ARIMA Modelling

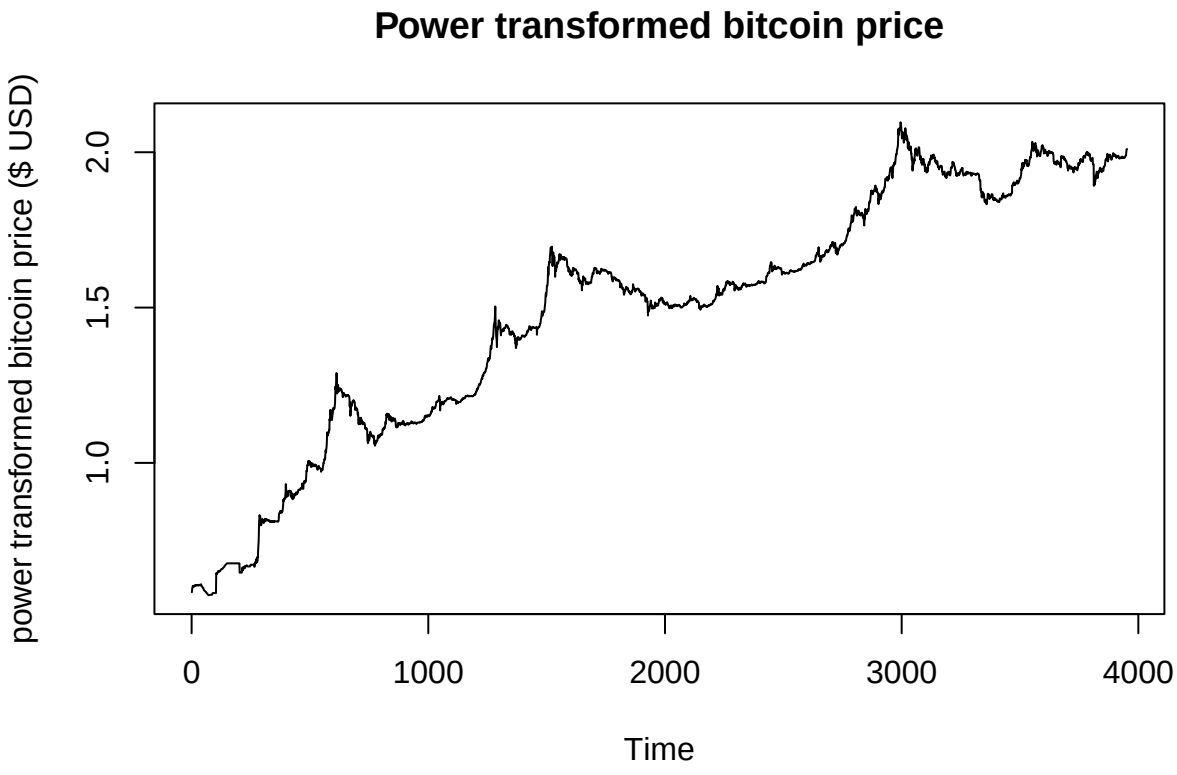
Alan Mathew

```
library(astsa)
```

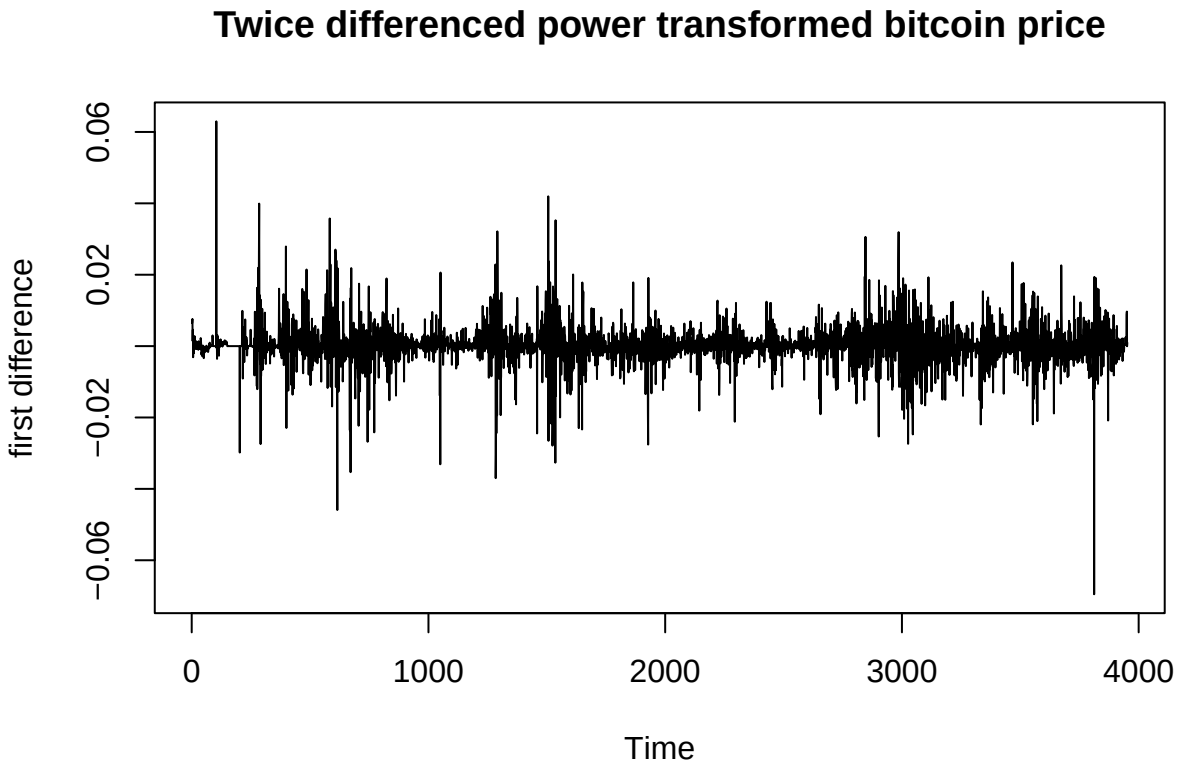
```
## Warning: package 'astsa' was built under R version 4.2.3
```

```
train = read.csv('train.csv')
Y_train = ts(train$price)
pwrY_train = Y_train^0.075
```

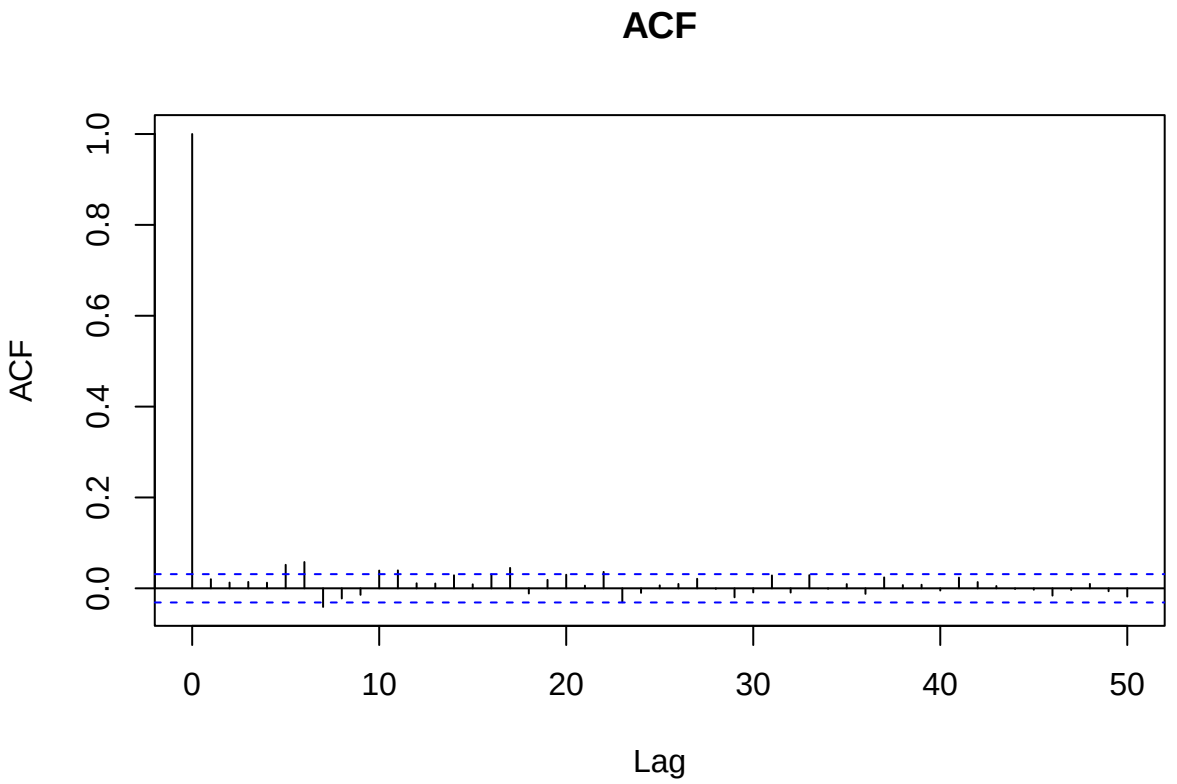
```
plot(pwrY_train, main="Power transformed bitcoin price", ylab="power transformed bitcoin price ($ USD)")
```



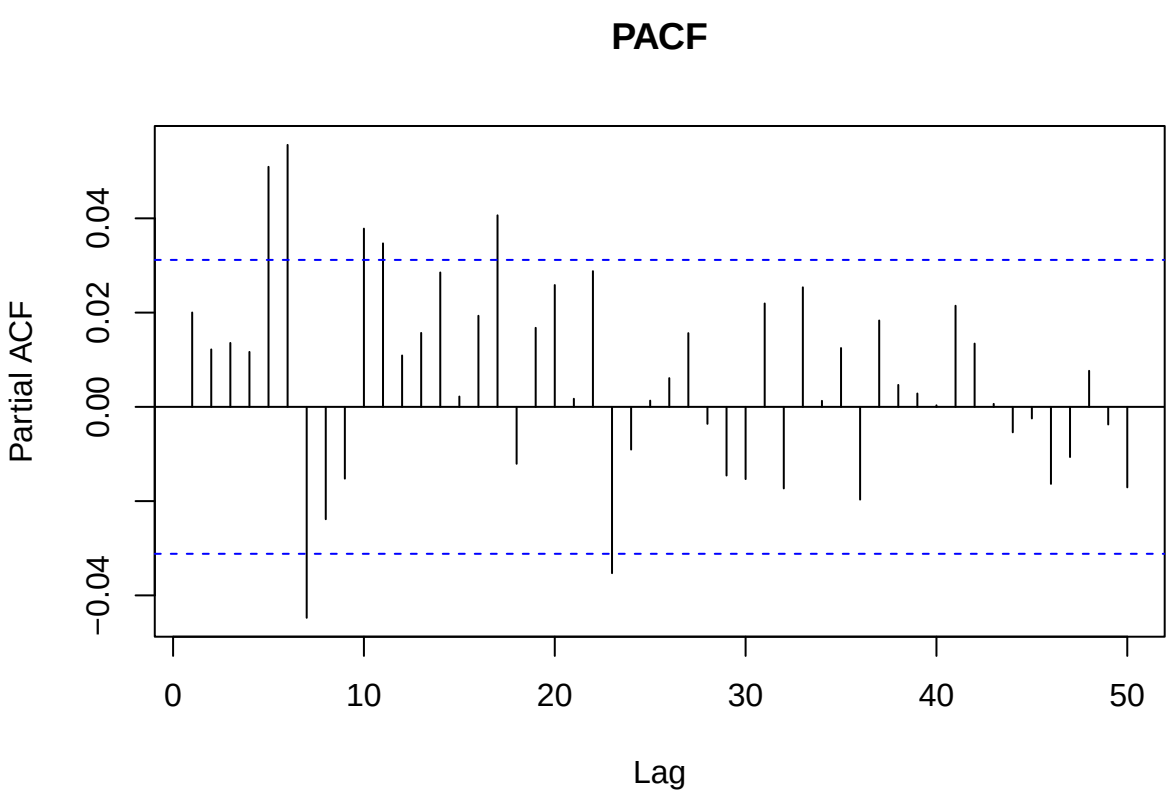

```
diffY_train = diff(pwrY_train)
plot(diffY_train, main="Twice differenced power transformed bitcoin price", ylab="first difference")
```




```
acf(diffY_train, lag.max=50, main="ACF")
```



```r
pacf(diffY_train, lag.max=50, main="PACF")
```



- No clear indication of exponential decay or lag cut-off in ACF and PACF, so best bet is to try a number of ARIMA models.
- So for power transformed data, the following ARIMA models are proposed:

```
##    p d q
## 1  0 1 1
## 2  0 1 2
## 3  0 1 3
## 4  0 1 4
## 5  0 1 5
## 6  0 1 6
## 7  1 1 0
## 8  1 1 1
## 9  1 1 2
## 10 1 1 3
## 11 1 1 4
## 12 1 1 5
## 13 1 1 6
## 14 2 1 0
## 15 2 1 1
## 16 2 1 2
## 17 2 1 3
## 18 2 1 4
## 19 2 1 5
## 20 2 1 6
## 21 3 1 0
```

```
## 22 3 1 1
## 23 3 1 2
## 24 3 1 3
## 25 3 1 4
## 26 3 1 5
## 27 3 1 6
## 28 4 1 0
## 29 4 1 1
## 30 4 1 2
## 31 4 1 3
## 32 4 1 4
## 33 4 1 5
## 34 4 1 6
## 35 5 1 0
## 36 5 1 1
## 37 5 1 2
## 38 5 1 3
## 39 5 1 4
## 40 5 1 5
## 41 5 1 6
## 42 6 1 0
## 43 6 1 1
## 44 6 1 2
## 45 6 1 3
## 46 6 1 4
## 47 6 1 5
## 48 6 1 6
```

```
df = read.csv('/Users/pivaldhingra/Desktop/University courses/STAT 443 project /Data_Group24.csv')
Y = ts(df$price)
pwrY = Y^0.075

test = read.csv('test.csv')
Y_test = ts(test$price)
# pwrY_test = Y_test^0.075
```

```
length(Y_test)
```

```
## [1] 1318
```

```
min_apse = Inf
min_apse_params = c()
best_preds = c()
best_lower = c()
best_upper = c()
for(i in 1:nrow(arima_params)) {
    row = arima_params[i,]
    p = strtoi(row[1,"p"])
    d = strtoi(row[1,"d"])
    q = strtoi(row[1,"q"])

    forecast = astsa::sarima.for(pwrY_train, n.ahead=1318, p, d, q, plot=FALSE)
    lower <- forecast$pred-1.96*forecast$se
```

```
    upper <- forecast$pred+1.96*forecast$se
    pwrY_pred = forecast$pred
    apse = mse(as.vector(Y_test), as.vector(pwrY_pred^(1/0.075)))
    if (apse < min_apse) {
        min_apse = apse
        min_apse_params = row
        best_preds = pwrY_pred
        best_lower = lower
        best_upper = upper
    }
}

print(paste("Minimum APSE: ", min_apse))
```

```
## [1] "Minimum APSE:  2603528011.87693"
```

```
print(min_apse_params)
```

```
##    p d q
## 25 3 1 4
```

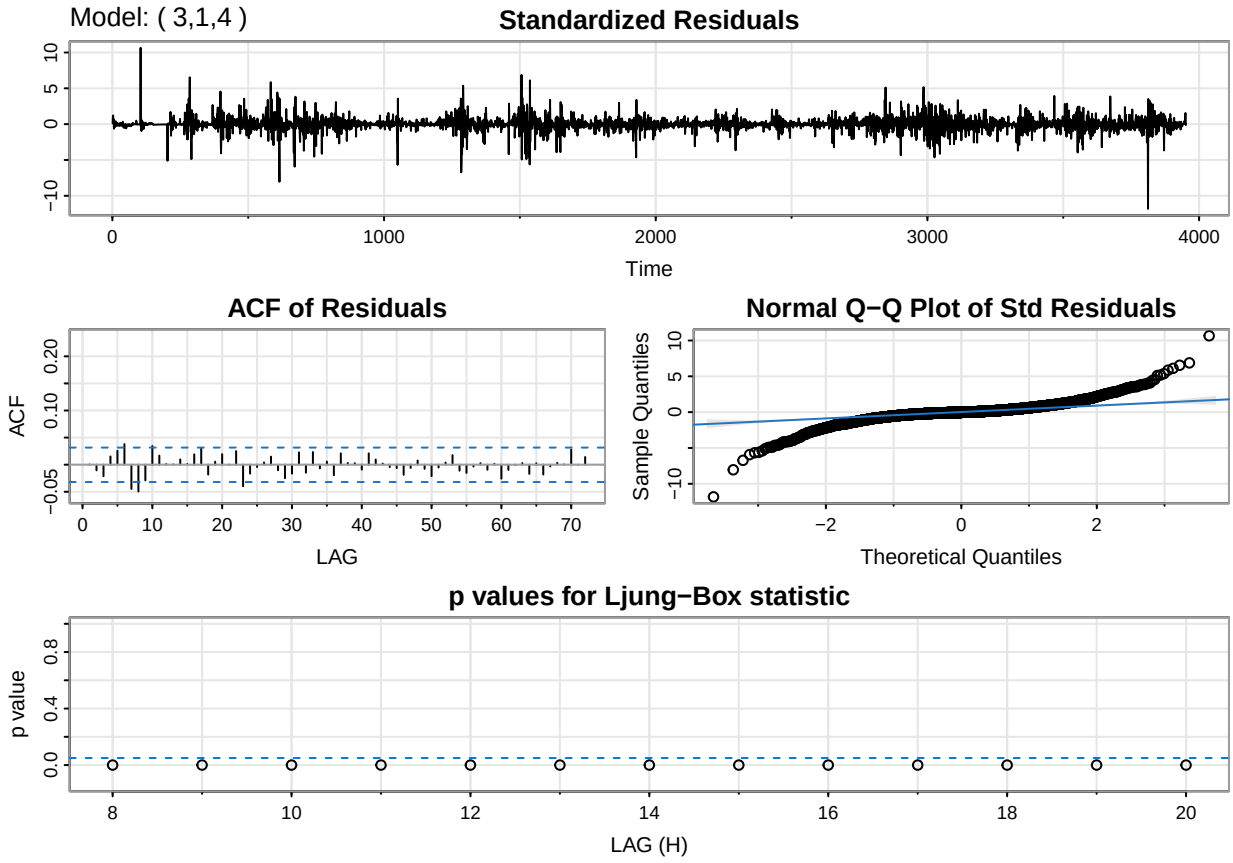
Selected ARIMA model

```
row = min_apse_params[1,]
p = strtoi(row[1,"p"])
d = strtoi(row[1,"d"])
q = strtoi(row[1,"q"])

sarima(pwrY_train, p, d, q)
```

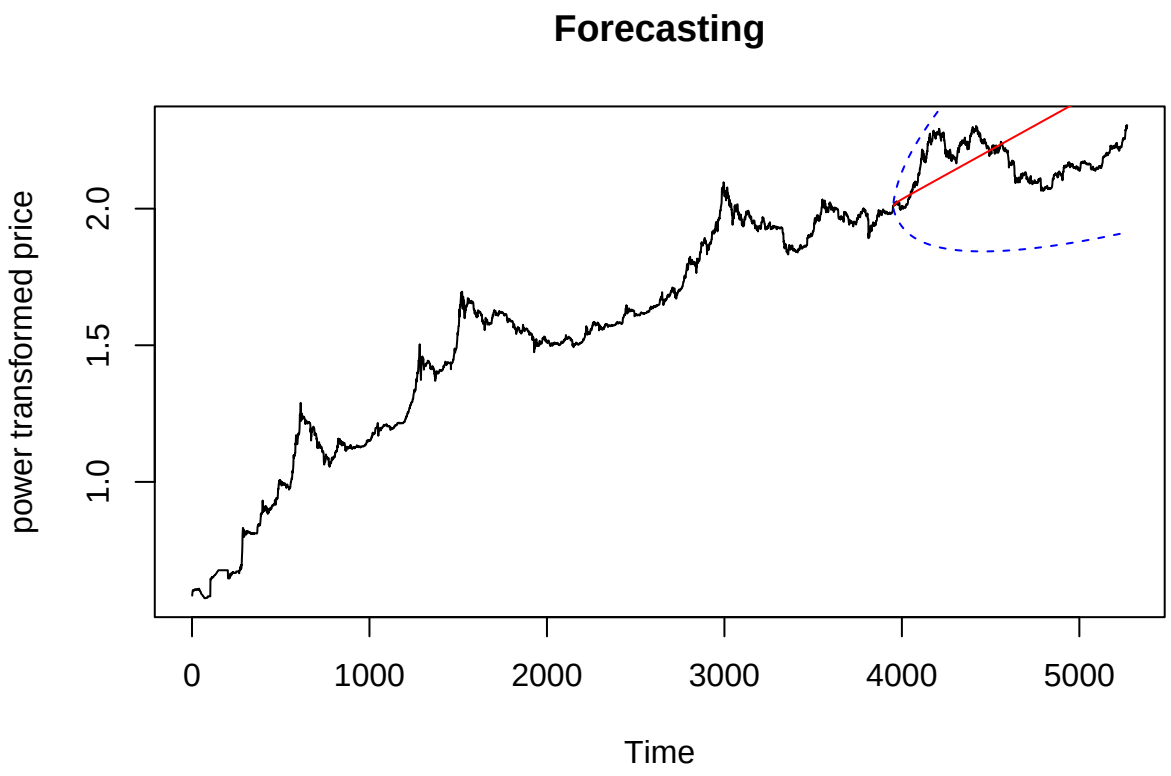
```
## initial  value -5.132617
## iter   2 value -5.132874
## iter   3 value -5.133019
## iter   4 value -5.133021
## iter   5 value -5.133024
## iter   6 value -5.133033
## iter   7 value -5.133057
## iter   8 value -5.133128
## iter   9 value -5.133304
## iter  10 value -5.133407
## iter  11 value -5.133438
## iter  12 value -5.133469
## iter  13 value -5.133646
## iter  14 value -5.133704
## iter  15 value -5.133709
## iter  16 value -5.133715
## iter  17 value -5.133746
## iter  18 value -5.133847
## iter  19 value -5.133882
## iter  20 value -5.133957
```

```
## iter  21 value -5.134019
## iter  22 value -5.134122
## iter  23 value -5.134152
## iter  24 value -5.134154
## iter  25 value -5.134154
## iter  26 value -5.134155
## iter  27 value -5.134164
## iter  28 value -5.134164
## iter  28 value -5.134164
## iter  28 value -5.134164
## final  value -5.134164
## converged
## initial  value -5.134487
## iter   2 value -5.134488
## iter   3 value -5.134490
## iter   4 value -5.134493
## iter   5 value -5.134526
## iter   6 value -5.134553
## iter   7 value -5.134580
## iter   8 value -5.134588
## iter   9 value -5.134590
## iter  10 value -5.134591
## iter  11 value -5.134591
## iter  12 value -5.134592
## iter  13 value -5.134594
## iter  14 value -5.134598
## iter  15 value -5.134606
## iter  16 value -5.134615
## iter  17 value -5.134624
## iter  18 value -5.134627
## iter  19 value -5.134627
## iter  20 value -5.134628
## iter  21 value -5.134631
## iter  22 value -5.134662
## iter  23 value -5.134674
## iter  24 value -5.134679
## iter  25 value -5.134680
## iter  26 value -5.134681
## iter  27 value -5.134682
## iter  28 value -5.134688
## iter  29 value -5.134701
## iter  30 value -5.134730
## iter  31 value -5.134762
## iter  32 value -5.134812
## iter  33 value -5.134827
## iter  34 value -5.134828
## iter  35 value -5.134829
## iter  36 value -5.134830
## iter  37 value -5.134830
## iter  38 value -5.134831
## iter  39 value -5.134832
## iter  40 value -5.134835
## iter  41 value -5.134852
## iter  42 value -5.134889
```

```
## iter  43 value -5.134915
## iter  44 value -5.134948
## iter  45 value -5.134951
## iter  46 value -5.134953
## iter  47 value -5.134954
## iter  48 value -5.134976
## iter  49 value -5.135004
## iter  50 value -5.135036
## iter  51 value -5.135083
## iter  52 value -5.135115
## iter  53 value -5.135137
## iter  54 value -5.135141
## iter  54 value -5.135141
## iter  54 value -5.135141
## final  value -5.135141
## converged

## Warning in sqrt(diag(fitit$var.coef)): NaNs produced

## Warning in sqrt(diag(fitit$var.coef)): NaNs produced

## <><><><><><><><><><><><><><><>
##
## Coefficients:
##          Estimate     SE t.value p.value
## ar1       -0.1119    NaN     NaN     NaN
## ar2        0.2664 0.0857  3.1080  0.0019
## ar3        0.7448    NaN     NaN     NaN
## ma1        0.1291    NaN     NaN     NaN
## ma2       -0.2444 0.0927 -2.6351  0.0084
## ma3       -0.7168    NaN     NaN     NaN
## ma4       -0.0195 0.0168 -1.1627  0.2450
## constant   0.0003 0.0001  2.4064  0.0162
##
## sigma^2 estimated as 3.464723e-05 on 3943 degrees of freedom
##
## AIC = -7.427848  AICc = -7.427839  BIC = -7.413539
##
```




```
plot(pwrY, main="Forecasting", ylab="power transformed price")
lines(pwrY_pred,col='red',type='l',pch='*')
lines(lower,col='blue',lty=2)
lines(upper,col='blue',lty=2)
```




```
plot(Y, main="Forecasting", ylab="price")
lines(pwrY_pred^(1/0.075),col='red',type='l',pch='*')
lines(lower^(1/0.075),col='blue',lty=2)
lines(upper^(1/0.075),col='blue',lty=2)
```