STA6714: Assignment 2

Kyle Scott

11/12/2021

QUESTION ONE: Crypto Prices

0. Read in the relevant csv files from kaggle and display the data

```
## we have five different crypto csvs to read in: bitcoin, cardano,
## iota, xrp, and ethereum

library(readr)
bitcoin <- read_csv("coin_Bitcoin.csv")
cardano <- read_csv("coin_Cardano.csv")
ethereum <- read_csv("coin_Ethereum.csv")
iota <- read_csv("coin_iota.csv")
XRP <- read_csv("coin_XRP.csv")

head(bitcoin)
```

```
## # A tibble: 6 x 10
##     SNo Name  Symbol Date                 High   Low  Open Close Volume
##   <dbl> <chr> <chr>  <dttm>              <dbl> <dbl> <dbl> <dbl>  <dbl>
## 1     1 Bitc~ BTC    2013-04-29 23:59:59  147. 134    134.  145.       0
## 2     2 Bitc~ BTC    2013-04-30 23:59:59  147. 134.   144   139        0
## 3     3 Bitc~ BTC    2013-05-01 23:59:59  140. 108.   139   117.       0
## 4     4 Bitc~ BTC    2013-05-02 23:59:59  126.  92.3 116.  105.        0
## 5     5 Bitc~ BTC    2013-05-03 23:59:59  108.  79.1 106.   97.8      0
## 6     6 Bitc~ BTC    2013-05-04 23:59:59  115   92.5  98.1 112.        0
## # ... with 1 more variable: Marketcap <dbl>
```

```
head(cardano)
```

```
## # A tibble: 6 x 10
##     SNo Name  Symbol Date                  High    Low   Open  Close Volume
##   <dbl> <chr> <chr>  <dttm>               <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1     1 Card~ ADA    2017-10-02 23:59:59 0.0301 0.0200 0.0246 0.0259 5.76e7
## 2     2 Card~ ADA    2017-10-03 23:59:59 0.0274 0.0207 0.0258 0.0208 1.70e7
## 3     3 Card~ ADA    2017-10-04 23:59:59 0.0228 0.0209 0.0209 0.0219 9.00e6
## 4     4 Card~ ADA    2017-10-05 23:59:59 0.0222 0.0209 0.0220 0.0215 5.56e6
```

```
## 5     5 Card~ ADA    2017-10-06 23:59:59 0.0215 0.0184 0.0214 0.0185 7.78e6
## 6     6 Card~ ADA    2017-10-07 23:59:59 0.0211 0.0176 0.0184 0.0209 7.41e6
## # ... with 1 more variable: Marketcap <dbl>
```

```
head(ethereum)
```

```
## # A tibble: 6 x 10
##     SNo Name  Symbol Date                 High   Low  Open Close Volume
##   <dbl> <chr> <chr>  <dttm>              <dbl> <dbl> <dbl> <dbl>  <dbl>
## 1     1 Ethe~ ETH    2015-08-08 23:59:59 2.80  0.715 2.79  0.753 6.74e5
## 2     2 Ethe~ ETH    2015-08-09 23:59:59 0.880 0.629 0.706 0.702 5.32e5
## 3     3 Ethe~ ETH    2015-08-10 23:59:59 0.730 0.637 0.714 0.708 4.05e5
## 4     4 Ethe~ ETH    2015-08-11 23:59:59 1.13  0.663 0.708 1.07  1.46e6
## 5     5 Ethe~ ETH    2015-08-12 23:59:59 1.29  0.884 1.06  1.22  2.15e6
## 6     6 Ethe~ ETH    2015-08-13 23:59:59 1.97  1.17  1.22  1.83  4.07e6
## # ... with 1 more variable: Marketcap <dbl>
```

```
head(iota)
```

```
## # A tibble: 6 x 10
##     SNo Name  Symbol Date                 High   Low  Open Close Volume
##   <dbl> <chr> <chr>  <dttm>              <dbl> <dbl> <dbl> <dbl>  <dbl>
## 1     1 IOTA  MIOTA  2017-06-14 23:59:59 0.606 0.496 0.592 0.529 1.42e7
## 2     2 IOTA  MIOTA  2017-06-15 23:59:59 0.543 0.300 0.528 0.364 1.03e7
## 3     3 IOTA  MIOTA  2017-06-16 23:59:59 0.448 0.310 0.353 0.411 6.92e6
## 4     4 IOTA  MIOTA  2017-06-17 23:59:59 0.444 0.414 0.427 0.420 3.10e6
## 5     5 IOTA  MIOTA  2017-06-18 23:59:59 0.426 0.394 0.421 0.406 2.51e6
## 6     6 IOTA  MIOTA  2017-06-19 23:59:59 0.421 0.388 0.405 0.412 3.54e6
## # ... with 1 more variable: Marketcap <dbl>
```

```
head(XRP)
```

```
## # A tibble: 6 x 10
##     SNo Name  Symbol Date                   High     Low    Open   Close Volume
##   <dbl> <chr> <chr>  <dttm>                <dbl>   <dbl>   <dbl>   <dbl>  <dbl>
## 1     1 XRP   XRP    2013-08-05 23:59:59 0.00598 0.00561 0.00587 0.00561      0
## 2     2 XRP   XRP    2013-08-06 23:59:59 0.00566 0.00463 0.00564 0.00468      0
## 3     3 XRP   XRP    2013-08-07 23:59:59 0.00468 0.00433 0.00467 0.00442      0
```

```
## 4     4 XRP   XRP    2013-08-08 23:59:59 0.00442 0.00418 0.00440 0.00425      0
## 5     5 XRP   XRP    2013-08-09 23:59:59 0.00437 0.00425 0.00426 0.00429      0
## 6     6 XRP   XRP    2013-08-10 23:59:59 0.00437 0.00428 0.00429 0.00431      0
## # ... with 1 more variable: Marketcap <dbl>
```

We can see that for both bitcoin and XRP, the Volume column is filled with 0's which should not be the case. Upon further inspection, the raw data does have 0's in that column for both bitcoin and XRP for the first 100 or 200 dates - This was not a problem with reading in the data.

1. Plotting data for bitcoin overtime

We want to plot the price of bitcoin overtime. For this purpose, we will use the closing price each day and plot it against time to create our version of the bitcoin price chart.

*I should note that this can be done by converting the 'Close' column to a time series and plotting the data that way, but I prefer the aesthetics of ggplot so that's what I decided to use for the first couple plots. The nice thing about using a time series, however, is the ability to plot smoother curves using moving averages and weighted averages, etc., if those plots are necessary.

```r
## We will need to read in the ggplot library to create our plot
library(ggplot2)
```

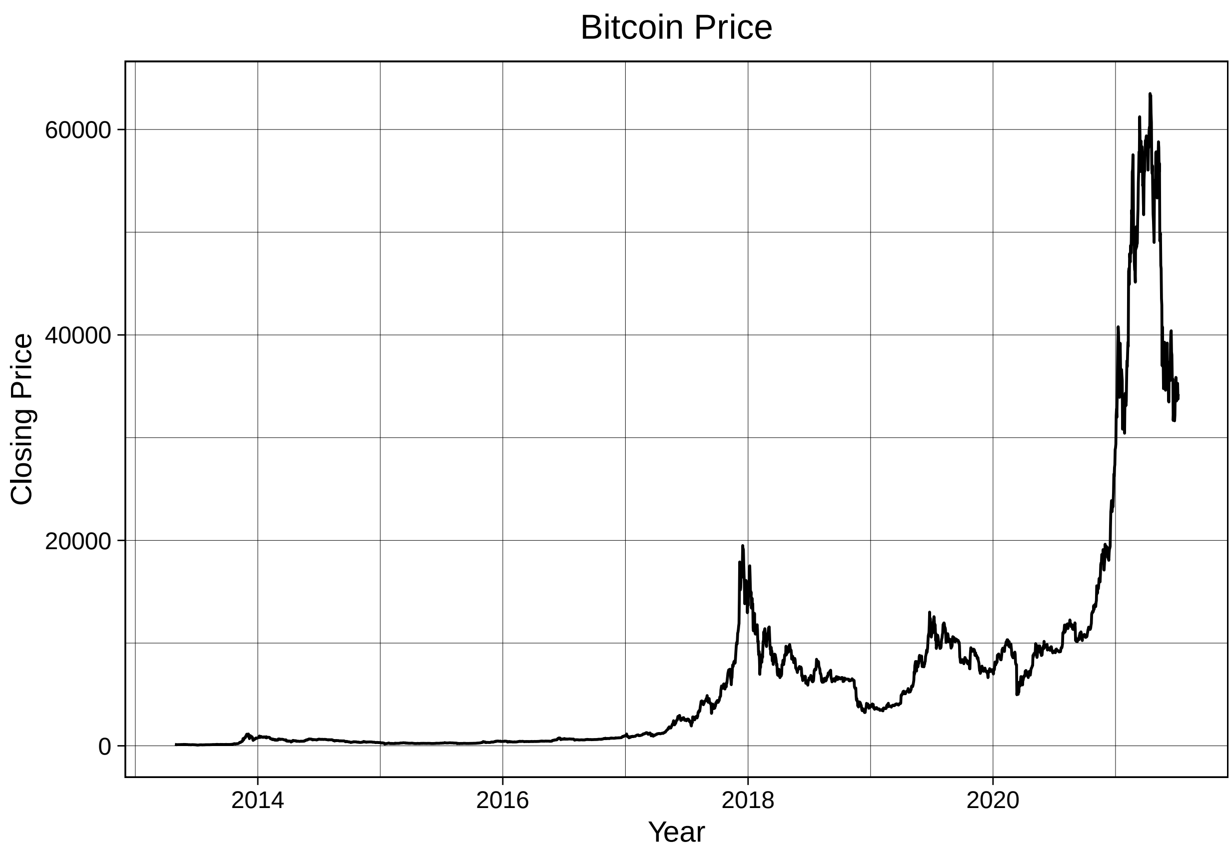
```
## Warning: replacing previous import 'vctrs::data_frame' by 'tibble::data_frame'
## when loading 'dplyr'
```

```r
## I came across an error with dates, so we need to save the date
## column as a date variable without the time clouding the column
library(anytime)
```

```
## Warning: package 'anytime' was built under R version 3.6.2
```

```r
bitcoin$Date <- as.Date(anytime(bitcoin$Date))


## Create our initial ggplot, scaling the x axis as a date
bit_price <- ggplot(bitcoin, aes(Date, Close)) +
  geom_line() +
  scale_x_date("Year") +
  ylab("Closing Price") +
```

```
  theme_linedraw()


## Add a title and center it

bit_price <- bit_price + ggtitle("Bitcoin Price") + theme(plot.title = element_text(hjust = 0.5))


## Plot!

bit_price
```

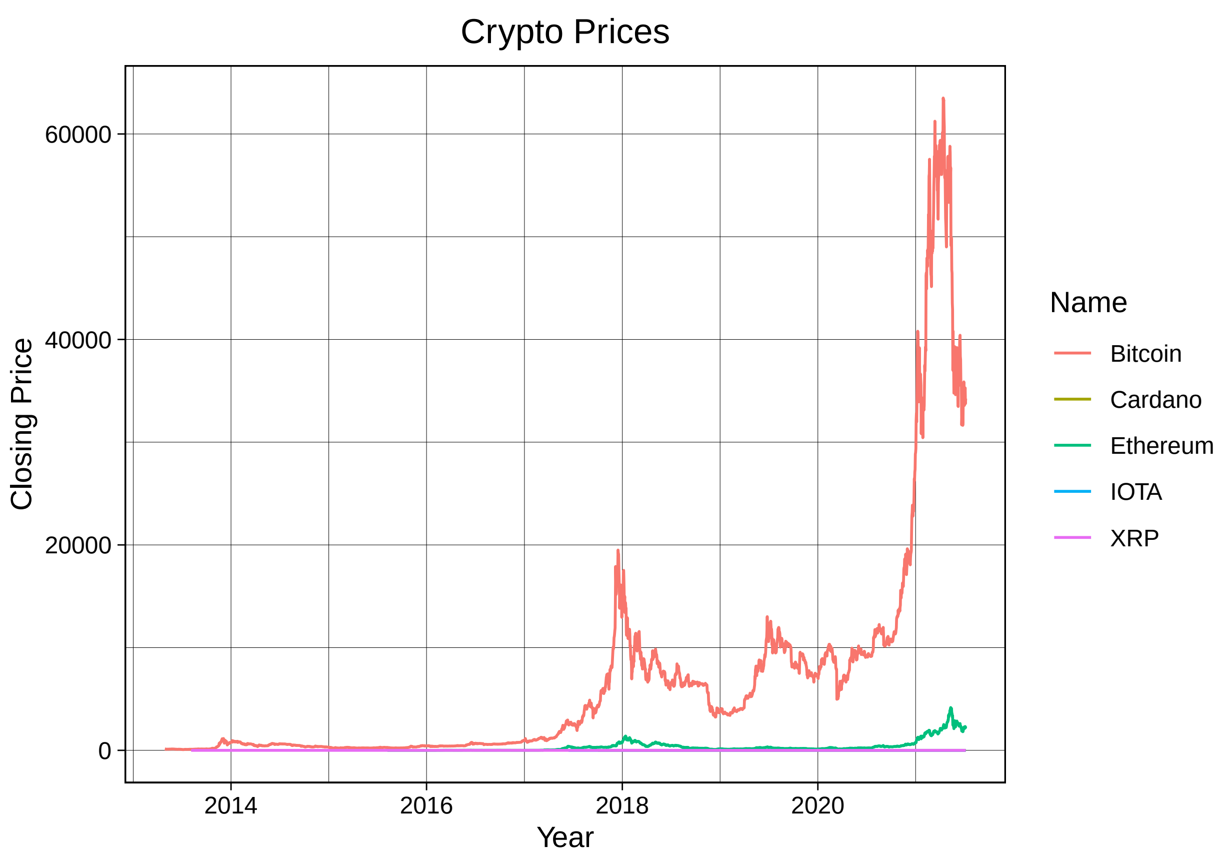


2. Plot all 5 Crypto's overtime on the same plot

We can use the same method as shown for bitcoin in part 1, however we just need to overlay the other 4 coins

```
## ggplot doesn't work well with separate dataframes

## first - combine all 5 cryptos into 1 df

## we have the 'Name' column to distinguish

cardano$Date <- as.Date(anytime(cardano$Date))

ethereum$Date <- as.Date(anytime(ethereum$Date))

iota$Date <- as.Date(anytime(iota$Date))

XRP$Date <- as.Date(anytime(XRP$Date))
```

```
crypto_full <- rbind(bitcoin, cardano, ethereum, iota, XRP)

## create the initial plot, coloring by 'Name'
crypto_price <- ggplot(data = crypto_full,
                       mapping = aes(Date, Close, color = Name)) +
  geom_line() +
  scale_x_date("Year") +
  ylab("Closing Price") +
  theme_linedraw()

## Add and center the title
crypto_price <- crypto_price + ggtitle("Crypto Prices") + theme(plot.title = element_text(hjust = 0.5))

## Plot!
crypto_price
```



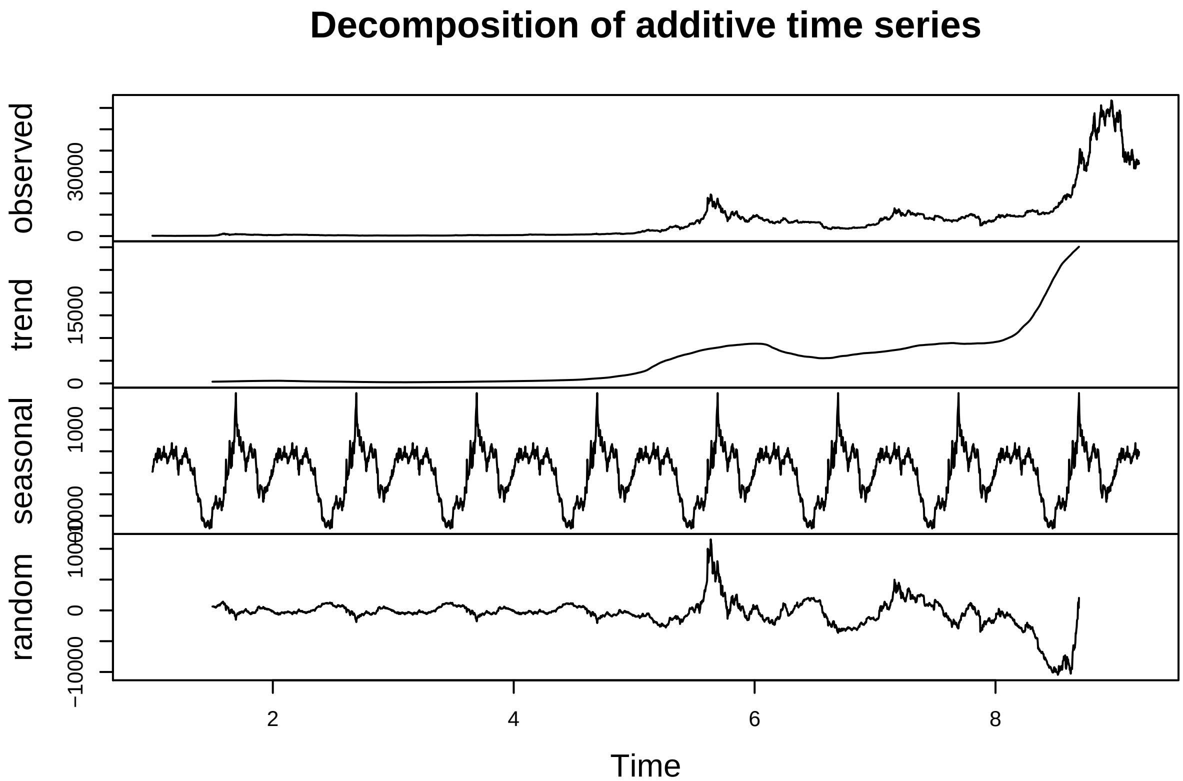
This plot doesn't show us much other than the fact that bitcoin has a price that is magnitudes higher than some of the other cryptos. Cardano, Ethereum and XRP are all cluttered at the very bottom of the graph with a high price of under 50.

3. Display the auto correlation of each trace

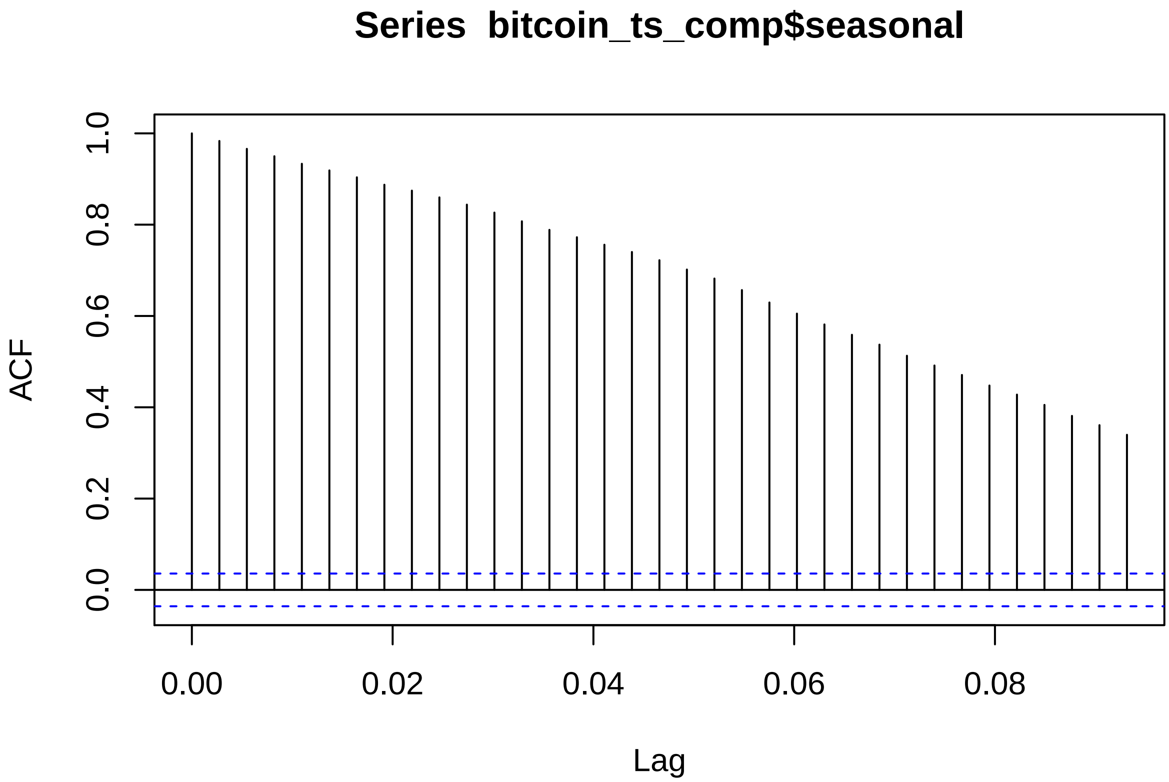
Now we will convert our data to time series so that we can display the auto correlation.

Bitcoin

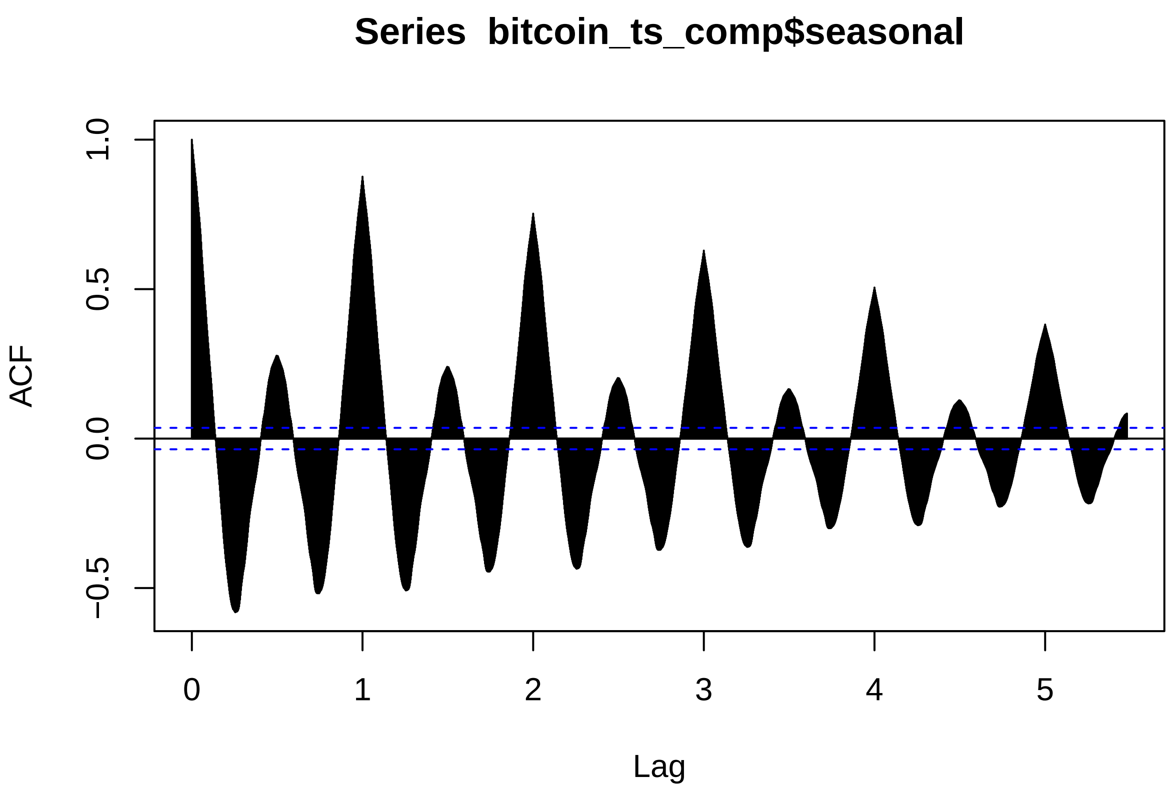
```
## create a time series for bitcoin
bitcoin_ts <- ts(bitcoin$Close, frequency = 365, start(2013, 119))
bitcoin_ts_comp <- decompose(bitcoin_ts)
plot(bitcoin_ts_comp)
```




```
## default acf plot
acf(bitcoin_ts_comp$seasonal)
```

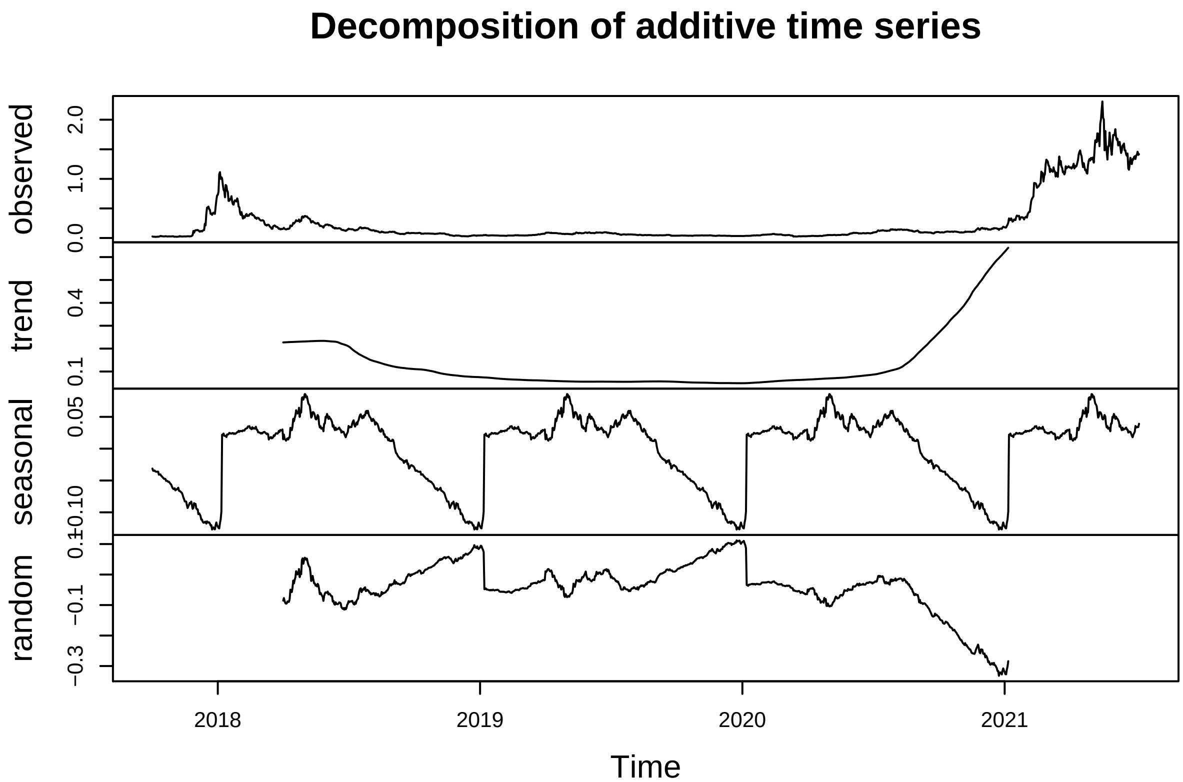



```r
## increase lag max to show trends
acf(bitcoin_ts_comp$seasonal, lag.max = 2000)
```

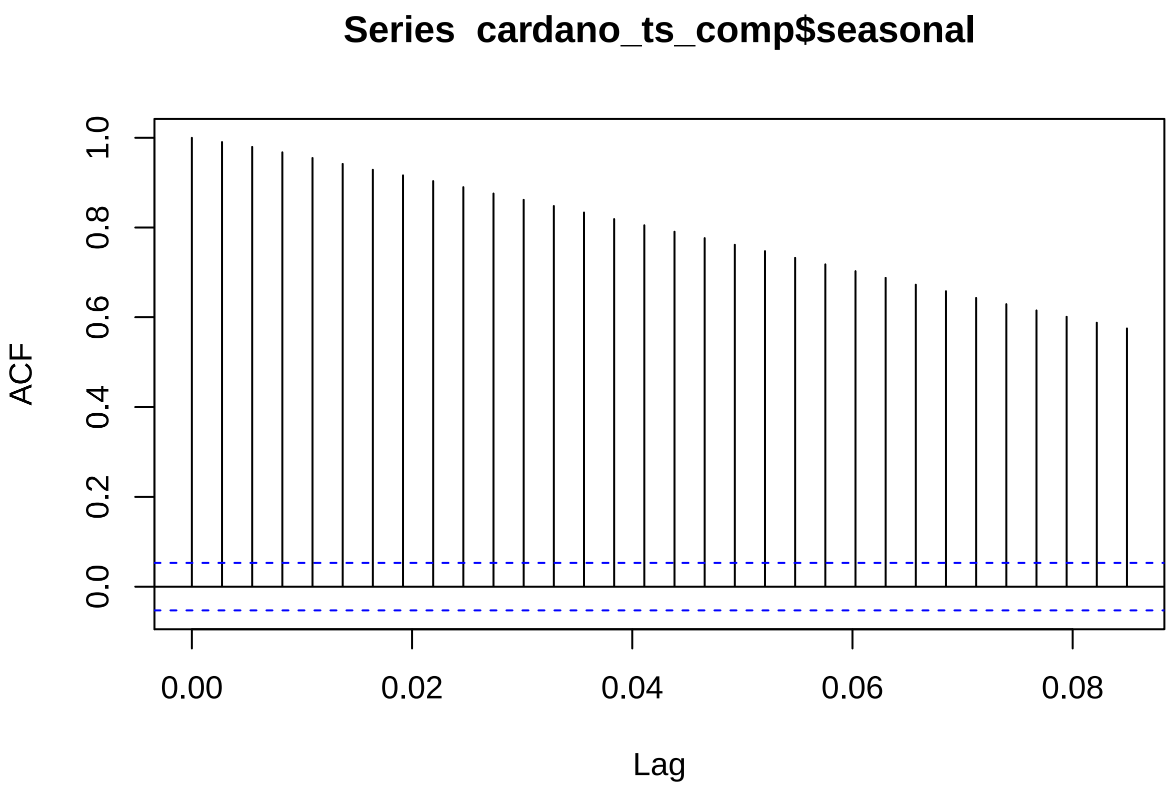


Cardano

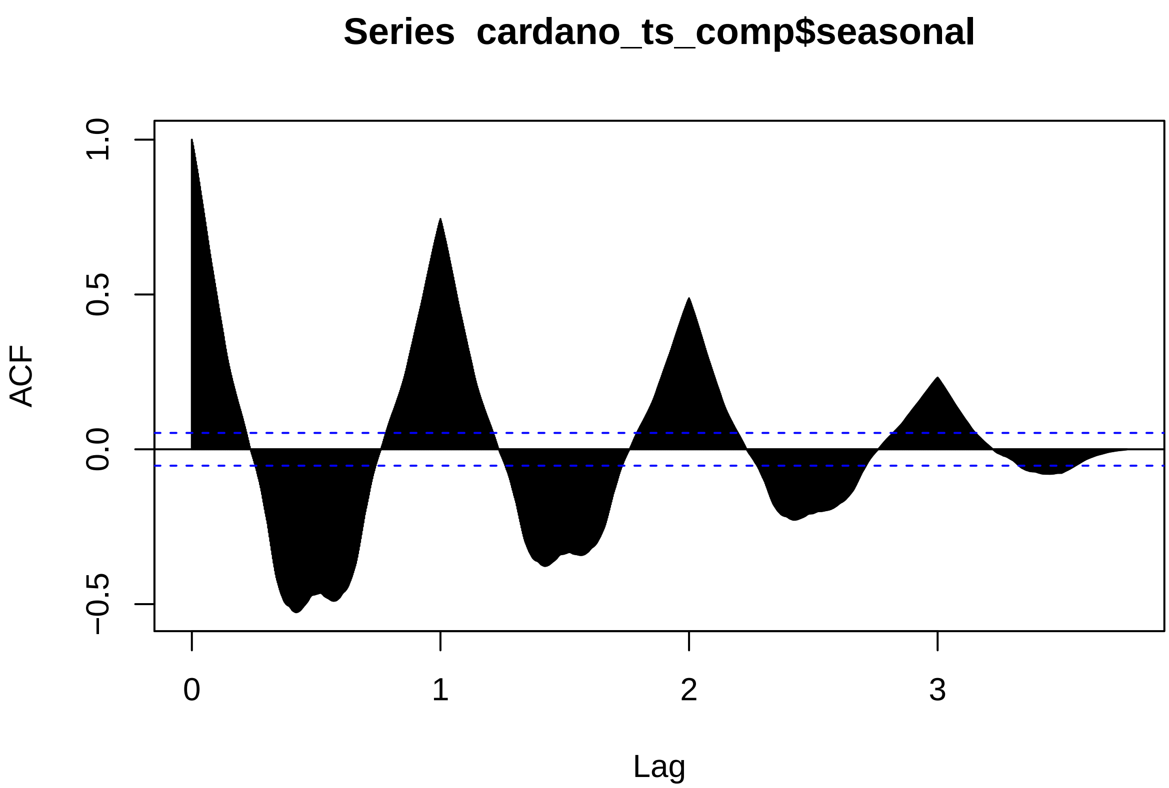
```r
cardano_ts <- ts(cardano$Close, frequency = 365, start = c(2017, 275))

cardano_ts_comp <- decompose(cardano_ts)

plot(cardano_ts_comp)
```




```r
## default acf plot

acf(cardano_ts_comp$seasonal)
```

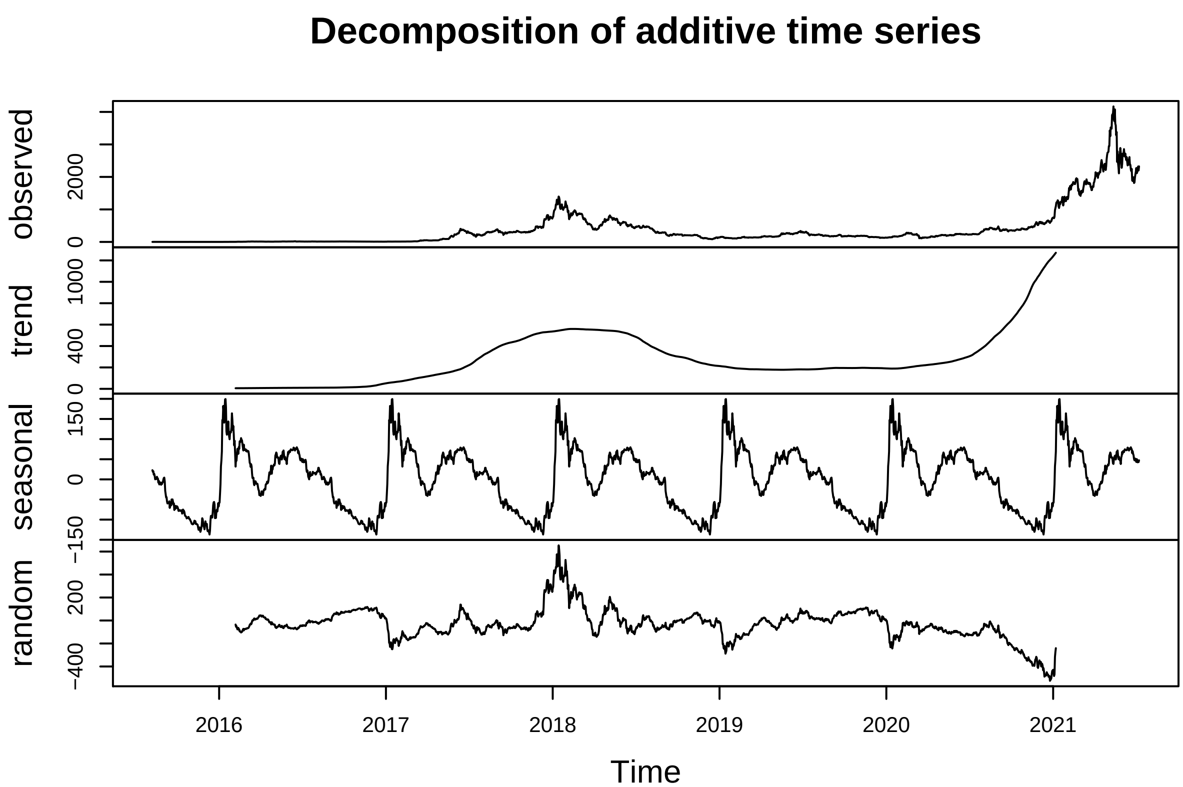



```r
## increase lag max to show trends
acf(cardano_ts_comp$seasonal, lag.max = 2000)
```

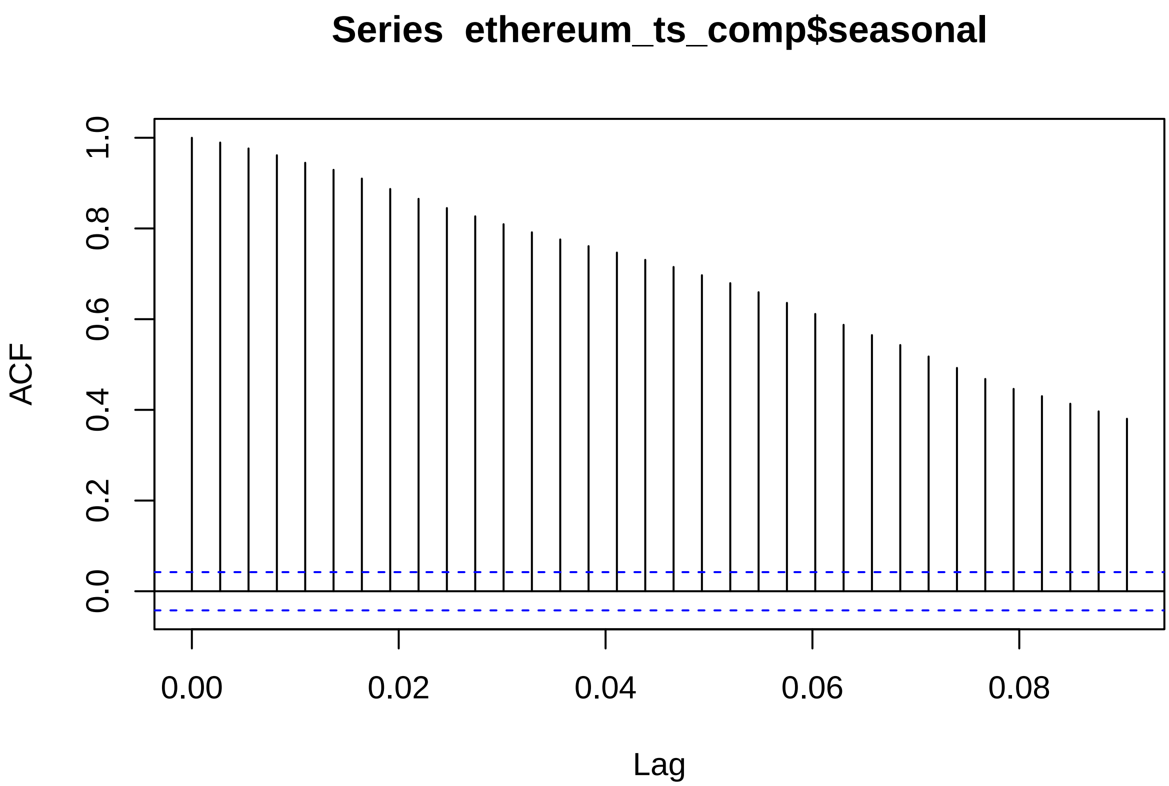


Ethereum

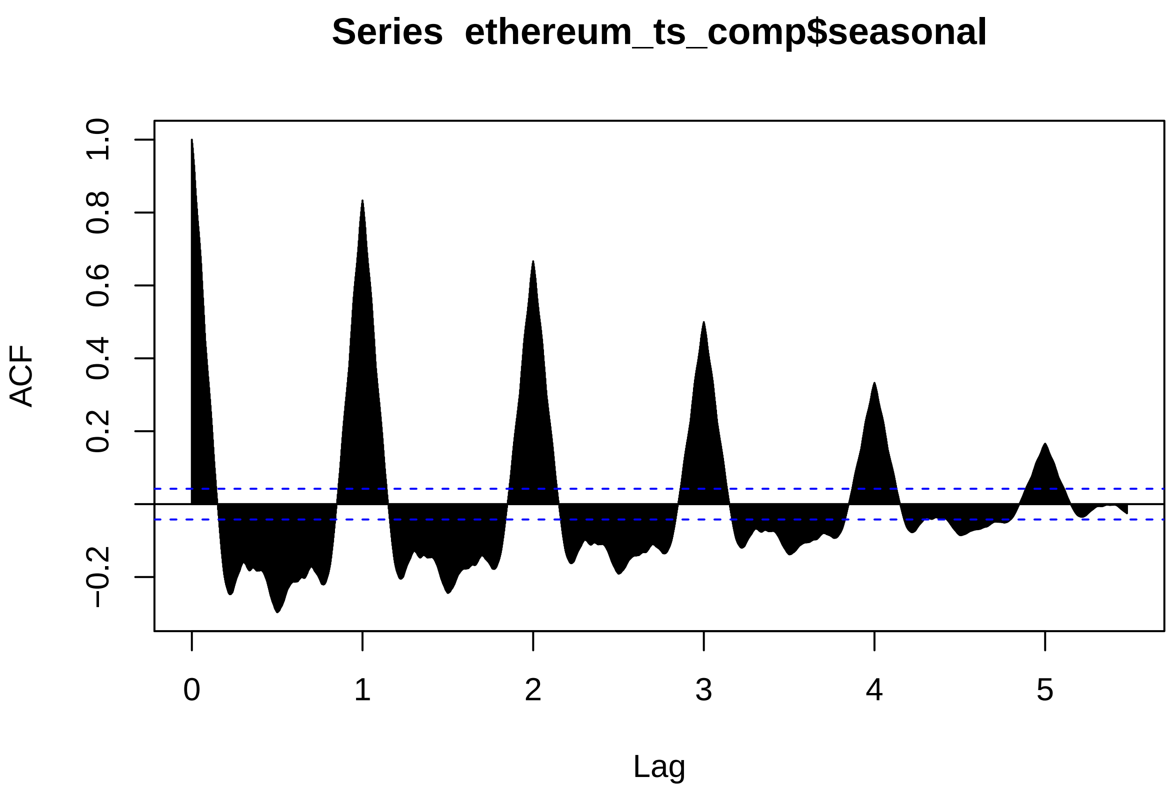
```r
ethereum_ts <- ts(ethereum$Close, frequency =  365, start = c(2015, 220))

ethereum_ts_comp <- decompose(ethereum_ts)

plot(ethereum_ts_comp)
```




```r
## default acf plot

acf(ethereum_ts_comp$seasonal)
```

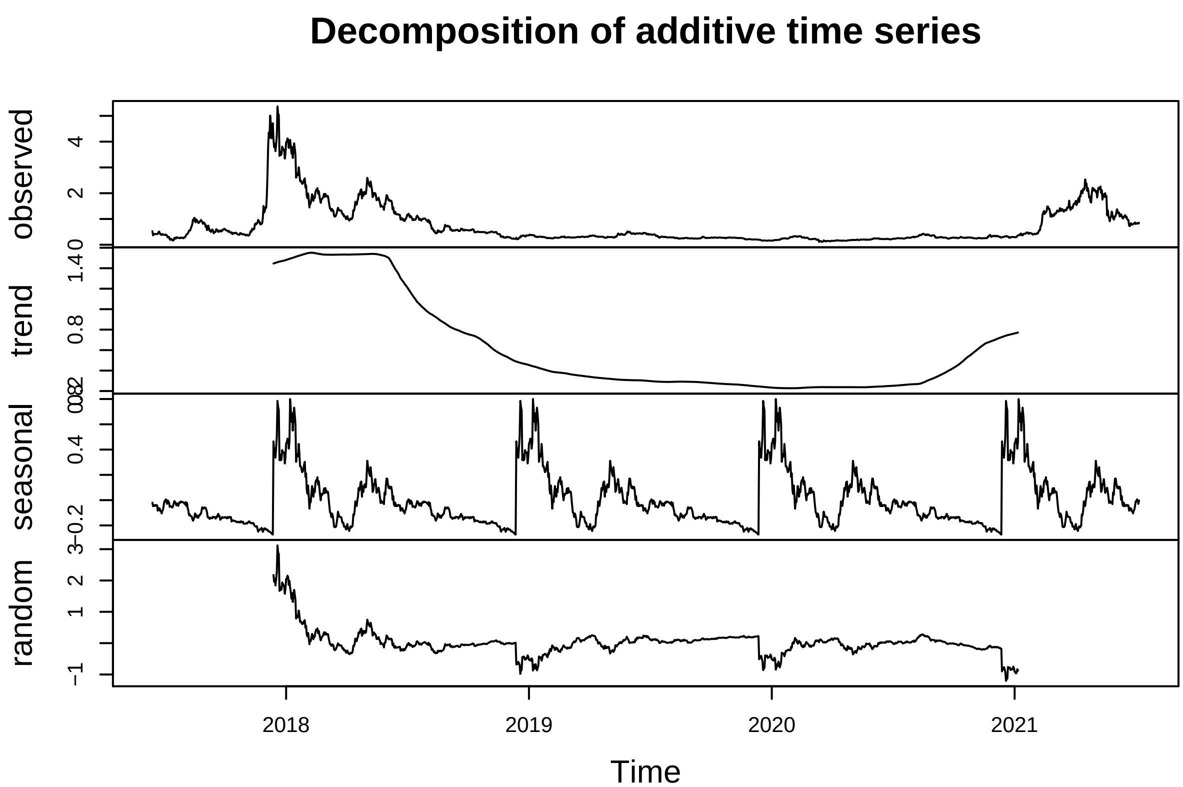



```r
## increase lag max to show trends
acf(ethereum_ts_comp$seasonal, lag.max = 2000)
```

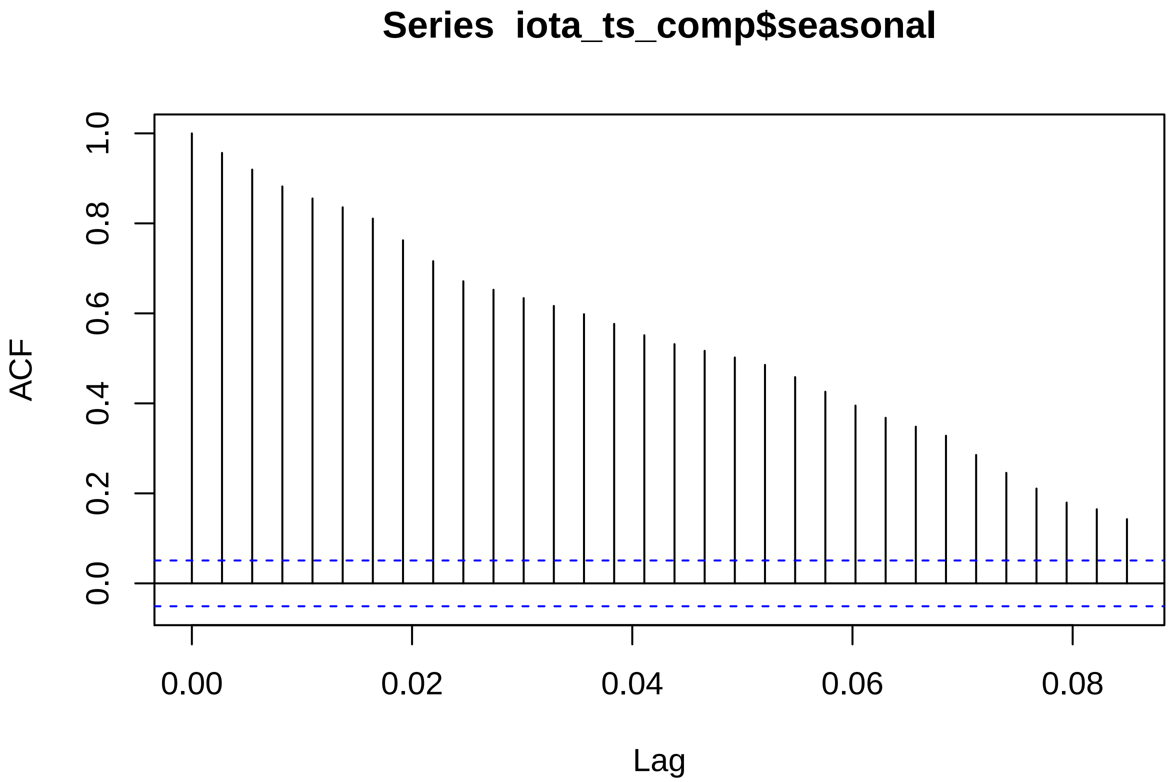


Iota

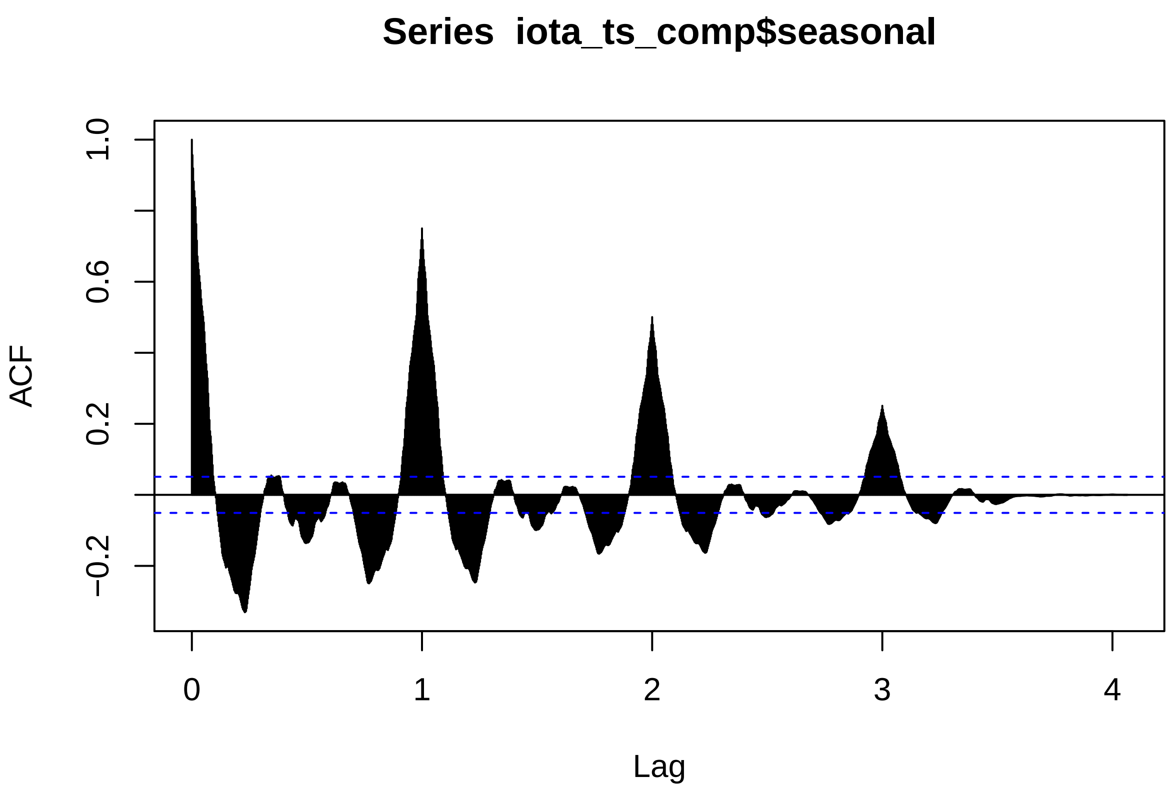
```
iota_ts <- ts(iota$Close, frequency = 365, start = c(2017, 165))

iota_ts_comp <- decompose(iota_ts)

plot(iota_ts_comp)
```




```
## default acf plot

acf(iota_ts_comp$seasonal)
```

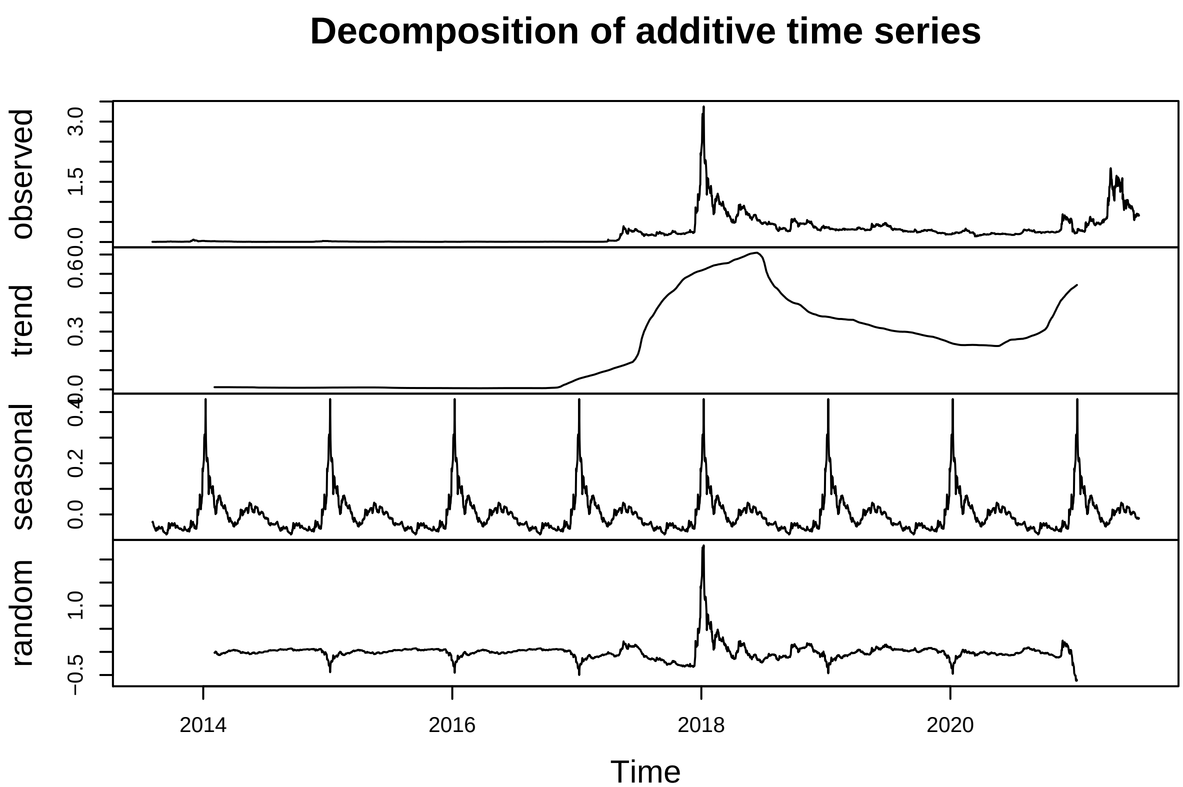



```r
## increase lag max to show trends
acf(iota_ts_comp$seasonal, lag.max = 2000)
```

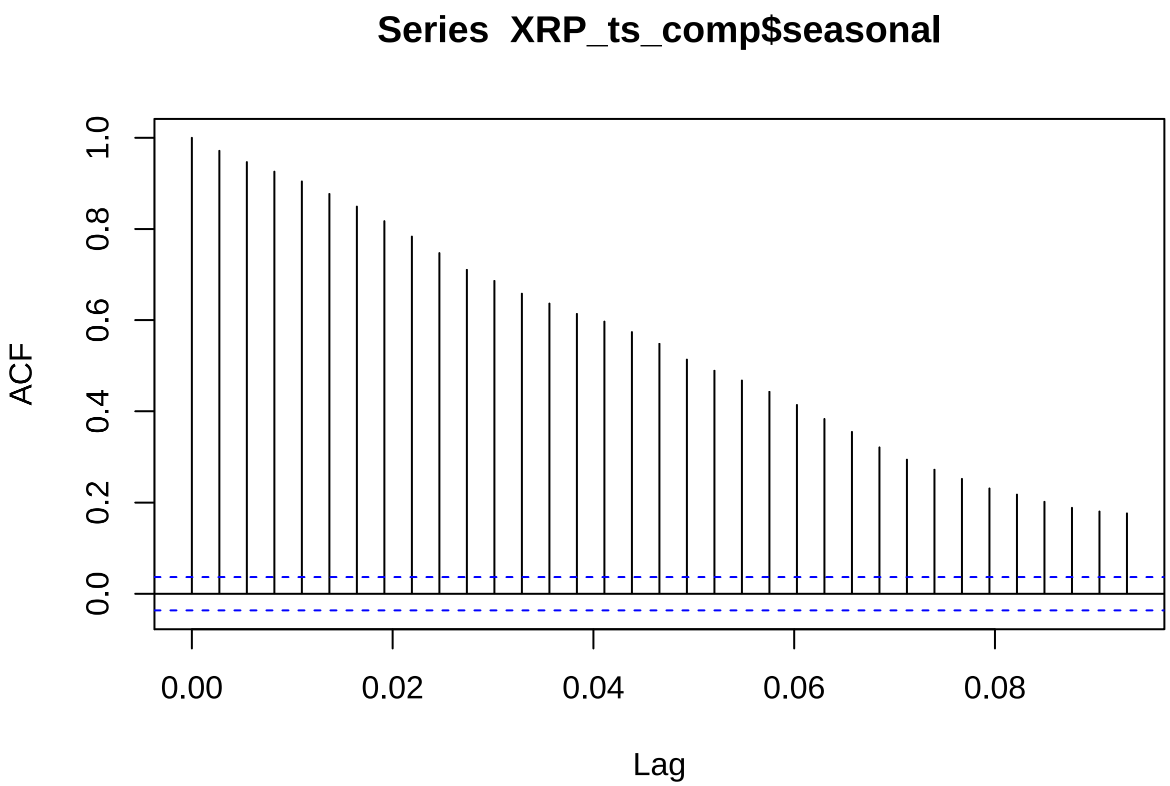


XRP

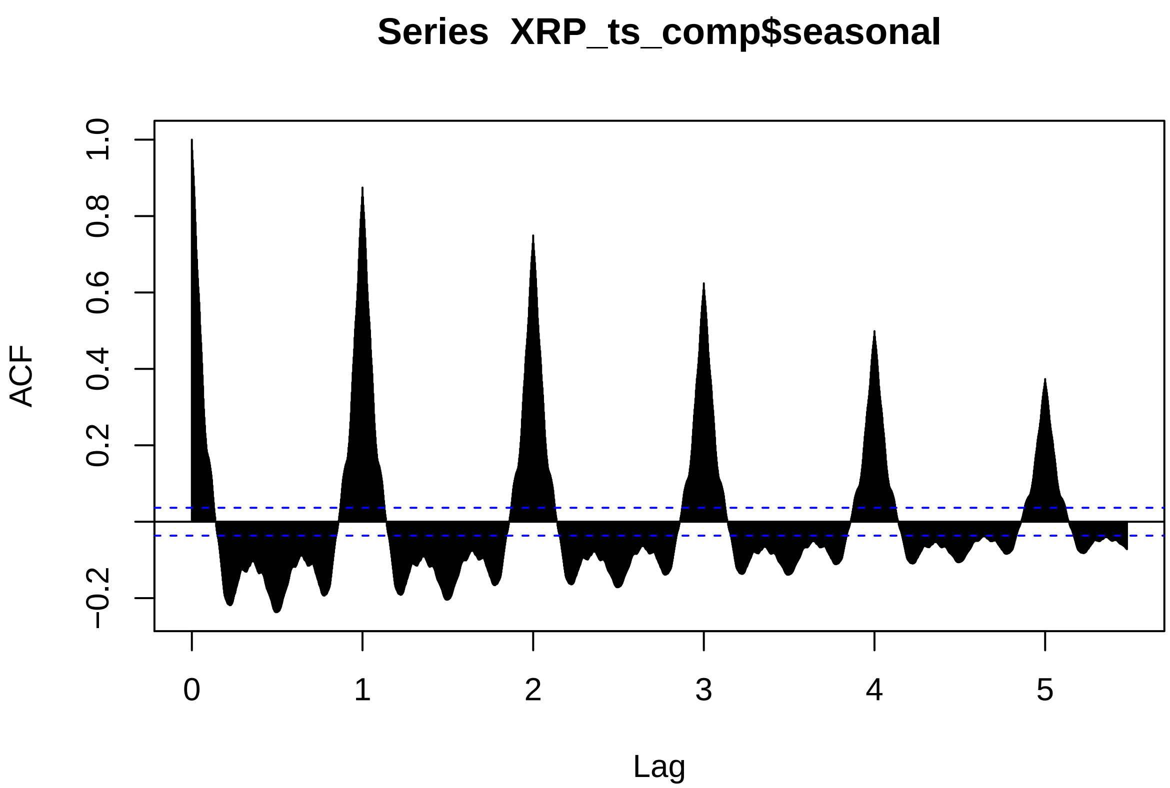
```
XRP_ts <- ts(XRP$Close, frequency = 365, start = c(2013, 217))

XRP_ts_comp <- decompose(XRP_ts)

plot(XRP_ts_comp)
```




```
## default acf plot

acf(XRP_ts_comp$seasonal)
```




```r
## increase lag max to show trends
acf(XRP_ts_comp$seasonal, lag.max = 2000)
```



Looking at all five of these ACF plots, we can see that none of them follow a uniform seasonal pattern. There does not appear to be any impact of the seasons on how well a specific cryptocurrency is performing. This shouldn't be all that surprising, currencies and stocks go up and down all the time for various reasons in which the date is completely irrelevant. It isn't like the ebs and flows of seasonal weather, which does follow a predictable quarterly pattern.

4. Use Decision Trees to predict the price of the cardano coin

```
d1 <- cardano$Close[1:1368]
d2 <- cardano$Close[2:1369]
d3 <- cardano$Close[3:1370]
d4 <- cardano$Close[4:1371]
d5 <- cardano$Close[5:1372]
d6 <- cardano$Close[6:1373]
cardano_five <- data.frame(d1, d2, d3, d4, d5, d6)


library(party)
```

```
## Warning: package 'party' was built under R version 3.6.2

## Loading required package: grid

## Loading required package: mvtnorm

## Loading required package: modeltools

## Loading required package: stats4

## Loading required package: strucchange

## Loading required package: zoo

## Warning: package 'zoo' was built under R version 3.6.2

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich
```

```r
card_five_tree <- ctree(d6 ~ d1 + d2 + d3 + d4 + d5, data = cardano_five)
card_five_tree
```

```
##
##   Conditional inference tree with 47 terminal nodes
##
## Response:  d6
## Inputs:  d1, d2, d3, d4, d5
## Number of observations:  1368
##
## 1) d5 <= 0.670286; criterion = 1, statistic = 1353.643
##   2) d5 <= 0.213189; criterion = 1, statistic = 1183.016
##     3) d5 <= 0.09819689; criterion = 1, statistic = 1026.355
##       4) d5 <= 0.06295813; criterion = 1, statistic = 741.808
##         5) d5 <= 0.04186575; criterion = 1, statistic = 425.204
##           6) d5 <= 0.03446628; criterion = 1, statistic = 204.783
##             7) d5 <= 0.0304375; criterion = 1, statistic = 83.611
##               8) d5 <= 0.0229862; criterion = 1, statistic = 35.656
##                 9)*  weights = 13
##               8) d5 > 0.0229862
##                 10)*  weights = 57
##             7) d5 > 0.0304375
##               11) d5 <= 0.03169259; criterion = 0.992, statistic = 10.055
##                 12)*  weights = 10
##               11) d5 > 0.03169259
##                 13) d3 <= 0.03198441; criterion = 0.968, statistic = 7.422
##                   14)*  weights = 8
##                 13) d3 > 0.03198441
##                   15) d1 <= 0.03440835; criterion = 0.996, statistic = 11.257
##                     16)*  weights = 18
##                   15) d1 > 0.03440835
##                     17)*  weights = 10
##           6) d5 > 0.03446628
##             18) d5 <= 0.03976585; criterion = 1, statistic = 32.672
```

```
##             19)*  weights = 89
##           18) d5 > 0.03976585
##             20)*  weights = 39
##       5) d5 > 0.04186575
##         21) d5 <= 0.050298; criterion = 1, statistic = 146.407
##           22) d5 <= 0.04650951; criterion = 1, statistic = 88.294
##             23) d5 <= 0.04414719; criterion = 1, statistic = 21.753
##               24)*  weights = 59
##             23) d5 > 0.04414719
##               25)*  weights = 43
##           22) d5 > 0.04650951
##             26)*  weights = 53
##         21) d5 > 0.050298
##           27) d5 <= 0.05697922; criterion = 0.996, statistic = 11.071
##             28) d3 <= 0.04767682; criterion = 0.997, statistic = 11.67
##               29)*  weights = 7
##             28) d3 > 0.04767682
##               30) d5 <= 0.0525815; criterion = 0.999, statistic = 13.847
##                 31)*  weights = 25
##               30) d5 > 0.0525815
##                 32)*  weights = 26
##           27) d5 > 0.05697922
##             33)*  weights = 41
##     4) d5 > 0.06295813
##       34) d5 <= 0.0783868; criterion = 1, statistic = 217.778
##         35) d5 <= 0.07057959; criterion = 1, statistic = 46.254
##           36)*  weights = 43
##         35) d5 > 0.07057959
##           37)*  weights = 56
##       34) d5 > 0.0783868
##         38) d5 <= 0.08943269; criterion = 1, statistic = 95.347
##           39) d5 <= 0.0851952; criterion = 1, statistic = 16.386
##             40)*  weights = 72
##           39) d5 > 0.0851952
```

```
##               41)*  weights = 27
##             38) d5 > 0.08943269
##               42) d4 <= 0.0933333; criterion = 1, statistic = 24.601
##                 43)*  weights = 36
##               42) d4 > 0.0933333
##                 44) d2 <= 0.0924532; criterion = 0.988, statistic = 9.261
##                   45)*  weights = 11
##                 44) d2 > 0.0924532
##                   46)*  weights = 28
##       3) d5 > 0.09819689
##         47) d5 <= 0.133891; criterion = 1, statistic = 241.277
##           48) d5 <= 0.116355; criterion = 1, statistic = 83.544
##             49) d5 <= 0.105469; criterion = 0.999, statistic = 14.321
##               50)*  weights = 31
##             49) d5 > 0.105469
##               51) d1 <= 0.1167117; criterion = 0.995, statistic = 10.877
##                 52)*  weights = 26
##               51) d1 > 0.1167117
##                 53)*  weights = 10
##           48) d5 > 0.116355
##             54) d5 <= 0.1254107; criterion = 0.993, statistic = 10.337
##               55) d3 <= 0.1175538; criterion = 0.952, statistic = 6.664
##                 56)*  weights = 9
##               55) d3 > 0.1175538
##                 57)*  weights = 21
##             54) d5 > 0.1254107
##               58)*  weights = 26
##         47) d5 > 0.133891
##           59) d5 <= 0.173172; criterion = 1, statistic = 102.926
##             60) d5 <= 0.15096; criterion = 1, statistic = 31.193
##               61)*  weights = 59
##             60) d5 > 0.15096
##               62)*  weights = 63
##           59) d5 > 0.173172
```

```
##          63) d5 <= 0.193557; criterion = 0.999, statistic = 14.991
##            64)*  weights = 17
##          63) d5 > 0.193557
##            65)*  weights = 19
##   2) d5 > 0.213189
##     66) d5 <= 0.438836; criterion = 1, statistic = 137.129
##       67) d5 <= 0.312875; criterion = 1, statistic = 94.966
##         68) d5 <= 0.256646; criterion = 1, statistic = 23.299
##           69)*  weights = 26
##         68) d5 > 0.256646
##           70)*  weights = 32
##       67) d5 > 0.312875
##         71) d5 <= 0.384315; criterion = 1, statistic = 32.792
##           72) d5 <= 0.3442336; criterion = 0.991, statistic = 9.803
##             73)*  weights = 17
##           72) d5 > 0.3442336
##             74)*  weights = 29
##         71) d5 > 0.384315
##           75)*  weights = 21
##     66) d5 > 0.438836
##       76) d5 <= 0.531913; criterion = 1, statistic = 15.984
##         77)*  weights = 10
##       76) d5 > 0.531913
##         78)*  weights = 17
## 1) d5 > 0.670286
##   79) d5 <= 1.384869; criterion = 1, statistic = 142.231
##     80) d5 <= 1.02715; criterion = 1, statistic = 89.278
##       81) d5 <= 0.9057194; criterion = 0.989, statistic = 9.394
##         82)*  weights = 19
##       81) d5 > 0.9057194
##         83)*  weights = 9
##     80) d5 > 1.02715
##       84) d5 <= 1.225582; criterion = 1, statistic = 44.347
##         85) d5 <= 1.135003; criterion = 0.993, statistic = 10.174
```

```
##          86)*  weights = 23
##        85) d5 > 1.135003
##          87)*  weights = 27
##      84) d5 > 1.225582
##        88)*  weights = 33
##  79) d5 > 1.384869
##    89) d5 <= 1.810469; criterion = 1, statistic = 28.859
##      90) d5 <= 1.552277; criterion = 0.999, statistic = 13.936
##        91)*  weights = 22
##      90) d5 > 1.552277
##        92)*  weights = 24
##    89) d5 > 1.810469
##      93)*  weights = 7
```

```r
d1 <- cardano$Close[1:1363]
d2 <- cardano$Close[2:1364]
d3 <- cardano$Close[3:1365]
d4 <- cardano$Close[4:1366]
d5 <- cardano$Close[5:1367]
d6 <- cardano$Close[6:1368]
d7 <- cardano$Close[7:1369]
d8 <- cardano$Close[8:1370]
d9 <- cardano$Close[9:1371]
d10 <- cardano$Close[10:1372]
d11 <- cardano$Close[11:1373]

cardano_ten <- data.frame(d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11)

library(party)

card_ten_tree <- ctree(d11 ~ d1 + d2 + d3 + d4 + d5 + d6 + d7 + d8 + d9 + d10, data = cardano_ten)
card_ten_tree
```

```
##
##   Conditional inference tree with 47 terminal nodes
```

```
##
## Response:  d11
## Inputs:  d1, d2, d3, d4, d5, d6, d7, d8, d9, d10
## Number of observations:  1363
##
## 1) d10 <= 0.670286; criterion = 1, statistic = 1348.676
##   2) d10 <= 0.213189; criterion = 1, statistic = 1178.043
##     3) d10 <= 0.09819689; criterion = 1, statistic = 1021.281
##       4) d10 <= 0.06295813; criterion = 1, statistic = 736.529
##         5) d10 <= 0.04380163; criterion = 1, statistic = 417.962
##           6) d10 <= 0.03446628; criterion = 1, statistic = 250.351
##             7) d10 <= 0.0304375; criterion = 1, statistic = 73.799
##               8) d10 <= 0.0229862; criterion = 1, statistic = 24.188
##                 9)*  weights = 8
##               8) d10 > 0.0229862
##                 10)*  weights = 57
##             7) d10 > 0.0304375
##               11) d10 <= 0.03169259; criterion = 0.985, statistic = 10.055
##                 12)*  weights = 10
##               11) d10 > 0.03169259
##                 13) d2 <= 0.03072724; criterion = 0.975, statistic = 9.109
##                   14)*  weights = 7
##                 13) d2 > 0.03072724
##                   15) d4 <= 0.03363213; criterion = 0.981, statistic = 9.651
##                     16)*  weights = 11
##                   15) d4 > 0.03363213
##                     17)*  weights = 18
##           6) d10 > 0.03446628
##             18) d10 <= 0.03976585; criterion = 1, statistic = 82.916
##               19)*  weights = 89
##             18) d10 > 0.03976585
##               20) d10 <= 0.04186575; criterion = 0.984, statistic = 9.992
##                 21)*  weights = 39
##               20) d10 > 0.04186575
```

```
##                 22)*  weights = 52
##         5) d10 > 0.04380163
##           23) d10 <= 0.0504212; criterion = 1, statistic = 95.549
##             24) d10 <= 0.04693137; criterion = 1, statistic = 42.075
##               25)*  weights = 59
##             24) d10 > 0.04693137
##               26)*  weights = 45
##           23) d10 > 0.0504212
##             27) d10 <= 0.05697922; criterion = 0.989, statistic = 10.631
##               28) d8 <= 0.04784594; criterion = 0.994, statistic = 11.883
##                 29)*  weights = 7
##               28) d8 > 0.04784594
##                 30) d10 <= 0.0525815; criterion = 0.997, statistic = 12.926
##                   31)*  weights = 24
##                 30) d10 > 0.0525815
##                   32)*  weights = 26
##             27) d10 > 0.05697922
##               33)*  weights = 41
##       4) d10 > 0.06295813
##         34) d10 <= 0.0783868; criterion = 1, statistic = 217.778
##           35) d10 <= 0.07057959; criterion = 1, statistic = 46.254
##             36)*  weights = 43
##           35) d10 > 0.07057959
##             37)*  weights = 56
##         34) d10 > 0.0783868
##           38) d10 <= 0.08943269; criterion = 1, statistic = 95.347
##             39) d10 <= 0.0851952; criterion = 0.999, statistic = 16.386
##               40)*  weights = 72
##             39) d10 > 0.0851952
##               41)*  weights = 27
##           38) d10 > 0.08943269
##             42) d9 <= 0.0933333; criterion = 1, statistic = 24.601
##               43) d1 <= 0.08870804; criterion = 0.97, statistic = 8.784
##                 44)*  weights = 22
```

```
##             43) d1 > 0.08870804
##               45)*  weights = 14
##           42) d9 > 0.0933333
##             46) d7 <= 0.0924532; criterion = 0.977, statistic = 9.261
##               47)*  weights = 11
##             46) d7 > 0.0924532
##               48)*  weights = 28
##    3) d10 > 0.09819689
##      49) d10 <= 0.133891; criterion = 1, statistic = 241.277
##        50) d10 <= 0.116355; criterion = 1, statistic = 83.544
##          51) d10 <= 0.105469; criterion = 0.998, statistic = 14.321
##            52)*  weights = 31
##          51) d10 > 0.105469
##            53) d4 <= 0.1237604; criterion = 0.995, statistic = 12.198
##              54)*  weights = 26
##            53) d4 > 0.1237604
##              55)*  weights = 10
##        50) d10 > 0.116355
##          56) d10 <= 0.1254107; criterion = 0.987, statistic = 10.337
##            57)*  weights = 30
##          56) d10 > 0.1254107
##            58)*  weights = 26
##      49) d10 > 0.133891
##        59) d10 <= 0.173172; criterion = 1, statistic = 102.926
##          60) d10 <= 0.15096; criterion = 1, statistic = 31.193
##            61)*  weights = 59
##          60) d10 > 0.15096
##            62)*  weights = 63
##        59) d10 > 0.173172
##          63) d10 <= 0.193557; criterion = 0.999, statistic = 14.991
##            64)*  weights = 17
##          63) d10 > 0.193557
##            65)*  weights = 19
##   2) d10 > 0.213189
```

```
##     66) d10 <= 0.438836; criterion = 1, statistic = 137.129
##       67) d10 <= 0.312875; criterion = 1, statistic = 94.966
##         68) d10 <= 0.256646; criterion = 1, statistic = 23.299
##           69)*  weights = 26
##         68) d10 > 0.256646
##           70)*  weights = 32
##       67) d10 > 0.312875
##         71) d10 <= 0.384315; criterion = 1, statistic = 32.792
##           72) d10 <= 0.3442336; criterion = 0.983, statistic = 9.803
##             73)*  weights = 17
##           72) d10 > 0.3442336
##             74)*  weights = 29
##         71) d10 > 0.384315
##           75)*  weights = 21
##     66) d10 > 0.438836
##       76) d10 <= 0.531913; criterion = 0.999, statistic = 15.984
##         77)*  weights = 10
##       76) d10 > 0.531913
##         78)*  weights = 17
## 1) d10 > 0.670286
##   79) d10 <= 1.384869; criterion = 1, statistic = 142.231
##     80) d10 <= 1.02715; criterion = 1, statistic = 89.278
##       81) d10 <= 0.9057194; criterion = 0.978, statistic = 9.394
##         82)*  weights = 19
##       81) d10 > 0.9057194
##         83)*  weights = 9
##     80) d10 > 1.02715
##       84) d10 <= 1.225582; criterion = 1, statistic = 44.347
##         85) d10 <= 1.135003; criterion = 0.986, statistic = 10.174
##           86)*  weights = 23
##         85) d10 > 1.135003
##           87)*  weights = 27
##       84) d10 > 1.225582
##         88)*  weights = 33
```

```
##   79) d10 > 1.384869
##     89) d10 <= 1.810469; criterion = 1, statistic = 28.859
##       90) d10 <= 1.552277; criterion = 0.998, statistic = 13.936
##         91)*  weights = 22
##       90) d10 > 1.552277
##         92)*  weights = 24
##     89) d10 > 1.810469
##       93)*  weights = 7
```

We can see through our two decision trees that the decision tree method takes into account very little what the price was many days prior to the current day. In fact, the two trees appear to be exactly the same. This particular tree appears mainly interested in the previous day with a small emphasis on the two days prior to that. Overall, there is zero difference between the trees at 5 days and at 10 days. Considering we are dealing with a time series, it makes sense to me that the day prior to the current day is by far the most important predictor of current price.

5. Use Decision Trees to predict the price of Cardano using Bitcoin and XRP

```r
# we have to make sure all the dates line up here. this way all data
  # begins at 10-02-2017
d1 <- bitcoin$Close[1618:2985]
d2 <- bitcoin$Close[1619:2986]
d3 <- bitcoin$Close[1620:2987]
d4 <- bitcoin$Close[1621:2988]
d5 <- bitcoin$Close[1622:2989]
d6 <- cardano$Close[6:1373]


cardano_five_w_bit <- data.frame(d1, d2, d3, d4, d5, d6)


library(party)
library(partykit)
```

```
## Warning: package 'partykit' was built under R version 3.6.2
```

```
## Loading required package: libcoin
```

```
##
## Attaching package: 'partykit'
## The following objects are masked from 'package:party':
##
##     cforest, ctree, ctree_control, edge_simple, mob, mob_control,
##     node_barplot, node_bivplot, node_boxplot, node_inner, node_surv,
##     node_terminal, varimp
```

```
card_five_w_bit_tree <- ctree(d6 ~ d1 + d2 + d3 + d4 + d5, data = cardano_five_w_bit)


print(card_five_w_bit_tree)
```

```
##
## Model formula:
## d6 ~ d1 + d2 + d3 + d4 + d5
##
## Fitted party:
## [1] root
## |   [2] d1 <= 33466.09636
## |   |   [3] d1 <= 12823.68919
## |   |   |   [4] d2 <= 7463.10613
## |   |   |   |   [5] d5 <= 6031.6001
## |   |   |   |   |   [6] d5 <= 4871.49: 0.043 (n = 140, err = 0.0)
## |   |   |   |   |   [7] d5 > 4871.49
## |   |   |   |   |   |   [8] d5 <= 5350.7267
## |   |   |   |   |   |   |   [9] d2 <= 5198.89699: 0.086 (n = 14, err = 0.0)
## |   |   |   |   |   |   |   [10] d2 > 5198.89699: 0.065 (n = 15, err = 0.0)
## |   |   |   |   |   |   [11] d5 > 5350.7267: 0.046 (n = 38, err = 0.0)
## |   |   |   |   [12] d5 > 6031.6001: 0.084 (n = 264, err = 0.8)
## |   |   |   [13] d2 > 7463.10613: 0.121 (n = 602, err = 7.2)
## |   |   [14] d1 > 12823.68919: 0.350 (n = 131, err = 12.8)
## |   [15] d1 > 33466.09636
## |   |   [16] d1 <= 48927.30455: 1.114 (n = 85, err = 20.0)
## |   |   [17] d1 > 48927.30455: 1.328 (n = 79, err = 5.9)
##
```

```
## Number of inner nodes:    8
## Number of terminal nodes: 9
```

```r
# we have to make sure all the dates line up here. this way all data
  # begins at 10-02-2017
d1 <- bitcoin$Close[1618:2985]
d2 <- bitcoin$Close[1619:2986]
d3 <- bitcoin$Close[1620:2987]
d4 <- bitcoin$Close[1621:2988]
d5 <- bitcoin$Close[1622:2989]
d6 <- XRP$Close[1520:2887]
d7 <- XRP$Close[1521:2888]
d8 <- XRP$Close[1522:2889]
d9 <- XRP$Close[1523:2890]
d10 <- XRP$Close[1524:2891]
d11 <- cardano$Close[6:1373]

cardano_five_w_bit_xrp <- data.frame(d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11)

library(party)
library(partykit)

card_five_w_bit_xrp_tree <- ctree(d11 ~ d1 + d2 + d3 + d4 + d5 + d6 + d7 + d8 + d9 + d10, data = cardano_f

print(card_five_w_bit_xrp_tree)
```

```
##
## Model formula:
## d11 ~ d1 + d2 + d3 + d4 + d5 + d6 + d7 + d8 + d9 + d10
##
## Fitted party:
## [1] root
## |   [2] d1 <= 33466.09636
## |   |   [3] d10 <= 0.70024
## |   |   |   [4] d3 <= 29374.15189
```

```
## |   |   |   |   [5] d10 <= 0.43473
## |   |   |   |   |   [6] d5 <= 10895.83044
## |   |   |   |   |   |   [7] d5 <= 8988.59621
## |   |   |   |   |   |   |   [8] d10 <= 0.31928
## |   |   |   |   |   |   |   |   [9] d6 <= 0.26868
## |   |   |   |   |   |   |   |   |   [10] d5 <= 8343.27668
## |   |   |   |   |   |   |   |   |   |   [11] d5 <= 6469.79814
## |   |   |   |   |   |   |   |   |   |   |   [12] d5 <= 5014.47998: 0.023 (n = 8, err = 0.0)
## |   |   |   |   |   |   |   |   |   |   |   [13] d5 > 5014.47998: 0.028 (n = 36, err = 0.0)
## |   |   |   |   |   |   |   |   |   |   [14] d5 > 6469.79814
## |   |   |   |   |   |   |   |   |   |   |   [15] d8 <= 0.24378
## |   |   |   |   |   |   |   |   |   |   |   |   [16] d2 <= 6767.31006: 0.028 (n = 19, err = 0.0)
## |   |   |   |   |   |   |   |   |   |   |   |   [17] d2 > 6767.31006: 0.035 (n = 92, err = 0.0)
## |   |   |   |   |   |   |   |   |   |   |   [18] d8 > 0.24378: 0.042 (n = 14, err = 0.0)
## |   |   |   |   |   |   |   |   |   [19] d5 > 8343.27668
## |   |   |   |   |   |   |   |   |   |   [20] d10 <= 0.23453
## |   |   |   |   |   |   |   |   |   |   |   [21] d9 <= 0.21548: 0.052 (n = 7, err = 0.0)
## |   |   |   |   |   |   |   |   |   |   |   [22] d9 > 0.21548: 0.048 (n = 17, err = 0.0)
## |   |   |   |   |   |   |   |   |   |   [23] d10 > 0.23453: 0.043 (n = 12, err = 0.0)
## |   |   |   |   |   |   |   |   [24] d6 > 0.26868: 0.052 (n = 138, err = 0.0)
## |   |   |   |   |   |   |   [25] d10 > 0.31928
## |   |   |   |   |   |   |   |   [26] d4 <= 4871.49
## |   |   |   |   |   |   |   |   |   [27] d5 <= 4017.26846: 0.042 (n = 46, err = 0.0)
## |   |   |   |   |   |   |   |   |   [28] d5 > 4017.26846: 0.047 (n = 17, err = 0.0)
## |   |   |   |   |   |   |   |   [29] d4 > 4871.49: 0.091 (n = 84, err = 0.0)
## |   |   |   |   |   |   [30] d5 > 8988.59621
## |   |   |   |   |   |   |   [31] d7 <= 0.25393: 0.087 (n = 120, err = 0.1)
## |   |   |   |   |   |   |   [32] d7 > 0.25393
## |   |   |   |   |   |   |   |   [33] d7 <= 0.33482
## |   |   |   |   |   |   |   |   |   [34] d2 <= 11354.02422
## |   |   |   |   |   |   |   |   |   |   [35] d10 <= 0.30244: 0.050 (n = 67, err = 0.0)
## |   |   |   |   |   |   |   |   |   |   [36] d10 > 0.30244: 0.060 (n = 24, err = 0.0)
## |   |   |   |   |   |   |   |   |   [37] d2 > 11354.02422: 0.071 (n = 7, err = 0.0)
## |   |   |   |   |   |   |   |   [38] d7 > 0.33482: 0.082 (n = 7, err = 0.0)
```

```
## |   |   |   |   |   [39] d5 > 10895.83044
## |   |   |   |   |   |   [40] d5 <= 18621.31437
## |   |   |   |   |   |   |   [41] d6 <= 0.30994
## |   |   |   |   |   |   |   |   [42] d10 <= 0.28333: 0.109 (n = 69, err = 0.0)
## |   |   |   |   |   |   |   |   [43] d10 > 0.28333: 0.129 (n = 31, err = 0.0)
## |   |   |   |   |   |   |   [44] d6 > 0.30994: 0.070 (n = 23, err = 0.0)
## |   |   |   |   |   |   [45] d5 > 18621.31437: 0.178 (n = 12, err = 0.0)
## |   |   |   |   [46] d10 > 0.43473
## |   |   |   |   |   [47] d10 <= 0.58359
## |   |   |   |   |   |   [48] d4 <= 6721.98
## |   |   |   |   |   |   |   [49] d5 <= 5738.35: 0.056 (n = 7, err = 0.0)
## |   |   |   |   |   |   |   [50] d5 > 5738.35: 0.103 (n = 78, err = 0.1)
## |   |   |   |   |   |   [51] d4 > 6721.98: 0.145 (n = 75, err = 0.1)
## |   |   |   |   |   [52] d10 > 0.58359
## |   |   |   |   |   |   [53] d5 <= 8929.28027
## |   |   |   |   |   |   |   [54] d9 <= 0.69727: 0.208 (n = 38, err = 0.0)
## |   |   |   |   |   |   |   [55] d9 > 0.69727: 0.250 (n = 7, err = 0.0)
## |   |   |   |   |   |   [56] d5 > 8929.28027: 0.157 (n = 13, err = 0.0)
## |   |   |   [57] d3 > 29374.15189: 0.471 (n = 16, err = 1.6)
## |   |   [58] d10 > 0.70024
## |   |   |   [59] d10 <= 1.20498
## |   |   |   |   [60] d4 <= 16624.59961
## |   |   |   |   |   [61] d9 <= 0.95937
## |   |   |   |   |   |   [62] d10 <= 0.80105: 0.266 (n = 16, err = 0.1)
## |   |   |   |   |   |   [63] d10 > 0.80105: 0.317 (n = 40, err = 0.1)
## |   |   |   |   |   [64] d9 > 0.95937
## |   |   |   |   |   |   [65] d9 <= 1.14442: 0.382 (n = 17, err = 0.0)
## |   |   |   |   |   |   [66] d9 > 1.14442: 0.454 (n = 7, err = 0.0)
## |   |   |   |   [67] d4 > 16624.59961: 0.749 (n = 7, err = 1.0)
## |   |   |   [68] d10 > 1.20498
## |   |   |   |   [69] d10 <= 2.39103
## |   |   |   |   |   [70] d9 <= 1.86119: 0.605 (n = 17, err = 0.1)
## |   |   |   |   |   [71] d9 > 1.86119: 0.786 (n = 9, err = 0.0)
## |   |   |   |   [72] d10 > 2.39103: 0.992 (n = 7, err = 0.1)
```

```
## |   [73] d1 > 33466.09636
## |   |   [74] d6 <= 0.39349
## |   |   |   [75] d10 <= 0.29652: 0.343 (n = 13, err = 0.0)
## |   |   |   [76] d10 > 0.29652: 0.472 (n = 7, err = 0.2)
## |   |   [77] d6 > 0.39349
## |   |   |   [78] d6 <= 0.64673
## |   |   |   |   [79] d1 <= 47909.33119: 0.981 (n = 17, err = 0.9)
## |   |   |   |   [80] d1 > 47909.33119: 1.164 (n = 46, err = 0.4)
## |   |   |   [81] d6 > 0.64673: 1.501 (n = 81, err = 4.8)
##
## Number of inner nodes:    40
## Number of terminal nodes: 41
```

Taking our two trees, the first tree gave us a total error of 46.7 while the second tree gave us a total error of 9.6. We can see that the second tree with the addition of the XRP coin has significantly reduced the error in our predicitive ability. In my opinion, this makes sense because Bitcoin did not act like a regular crypto during the peak of the crypto boom. Bitcoin shot far above its counterparts and likely wouldn't be a good price predictor for other currencies on its own.

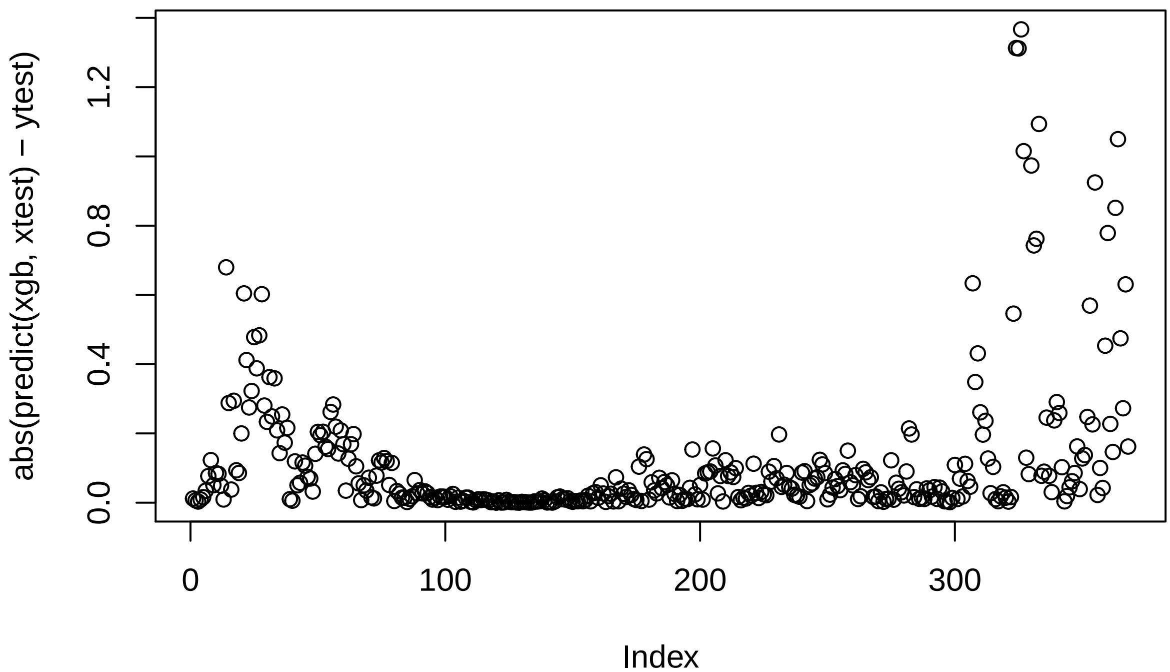
6. Use XGBoost to predict the price of Cardano using the previous 5 days of Bitcoin and XRP

```
# we have to make sure all the dates line up here. this way all data
  # begins at 10-02-2017
d1 <- bitcoin$Close[1618:2985]
d2 <- bitcoin$Close[1619:2986]
d3 <- bitcoin$Close[1620:2987]
d4 <- bitcoin$Close[1621:2988]
d5 <- bitcoin$Close[1622:2989]
d6 <- cardano$Close[6:1373]


cardano_xgb1 <- data.frame(d1, d2, d3, d4, d5, d6)


cardano_xgb1 <- as.matrix(cardano_xgb1)
```

```r
library(xgboost)
```

```
## Warning: package 'xgboost' was built under R version 3.6.2
```

```r
#setup train and test sets
train = sample(1:dim(cardano_xgb1)[1],1000)
test = setdiff(1:dim(cardano_xgb1)[1],train)

xtrain = cardano_xgb1[train,-6]
ytrain = cardano_xgb1[train,6]

xtest = cardano_xgb1[test,-6]
ytest = cardano_xgb1[test,6]

params = list(eta = 0.1, colsample_bylevel = 2/3,
                subsample = 3/4, max_depth = 6, min_child_weigth = 1)
# create xgb model
xgb = xgboost(xtrain, label = ytrain, nrounds = 750,
            params = params, verbose = 0, verbosity = 0)

# plot the absolute value of the error within the test set
plot(abs(predict(xgb, xtest) - ytest))
```

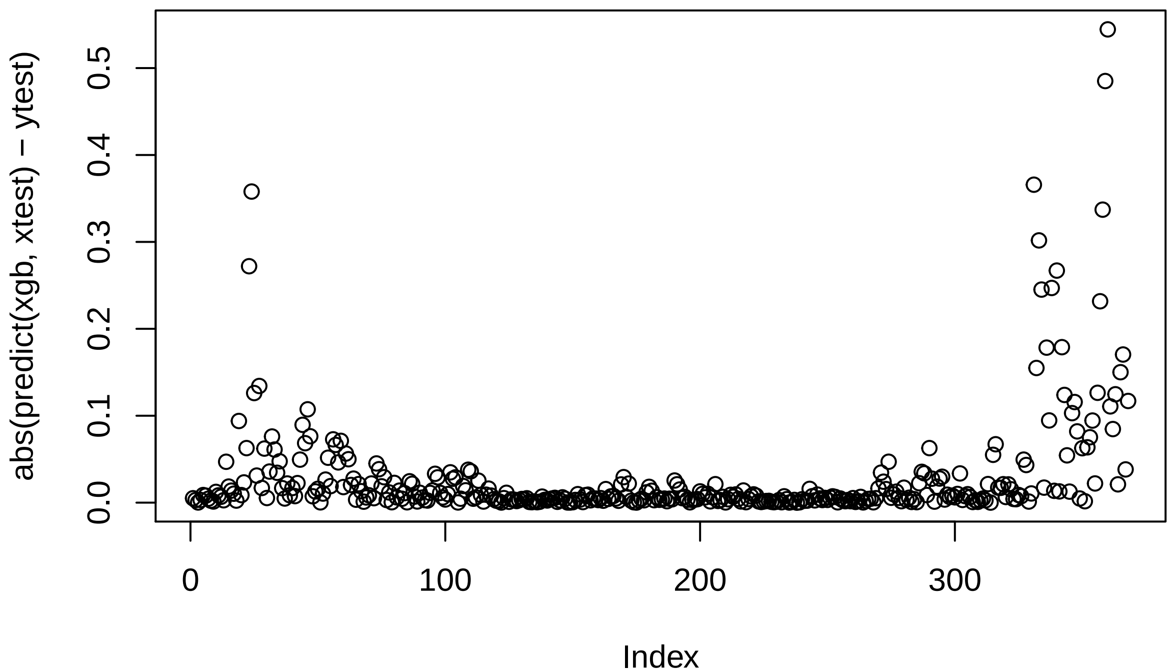



```
total_err <- sum(abs(predict(xgb, xtest) - ytest))
# print the sum of all errors
total_err
```

```
## [1] 42.13917
```

```
# we have to make sure all the dates line up here. this way all data
  # begins at 10-02-2017
d1 <- bitcoin$Close[1618:2985]
d2 <- bitcoin$Close[1619:2986]
d3 <- bitcoin$Close[1620:2987]
d4 <- bitcoin$Close[1621:2988]
d5 <- bitcoin$Close[1622:2989]
d6 <- XRP$Close[1520:2887]
d7 <- XRP$Close[1521:2888]
d8 <- XRP$Close[1522:2889]
d9 <- XRP$Close[1523:2890]
d10 <- XRP$Close[1524:2891]
d11 <- cardano$Close[6:1373]


cardano_xgb <- data.frame(d1, d2, d3, d4, d5, d6, d7, d8, d9, d10, d11)
```

```
cardano_xgb <- as.matrix(cardano_xgb)


library(xgboost)


#setup train and test sets
train = sample(1:dim(cardano_xgb)[1],1000)
test = setdiff(1:dim(cardano_xgb)[1],train)


xtrain = cardano_xgb[train,-11]
ytrain = cardano_xgb[train,11]


xtest = cardano_xgb[test,-11]
ytest = cardano_xgb[test,11]


params = list(eta = 0.1, colsample_bylevel = 2/3,
                subsample = 3/4, max_depth = 6, min_child_weigth = 1)
#create xgb model
xgb = xgboost(xtrain, label = ytrain, nrounds = 750,
            params = params, verbose = 0, verbosity = 0)


#plot the absolute value of all errors
plot(abs(predict(xgb, xtest) - ytest))
```




```r
total_err <- sum(abs(predict(xgb, xtest) - ytest))
# print the sum of all errors
total_err
```

```
## [1] 10.59575
```

We can see that our model did find a quite a large decrease in total error when adding in the values from the XRP coin to the already existing bitcoin model.