Harmonic Regression

Austin Stephen

4/13/2022

TLDR

- I implemented harmonic regression by hand.
- It did not work very well but generates some interesting plots. This is probably because I don't know how to find a good period just looking at the data and the data changes a lot.
- The last plot shows performance on the test set.
- Next step for me is to write a function that adds together the predictions from a harmonic regression like model and the removed trend from the regression.
- Another next step is for me to implement a different algorithm.

Formulation:

$X_t = s_t + Y$ where s_t is a periodic function and Y is the noise term

$s_t = a_0 + \Sigma_{j=1}^{k}(a_j cos(\lambda_j t) + b_j sin(\lambda_j t))$

$a_1, a_2, ...a_n$ and $b_1, b_2, .., b_n$ are unknown parameters. $\lambda_1, .., \lambda_k$ are fixed frequencies as integer multiples of $2\pi/d$ where d is the period.

A harmonic regression model allows for noise but no trend component. As a result, it is trained on the residuals of the multiple linear regression model used to subtract out the trend in the data. I fit the a_j and b_j components using the least squares method with the built in function in R.

The number of terms k (indexing the summation) are refered to as the k-integer valued fourier components. I build a model with 1 fourier component and another with 2. Each fourier component must be a fixed integer multiple of f_1 which denotes the first component.

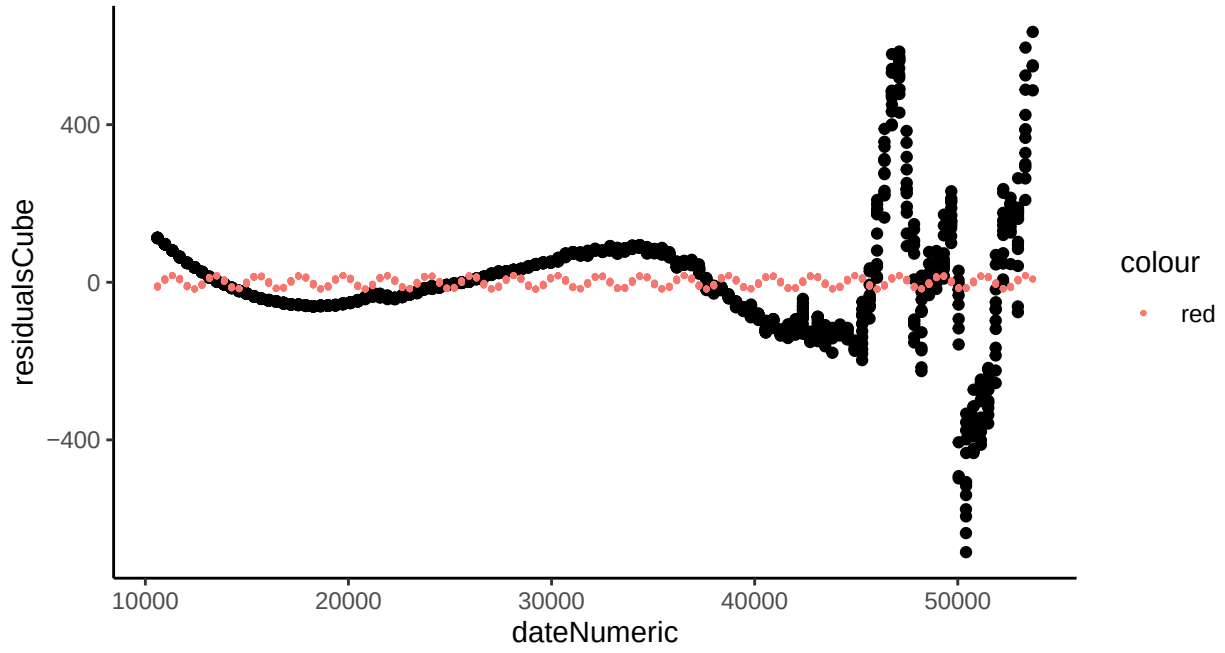
All of the math comes from Introduction to Time Series and Forecasting by Brockwell and Davis. Chapter 1.3 "Some simple models".

One fourier component

$s_t = a_0 + (a_1 cos(\lambda_j t) + b_1 sin(\lambda_j t))$

```
## one fourier component
# Difficult to infer lambda based on our data
pi = 3.14159265
period = 200
lambdaTerm <- 2* pi / period
```

```r
data <- data %>% mutate(aTerm = sin(lambdaTerm * dateNumeric),
                        bTerm = cos(lambdaTerm * dateNumeric))

model <- lm(residualsCube ~ aTerm + bTerm, data = data)

#summary(model)

# showing what the model predicts
tmp <- data.frame( aTerm = sin(lambdaTerm * data$dateNumeric),
                   bTerm = cos(lambdaTerm * data$dateNumeric),
                   dateNumeric = data$dateNumeric)

tmp$predictions <- predict(model,tmp)

data %>% ggplot(aes(x=dateNumeric,y=residualsCube))+
  geom_point()+
  geom_point(aes(y = tmp$predictions, color = "red"), size = .5)+
  theme_classic()
```

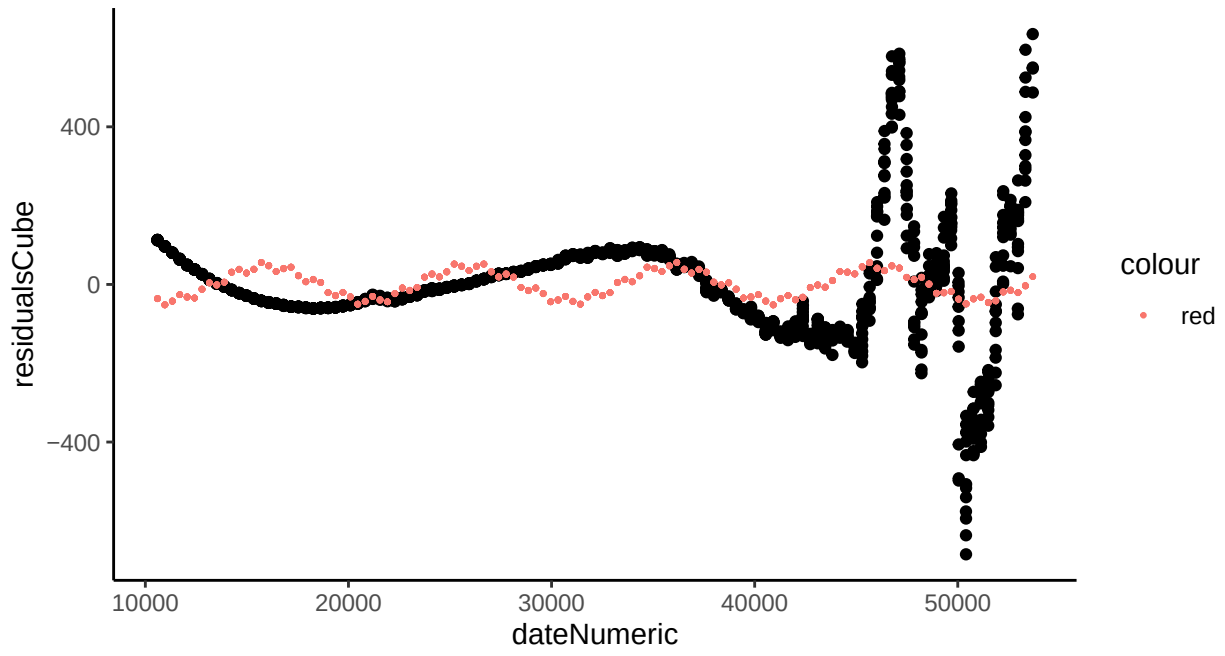


Two Fourier Components

$s_t = a_0 + (a_1 cos(\lambda_j t) + b_1 sin(\lambda_j t)) + (a_2 cos(\lambda_j t) + b_2 sin(\lambda_j t))$

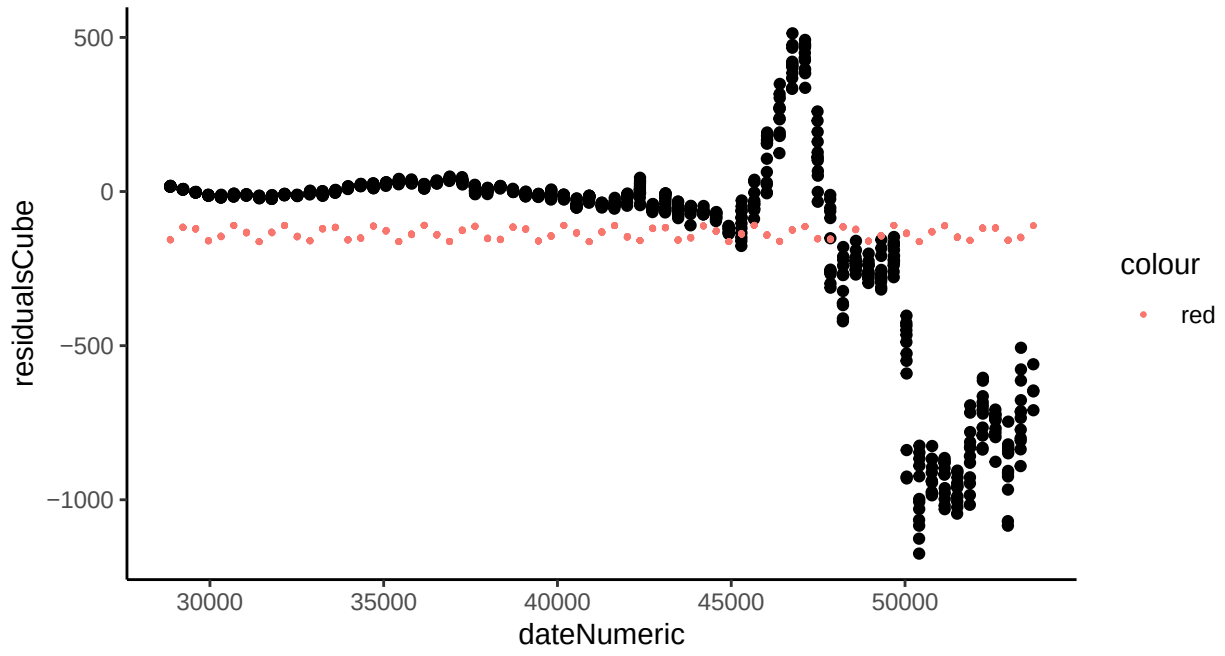
```r
## two fourier components
# Difficult to infer lambda based on out data
period = 500

lambdaTerm1 <- 2* pi / period

## fixed integer multiple
```

```r
lambdaTerm2 <- 2* pi / (period* 20)


data2 <- data %>% mutate(a1Term = sin(lambdaTerm1 * dateNumeric),
                         a2Term = sin(lambdaTerm2 * dateNumeric),
                         b1Term = cos(lambdaTerm1 * dateNumeric),
                         b2Term = cos(lambdaTerm2 * dateNumeric)
                  )

model2 <- lm(residualsCube ~ a1Term + b1Term + a2Term + b2Term, data = data2)
#summary(model)

# showing what the model predicts
tmp2 <- data.frame(a1Term = sin(lambdaTerm1 * data2$dateNumeric),
                   a2Term = sin(lambdaTerm2 * data2$dateNumeric),
                   b1Term = cos(lambdaTerm1 * data2$dateNumeric),
                   b2Term = cos(lambdaTerm2 * data2$dateNumeric),
                   dateNumeric = data2$dateNumeric)

tmp2$predictions <- predict(model2,tmp2)

data %>% ggplot(aes(x=dateNumeric,y=residualsCube))+
  geom_point()+
  geom_point(aes(y = tmp2$predictions, color = "red"),size = .5)+
  theme_classic()
```



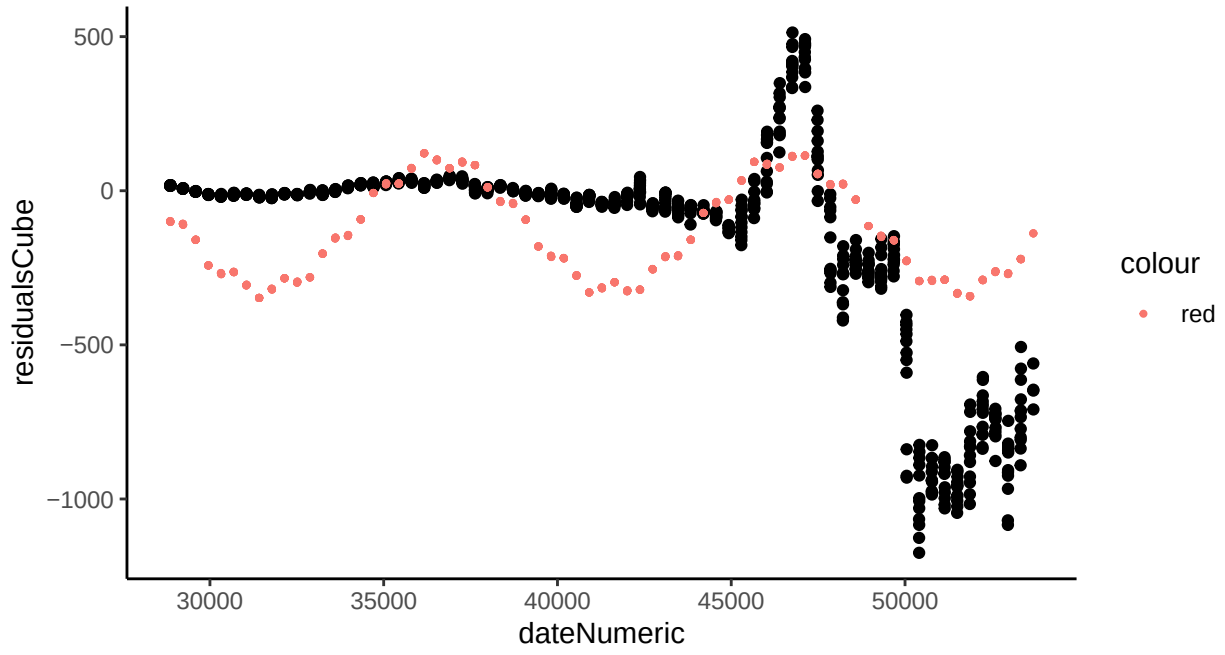
Same thing but with data from 1950 on

```
period = 500

lambdaTerm <- 2* pi / period

data_1950 <- data_1950 %>% mutate(aTerm = sin(lambdaTerm * dateNumeric),
                                  bTerm = cos(lambdaTerm * dateNumeric))

model <- lm(residualsCube ~ aTerm + bTerm, data =data_1950 )
#summary(model)

# showing what the model predicts
tmp <- data.frame( aTerm = sin(lambdaTerm * data_1950$dateNumeric),
                   bTerm = cos(lambdaTerm * data_1950$dateNumeric),
                   dateNumeric = data_1950$dateNumeric)

tmp$predictions <- predict(model,tmp)
data_1950 %>% ggplot(aes(x=dateNumeric,y=residualsCube))+
  geom_point()+
  geom_point(aes(y = tmp$predictions, color = "red"),size = .5)+
  theme_classic()
```




```
## 2 fourier components
lambdaTerm1 <- 2* pi / period

## fixed integer multiple
lambdaTerm2 <- 2* pi / (period* 20)


data_1950 <- data_1950 %>% mutate(
                              a1Term = sin(lambdaTerm1 * dateNumeric),
                              a2Term = sin(lambdaTerm2 * dateNumeric),
```

```
                              b1Term = cos(lambdaTerm1 * dateNumeric),
                              b2Term = cos(lambdaTerm2 * dateNumeric)
                  )

model2 <- lm(residualsCube ~ a1Term + b1Term + a2Term + b2Term, data = data_1950)
#summary(model)

# showing what the model predicts
tmp2 <- data.frame(a1Term = sin(lambdaTerm1 * data_1950$dateNumeric),
                   a2Term = sin(lambdaTerm2 * data_1950$dateNumeric),
                   b1Term = cos(lambdaTerm1 * data_1950$dateNumeric),
                   b2Term = cos(lambdaTerm2 * data_1950$dateNumeric),
                   dateNumeric = data_1950$dateNumeric)

tmp2$predictions <- predict(model2,tmp2)

## plot curve -> add the curve
data_1950 %>% ggplot(aes(x=dateNumeric,y=residualsCube))+
  geom_point()+
  geom_point(aes(y = tmp2$predictions, color = "red"),size = .75)+
  theme_classic()
```



Train/test analysis

```
## train on 1950-2005 -> predict 2005-current 16 years
train <- data_1950 %>% filter(dateNumeric < 48943)
test <- data_1950 %>% filter(dateNumeric >= 48943)
data_1950 <- data_1950 %>%
  mutate ( test = case_when(
```

```
    dateNumeric >= 48588 ~ 1,
    TRUE ~ 0)
           )

## 2 fourier components
# first term
lambdaTerm1 <- 2* pi / period

## fixed integer multiple
lambdaTerm2 <- 2* pi / (period* 20)

# setting up the training data
train <- train %>% mutate(
                              a1Term = sin(lambdaTerm1 * dateNumeric),
                              a2Term = sin(lambdaTerm2 * dateNumeric),
                              b1Term = cos(lambdaTerm * dateNumeric),
                              b2Term = cos(lambdaTerm2 * dateNumeric)
                  )

# setting the test data to write to a csv file
test <- test %>% mutate(
                              a1Term = sin(lambdaTerm1 * dateNumeric),
                              a2Term = sin(lambdaTerm2 * dateNumeric),
                              b1Term = cos(lambdaTerm * dateNumeric),
                              b2Term = cos(lambdaTerm2 * dateNumeric)
                  )
# build the model on training data
model2 <- lm(residualsCube ~ a1Term + b1Term + a2Term + b2Term, data = train)
#summary(model)

# rmse value
rmse <- sqrt(mean(model2$residuals^2))

# get the test set predictions
test$predictions <- predict(model2,test)
test$rmse <- rmse

test <- test %>% select(predictions,rmse)

write.csv(test,"../data/AustinModel1.csv", row.names = FALSE)

# for showing what the model predicts

data_1950$predictions <- predict(model2,data_1950)



## plot curve -> add the curve
pick <- function(condition){
  function(d) d %>% filter_(condition)
}

data_1950 %>% ggplot(aes(x=dateNumeric,y=residualsCube))+
```

```
  geom_point(alpha =.5)+
  geom_smooth(data = pick(~test == 1),
              aes(y = predictions),
              se=FALSE,
              color= "darkred")+
  geom_point(data = pick(~test == 1),
             aes(y = predictions, color = "darkred"))+
  theme_classic()
```

```
## Warning: ‘filter_()‘ was deprecated in dplyr 0.7.0.
## Please use ‘filter()‘ instead.
## See vignette('programming') for more help
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was generated.
```

```
## ‘geom_smooth()‘ using method = 'loess' and formula 'y ~ x'
```