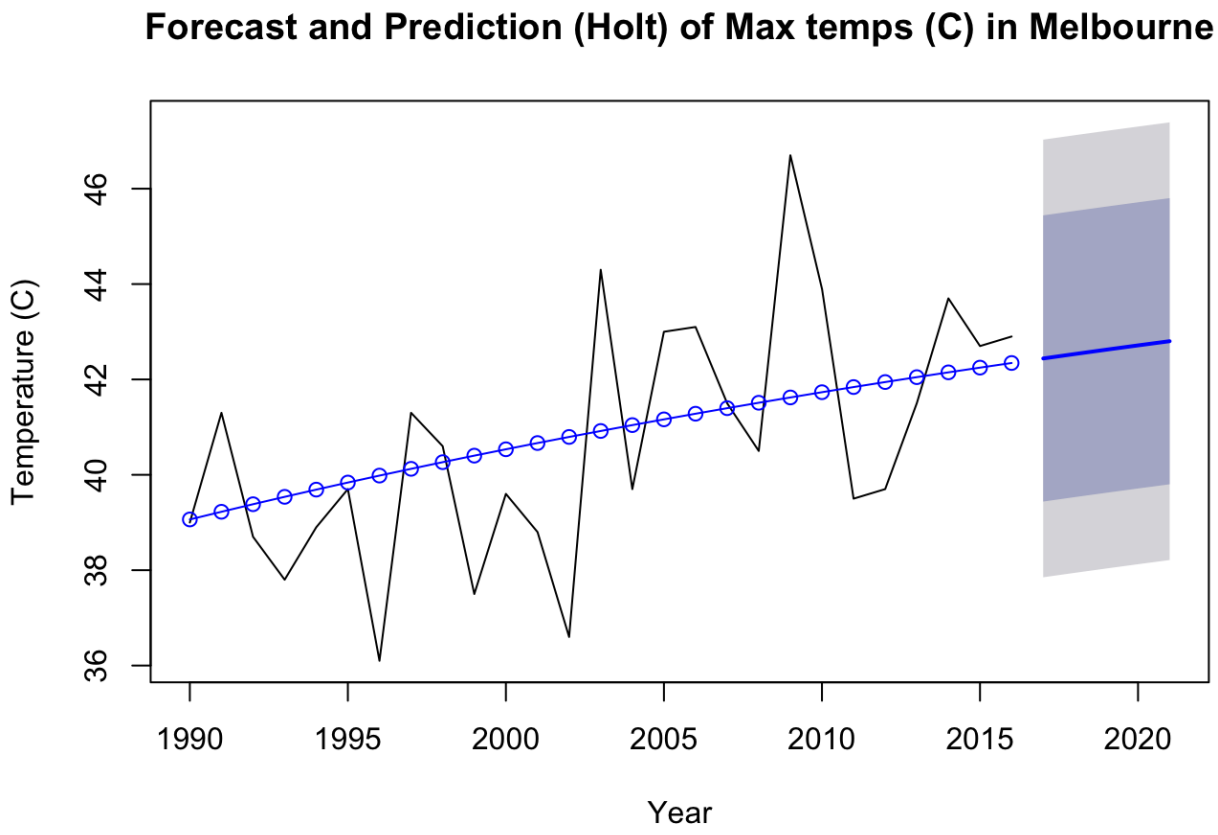

```
# show the model to get the AIC
maxtemp.after1990.fit$model
```

```
## Simple exponential smoothing
##
## Call:
##  ses(y = maxtemp.after1990, h = 5)
##
##   Smoothing parameters:
##     alpha = 0.2164
##
##   Initial states:
##     l = 39.345
##
##   sigma:  2.4135
##
##      AIC     AICc      BIC
## 140.4868 141.5302 144.3743
```

2d. Damped Holt's Linear trend

Now use a damped Holt's linear trend to also predict out five years. Make sure initial="optimal." As above, create a similar plot to 1C, but use the Holt fit instead.

```
maxtemp.after1990.holtfit <- holt(maxtemp.after1990, initial="optimal", h=5, da
  mped=TRUE)
plot(maxtemp.after1990.holtfit, ylab="Temperature (C)", xlab= "Year", main="For
  ecast and Prediction (Holt) of Max temps (C) in Melbourne")
lines(fitted(maxtemp.after1990.holtfit), col="blue", type="o")
```




```
# show the AIC
maxtemp.after1990.holtfit$model
```

```
## Damped Holt's method
##
## Call:
##  holt(y = maxtemp.after1990, h = 5, damped = TRUE, initial = "optimal")
##
##   Smoothing parameters:
##     alpha = 1e-04
##     beta  = 1e-04
##     phi   = 0.98
##
##   Initial states:
##     l = 38.8993
##     b = 0.1678
##
##   sigma:  2.3409
##
##      AIC     AICc      BIC
## 141.3865 145.5865 149.1615
```

2e. Compare AICs

Compare the AICc of the ses() and holt() models. Which model is better here?

The ses model has a lower AICc score (141.53) than the holt model score (145.59) and therefore, the ses model is slightly better. Lower indicates a more parsimonious model (https://www.r-bloggers.com/how-do-i-interpret-the-aic/ (https://www.r-bloggers.com/how-do-i-interpret-the-aic/))

3. Wands Choose the Wizard

3a. Dygraphs library

Utilize the dygraphs library. Read in both Unit11TimeSeries_Ollivander and _Gregorovitch.csv as two different data frames. They do not have headers, so make sure you account for that. This is a time series of Wands sold over years

```
library(dygraphs)
data.ollivander = read.csv("./Unit11TimeSeries_Ollivander.csv", header=FALSE)
summary(data.ollivander)
```

```
##         V1           V2
##  1/1/1970: 1   Min.   : 200
##  1/1/1971: 1   1st Qu.:1258
##  1/1/1972: 1   Median :1312
##  1/1/1973: 1   Mean   :1325
##  1/1/1974: 1   3rd Qu.:1452
##  1/1/1975: 1   Max.   :1797
##  (Other) :42
```

```
names(data.ollivander) = c("Date", "WandsSold_O")
head(data.ollivander)
```

```
##       Date WandsSold_O
## 1 1/1/1970        1345
## 2 1/1/1971        1304
## 3 1/1/1972        1168
## 4 1/1/1973        1252
## 5 1/1/1974        1296
## 6 1/1/1975        1458
```

```
str(data.ollivander)
```

```
## 'data.frame':    48 obs. of  2 variables:
##  $ Date       : Factor w/ 48 levels "1/1/1970","1/1/1971",..: 1 2 3 4 5 6 7
8 9 10 ...
##  $ WandsSold_O: int  1345 1304 1168 1252 1296 1458 1443 1282 1450 1338 ...
```

```
data.gregorovitch = read.csv("./Unit11TimeSeries_Gregorovitch.csv", header=FALS
  E)
summary(data.gregorovitch)
```

```
##         V1           V2
##  1/1/1970: 1   Min.   :   0
##  1/1/1971: 1   1st Qu.: 800
##  1/1/1972: 1   Median :1189
##  1/1/1973: 1   Mean   :1049
##  1/1/1974: 1   3rd Qu.:1431
##  1/1/1975: 1   Max.   :1863
##  (Other) :42
```

```
names(data.gregorovitch) = c("Date", "WandsSold_G")
head(data.gregorovitch)
```

```
##       Date WandsSold_G
## 1 1/1/1970        1268
## 2 1/1/1971        1295
## 3 1/1/1972        1349
## 4 1/1/1973        1298
## 5 1/1/1974        1493
## 6 1/1/1975        1432
```

```
str(data.gregorovitch)
```

```
## 'data.frame':    48 obs. of  2 variables:
##  $ Date       : Factor w/ 48 levels "1/1/1970","1/1/1971",..: 1 2 3 4 5 6 7
8 9 10 ...
##  $ WandsSold_G: int  1268 1295 1349 1298 1493 1432 1431 1291 1247 1403 ...
```

3b. Convert dates

You don’t have your information in the proper format! In both data sets, you’ll need to first convert the date-like variable to an actual Date class.

```
# update the date on data.ollivander
data.ollivander$Date <- as.Date(data.ollivander$Date, "%m/%d/%Y")
str(data.ollivander)
```

```
## 'data.frame':    48 obs. of  2 variables:
##  $ Date       : Date, format: "1970-01-01" "1971-01-01" ...
##  $ WandsSold_O: int  1345 1304 1168 1252 1296 1458 1443 1282 1450 1338 ...
```

```
# update the date on data.gregorovitch
data.gregorovitch$Date <- as.Date(data.gregorovitch$Date, "%m/%d/%Y")
str(data.gregorovitch)
```

```
## 'data.frame':    48 obs. of  2 variables:
##  $ Date       : Date, format: "1970-01-01" "1971-01-01" ...
##  $ WandsSold_G: int  1268 1295 1349 1298 1493 1432 1431 1291 1247 1403 ...
```

3c. Library xts

Use the library xts (and the xts() function in it) to make each data frame an xts object (effectively, a time series). You'll want to order.by the Date variable.

```
library(xts)

# handle the ollivander data
data.ollivander.xts <- xts(data.ollivander[,-1], order.by = data.ollivander$Dat
  e)
str(data.ollivander.xts)
```

```
## An 'xts' object on 1970-01-01/2017-01-01 containing:
##   Data: int [1:48, 1] 1345 1304 1168 1252 1296 1458 1443 1282 1450 1338 ...
##   Indexed by objects of class: [Date] TZ: UTC
##   xts Attributes:
##  NULL
```

```
# handle the gergorovitch data
data.gregorovitch.xts <- xts(data.gregorovitch[,-1], order.by = data.gregorovit
  ch$Date)
str(data.gregorovitch.xts)
```

```
## An 'xts' object on 1970-01-01/2017-01-01 containing:
##   Data: int [1:48, 1] 1268 1295 1349 1298 1493 1432 1431 1291 1247 1403 ...
##   Indexed by objects of class: [Date] TZ: UTC
##   xts Attributes:
##  NULL
```

3d. Bind 2 xts objects

Bind the two xts objects together and create a dygraph from it. Utilize the help() index if you're stuck

- Give an effective title and x/y axes
- Label each Series (via dySeries) to be the appropriate wand-maker. So, one line should create a label for Ollivander and the other for Gregorovitch.
- Stack this graph and modify the two lines to be different colors (and not the default ones!) Any colors are fine, but make sure they're visible and that Ollivander is a different color than Gregorovitch.

- Activate a range selector and make it big enough to view.
- Use dyShading to illuminate approximately when Voldemort was revived and at-large: between 1995 to 1999.
- Enable Highlighting on the graph, so mousing over a line bolds it.

```
library(dygraphs)
library(RColorBrewer)
data <- merge(data.ollivander.xts, data.gregorovitch.xts, all = TRUE)
dygraph(data, main = "Wands Sold", xlab = "Time", ylab = "Number of Wands") %>%
  dySeries("data.ollivander.xts", label = "Ollivander") %>%
  dySeries("data.gregorovitch.xts", label = "Gregorovitch") %>%
  dyOptions(colors = brewer.pal(2, "Set1")) %>%
  dyShading(from = "1995-1-1", to = "1999-1-1", color = "#9FE6F5") %>%
  # dyOptions(stackedGraph = TRUE) %>%
  dyHighlight(highlightSeriesOpts = list(strokeWidth = 2)) %>%
  dyRangeSelector(height = 80)
```