Sam's Remarkedly Basic Understanding of Time Series Analysis

Samuel Holt

April 9, 2019

Contents

Importing and Converting to Time Series **2**

Packages . . . 2
STEP 1 . . . 2
STEP 2 . . . 2
STEP 3 . . . 2

Fitting a Model **2**

STEP 4a . . . 2
STEP 4b . . . 3
STEP 4b . . . 3

Evaluating the Fitted Model **3**

STEP 5 . . . 3

Forecast with a Fitted Model **4**

STEP 6a . . . 4
STEP 6b . . . 4
STEP 6c . . . 4

AutoCorrelation **4**

Autocorrelation Coefficient . . . 4
AutoCorrelation Functions . . . 5

Transformation **5**

BoxCox Transformation . . . 5
Log Transformation . . . 5
DickeyFuller Test . . . 5
Differencing . . . 6

Simulating Models **6**

ARIMA Models . . . 6
Visualise Simulation . . . 6

Importing and Converting to Time Series

Packages

```
library(TSA)
library(tseries)
library(fUnitRoots)
```

STEP 1

Import Data

```
data <- read.csv('data.csv', header = T/F)
```

STEP 2

Convert Data

```
data_ts <- ts(data,
              start = c(year, month),
              end = c(year, month),
              frequency = OBSperUNIT)
```

STEP 3

Investigate Time Series with Visuals

```
#  Trend
#   Seasonality
#   Change in Variance
#   Internation Point
#   Behavior (AR/MA) >>> (Smooth/Jaggered)
plot(data_ts, type = 'b')
```

Fitting a Model

STEP 4a

FIT LINEAR MODEL

```
t <- time(data_ts)
lin_mod <- lm(data_ts ~ t)
summary(lin_mod)
# look at sig vals for coeffs and Rsqu value for explanability
```

STEP 4b

FIT QUADRATIC MODEL

```
t <- time(data_ts)
t2 <- t^2
quad_mod <- lm(data_ts ~ t + t2)
summary(quad_mod)
```

STEP 4b

FIT HARMONIC MODEL

```
har. <- harmonic(data_ts, 1)
har_mod <- lm(data_ts ~ har.)
summary(har_mod)
```

Evaluating the Fitted Model

STEP 5

Standardise Residuals

```
res <- rstudent(model)
```

Investigate White Noise Characteristics

```
plot(res)
```

Investigate Normality of Residuals

```
qqnorm(res)
qqline(res)
hist(res)
shapiro.test(res) # provides a p value for normality
```

Forecast with a Fitted Model

STEP 6a

FORECAST LINEAR MODEL

```
h <- 5 # 5 steps into the future
t <- seq(max(time(data_ts) + 1, max(time(data_ts) + h, 1)))
new <- data.frame(t)
pred <- predict(lin_mod, new, interval = 'prediction')
```

STEP 6b

FORECAST QUADRATIC MODEL

```
h <- 5 # 5 steps in the future
t <- seq(max(time(data_ts) + 1, max(time(data_ts) + h, 1)))
t2 <- t^2
new <- data.frame(t, t2)
pred <- predict(quad_mod, new, interval = 'prediction')
```

STEP 6c

FORECAST HARMONIC MODEL

```
h <- 5 # 5 steps into the future
t <- seq(max(time(data_ts) + 1, max(time(data_ts) + h, 1)))
t_cos <- cos(2*pi*t)
t_sin <- sin(2*pi*t)
new <- data.frame(t_cos, t_sin)
pred <- predict(har_mod, new, interval = 'prediction')
```

AutoCorrelation

Autocorrelation Coefficient

```
y <- data
x <- zlag(data)
index <- 2:length(x)
cor(y[index], x[index])
```

AutoCorrelation Functions

ACF

```
acf(data) # gives q of ARMA(q, p), a pattern means just AR(p)
```

PACF

```
pacf(data) # gives p of ARMA(q, p), a pattern means just MA(q)
```

EACF

```
eacf(data) # top left and the closest group provides possible models
```

Transformation

BoxCox Transformation

```
data_ts_tran = BoxCox.ar(data_ts) # if lambda = 0, go with log transformation, if lambda =/= 0, BoxCox
data_ts_tran$ci
lambda =  # median of $ci values
data_ts_boxcox = (data_ts ^ lambda - 1) / lambda
```

Log Transformation

```
data_ts_log <- log(data_ts)
```

DickeyFuller Test

```
# find k for lags in adf test
x <- ar(diff(data_ts))
k <- x$order
adfTest(data_ts,
        lags = k,
        type = 'c') # type argument can be:
                    # regression no constant ('nc')
                    # regression with constant ('c')
                    # regression with constant but no time trend ('ct') default is 'c'

# Alternative ADF Test
adf.test(gold,
         alternative = c('stationary', 'explosive'),
         k = trunc((length(gold) - 1) ^ (1 / 3)))
```

Differencing

```
data_ts_diff <- diff(data_ts, differences = ) # set differences, if ADF test fails to reject Null, set
```

Simulating Models

ARIMA Models

```
set.seed(1234) # have the randomisation be repeatable
n = 100 # 100 data points
arima1 <- arima.sim(model = list(order(2,0,2), ar = 0.5, ma = 0.5),
                    n = n)
```

Visualise Simulation

```
plot(arima1)
acf(arima1)
pacf(arima1)
eacf(arima1) # 0, 0 valid model
```