Timeseries-toolbox.R

Alexandros

2022-10-23

```
# Fable package ------------------------------------------------------------

#https://cran.r-project.org/web/packages/fable/vignettes/fable.html

library(fable)
```

```
## Warning: package 'fable' was built under R version 4.1.3
```

```
## Loading required package: fabletools
```

```
## Warning: package 'fabletools' was built under R version 4.1.3
```

```
library(tsibble)
```

```
## Warning: package 'tsibble' was built under R version 4.1.3
```

```
##
## Attaching package: 'tsibble'
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, union
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

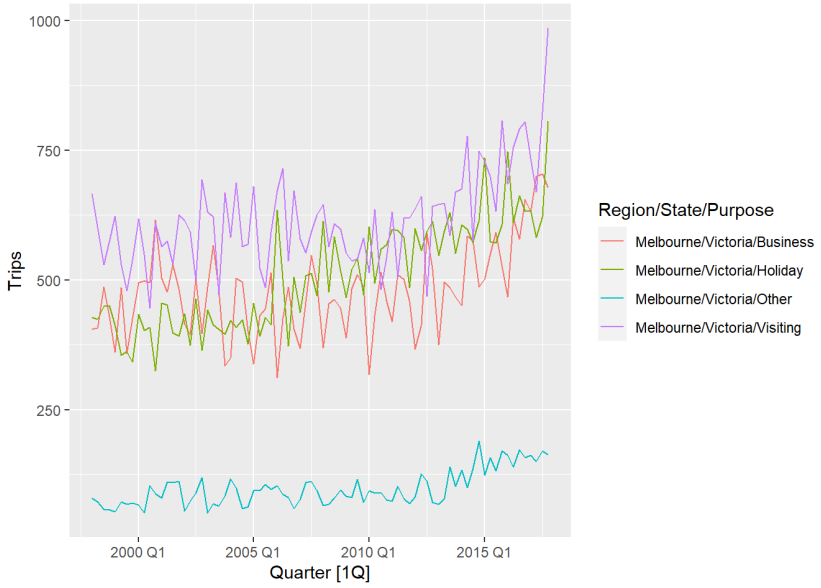
```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
#tsibble::tourism data set

tourism_melb <- tourism %>%
  filter(Region == "Melbourne")

tourism_melb %>%
  group_by(Purpose)  %>% head
```

```
## # A tsibble: 6 x 5 [1Q]
## # Key:       Region, State, Purpose [1]
## # Groups:    Purpose [1]
##   Quarter Region    State    Purpose  Trips
##     <qtr> <chr>     <chr>    <chr>    <dbl>
## 1 1998 Q1 Melbourne Victoria Business  405.
## 2 1998 Q2 Melbourne Victoria Business  408.
## 3 1998 Q3 Melbourne Victoria Business  486.
## 4 1998 Q4 Melbourne Victoria Business  429.
## 5 1999 Q1 Melbourne Victoria Business  361.
## 6 1999 Q2 Melbourne Victoria Business  486.
```

```
#The variable that we'd like to estimate is the number of overnight trips (in thousands)
#represented by the Trips variable

tourism_melb %>%
  autoplot(Trips)
```




```
### about ets https://math.unm.edu/~lil/Stat581/8-ets.pdf

#fit an ets and an arima model

fit <- tourism_melb %>%
  model(
    ets = ETS(Trips ~ trend("A")),
    arima = ARIMA(Trips)
  )
fit
```

```
## # A mable: 4 x 5
## # Key:     Region, State, Purpose [4]
##   Region    State    Purpose          ets                           arima
##   <chr>     <chr>    <chr>         <model>                        <model>
## 1 Melbourne Victoria Business <ETS(A,A,A)> <ARIMA(0,1,2)(1,0,1)[4] w/ drift>
## 2 Melbourne Victoria Holiday  <ETS(M,A,A)>          <ARIMA(0,1,1) w/ drift>
## 3 Melbourne Victoria Other    <ETS(A,A,N)>          <ARIMA(0,1,1) w/ drift>
## 4 Melbourne Victoria Visiting <ETS(M,A,A)>         <ARIMA(0,1,1)(1,0,2)[4]>
```

```
fit %>% class
```

```
## [1] "mdl_df"     "tbl_df"     "tbl"        "data.frame"
```

```
#A mable contains a row for each time series (uniquely identified by the key variables),
#and a column for each model specification

fit %>%
  select(Region, State, Purpose, arima) %>%
  coef()
```

```
## # A tibble: 13 x 9
##    Region   State   Purpose  .model term   estimate std.error statistic  p.value
##    <chr>    <chr>   <chr>    <chr>  <chr>     <dbl>     <dbl>     <dbl>    <dbl>
##  1 Melbour~ Victor~ Business arima  ma1      -0.555    0.130     -4.28  5.29e- 5
##  2 Melbour~ Victor~ Business arima  ma2      -0.233    0.129     -1.81  7.47e- 2
##  3 Melbour~ Victor~ Business arima  sar1      0.946    0.0634    14.9   1.08e-24
##  4 Melbour~ Victor~ Business arima  sma1     -0.772    0.145     -5.34  8.81e- 7
##  5 Melbour~ Victor~ Business arima  const~    0.192    0.213      0.903 3.69e- 1
##  6 Melbour~ Victor~ Holiday  arima  ma1      -0.931    0.0851   -10.9   1.77e-17
##  7 Melbour~ Victor~ Holiday  arima  const~    3.65     0.571      6.39  1.06e- 8
##  8 Melbour~ Victor~ Other    arima  ma1      -0.750    0.0708   -10.6   8.19e-17
##  9 Melbour~ Victor~ Other    arima  const~    1.24     0.640      1.93  5.70e- 2
## 10 Melbour~ Victor~ Visiting arima  ma1      -0.838    0.0652   -12.8   5.03e-21
## 11 Melbour~ Victor~ Visiting arima  sar1      0.659    0.193      3.41  1.03e- 3
## 12 Melbour~ Victor~ Visiting arima  sma1     -0.402    0.206     -1.95  5.47e- 2
## 13 Melbour~ Victor~ Visiting arima  sma2      0.322    0.143      2.26  2.68e- 2
```

```
fit %>% glance()
```

```
## # A tibble: 8 x 14
##   Region   State   Purpose .model   sigma2 log_lik   AIC  AICc   BIC   MSE  AMSE
##   <chr>    <chr>   <chr>   <chr>     <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Melbour~ Victor~ Busine~ ets     3.53e+3   -498. 1014. 1016. 1035. 3180. 3520.
## 2 Melbour~ Victor~ Busine~ arima   3.67e+3   -435.  882.  883.  896.   NA    NA
## 3 Melbour~ Victor~ Holiday ets     1.10e-2   -487.  992.  994. 1013. 2548. 2574.
## 4 Melbour~ Victor~ Holiday arima   3.07e+3   -429.  864.  865.  872.   NA    NA
## 5 Melbour~ Victor~ Other   ets     4.97e+2   -422.  853.  854.  865.  472.  512.
## 6 Melbour~ Victor~ Other   arima   4.89e+2   -356.  718.  719.  725.   NA    NA
## 7 Melbour~ Victor~ Visiti~ ets     1.09e-2   -503. 1024. 1026. 1045. 3714. 3860.
## 8 Melbour~ Victor~ Visiti~ arima   4.24e+3   -442.  893.  894.  905.   NA    NA
## # ... with 3 more variables: MAE <dbl>, ar_roots <list>, ma_roots <list>
```

```
#If you're working with a single model (or want to look at one model in particular), the report()


fit %>%
  filter(Purpose == "Holiday") %>%
  select(ets) %>%
  report()
```

```
## Series: Trips
## Model: ETS(M,A,A)
##   Smoothing parameters:
##     alpha = 0.03084501
##     beta  = 0.03084499
##     gamma = 0.0001000967
##
##   Initial states:
##      l[0]      b[0]     s[0]    s[-1]     s[-2]    s[-3]
##  424.0777 -2.535481 -26.7441 4.256618 -10.10668 32.59417
##
##   sigma^2:  0.011
##
##       AIC      AICc       BIC
##  991.7305  994.3020 1013.1688
```

```
#the augment() function may be more convenient, which provides the original
#data along with both fitted values and their residuals.

fit %>%
  augment()
```

```
## # A tsibble: 640 x 9 [1Q]
## # Key:       Region, State, Purpose, .model [8]
##    Region    State    Purpose  .model Quarter Trips .fitted  .resid  .innov
##    <chr>     <chr>    <chr>    <chr>    <qtr> <dbl>   <dbl>   <dbl>   <dbl>
##  1 Melbourne Victoria Business ets    1998 Q1  405.    396.    9.54    9.54
##  2 Melbourne Victoria Business ets    1998 Q2  408.    483.  -75.1   -75.1
##  3 Melbourne Victoria Business ets    1998 Q3  486.    487.   -1.13   -1.13
##  4 Melbourne Victoria Business ets    1998 Q4  429.    454.  -25.2   -25.2
##  5 Melbourne Victoria Business ets    1999 Q1  361.    391.  -30.3   -30.3
##  6 Melbourne Victoria Business ets    1999 Q2  486.    466.   19.9    19.9
##  7 Melbourne Victoria Business ets    1999 Q3  359.    492. -133.   -133.
##  8 Melbourne Victoria Business ets    1999 Q4  426.    424.    1.49    1.49
##  9 Melbourne Victoria Business ets    2000 Q1  495.    364.  130.    130.
## 10 Melbourne Victoria Business ets    2000 Q2  499.    477.   22.0    22.0
## # ... with 630 more rows
```

```
#arrange arcoding to MASE

fit %>%
  accuracy() %>%
  arrange(MASE)
```

```
## # A tibble: 8 x 13
##   Region  State Purpose .model .type     ME  RMSE   MAE    MPE  MAPE  MASE RMSSE
##   <chr>   <chr> <chr>   <chr>  <chr>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 Melbou~ Vict~ Holiday ets    Trai~  4.67   50.5  37.2  0.217  7.29 0.675 0.697
## 2 Melbou~ Vict~ Busine~ ets    Trai~  3.31   56.4  42.9 -0.753  9.31 0.691 0.740
## 3 Melbou~ Vict~ Busine~ arima  Trai~  2.54   58.2  46.0 -1.17  10.1  0.741 0.765
## 4 Melbou~ Vict~ Holiday arima  Trai~ -4.64   54.3  41.4 -2.44   8.46 0.752 0.751
## 5 Melbou~ Vict~ Other   arima  Trai~ -0.344  21.7  17.0 -6.16  19.5  0.763 0.772
## 6 Melbou~ Vict~ Other   ets    Trai~ -0.142  21.7  17.0 -5.97  19.6  0.767 0.773
## 7 Melbou~ Vict~ Visiti~ ets    Trai~  8.17   60.9  51.4  0.433  8.28 0.819 0.782
## 8 Melbou~ Vict~ Visiti~ arima  Trai~  6.89   63.1  51.7  0.106  8.44 0.825 0.809
## # ... with 1 more variable: ACF1 <dbl>
```

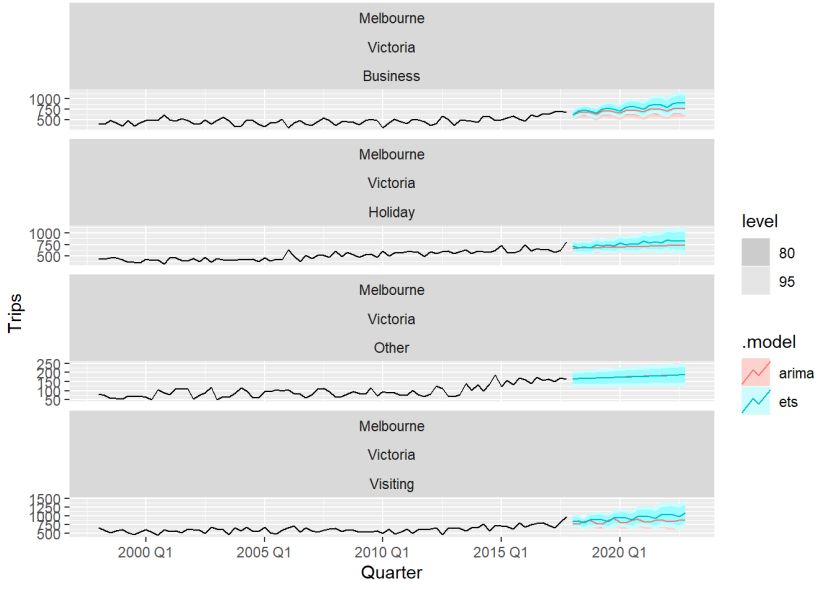
```
#Forecasts from these models can be produced directly as our specified
#models do not require any additional data.


fc <- fit %>%
  forecast(h = "5 years")
fc
```

```
## # A fable: 160 x 7 [1Q]
## # Key:     Region, State, Purpose, .model [8]
##    Region    State    Purpose  .model Quarter        Trips .mean
##    <chr>     <chr>    <chr>    <chr>    <qtr>       <dist> <dbl>
##  1 Melbourne Victoria Business ets    2018 Q1 N(619, 3533)  619.
##  2 Melbourne Victoria Business ets    2018 Q2 N(709, 3766)  709.
##  3 Melbourne Victoria Business ets    2018 Q3 N(738, 4042)  738.
##  4 Melbourne Victoria Business ets    2018 Q4 N(713, 4364)  713.
##  5 Melbourne Victoria Business ets    2019 Q1 N(664, 4735)  664.
##  6 Melbourne Victoria Business ets    2019 Q2 N(755, 5159)  755.
##  7 Melbourne Victoria Business ets    2019 Q3 N(784, 5640)  784.
##  8 Melbourne Victoria Business ets    2019 Q4 N(759, 6181)  759.
##  9 Melbourne Victoria Business ets    2020 Q1 N(710, 6786)  710.
## 10 Melbourne Victoria Business ets    2020 Q2 N(800, 7458)  800.
## # ... with 150 more rows
```

```
#The resulting forecasts are contained in a "fable" (forecast table),

#Confidence intervals can be extracted from the distribution using the hilo() function.

fc %>%
  hilo(level = c(80, 95))
```

```
## # A tsibble: 160 x 9 [1Q]
## # Key:       Region, State, Purpose, .model [8]
##    Region State Purpose .model Quarter        Trips .mean                `80%`
##    <chr>  <chr> <chr>   <chr>    <qtr>       <dist> <dbl>               <hilo>
##  1 Melbo~ Vict~ Busine~ ets    2018 Q1 N(619, 3533)  619. [542.3864, 694.7363]80
##  2 Melbo~ Vict~ Busine~ ets    2018 Q2 N(709, 3766)  709. [630.4443, 787.7426]80
##  3 Melbo~ Vict~ Busine~ ets    2018 Q3 N(738, 4042)  738. [656.9891, 819.9425]80
##  4 Melbo~ Vict~ Busine~ ets    2018 Q4 N(713, 4364)  713. [628.7879, 798.1010]80
##  5 Melbo~ Vict~ Busine~ ets    2019 Q1 N(664, 4735)  664. [575.8480, 752.2196]80
##  6 Melbo~ Vict~ Busine~ ets    2019 Q2 N(755, 5159)  755. [662.5141, 846.6176]80
##  7 Melbo~ Vict~ Busine~ ets    2019 Q3 N(784, 5640)  784. [687.6922, 880.1843]80
##  8 Melbo~ Vict~ Busine~ ets    2019 Q4 N(759, 6181)  759. [658.1603, 859.6735]80
##  9 Melbo~ Vict~ Busine~ ets    2020 Q1 N(710, 6786)  710. [603.9336, 815.0789]80
## 10 Melbo~ Vict~ Busine~ ets    2020 Q2 N(800, 7458)  800. [689.3620, 910.7146]80
## # ... with 150 more rows, and 1 more variable: 95% <hilo>
```

```
fc %>%
  autoplot(tourism_melb)
```



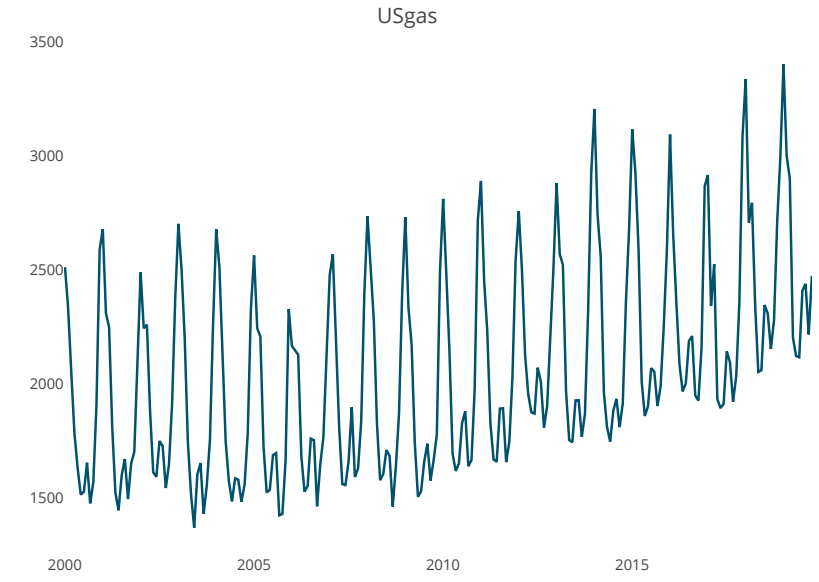
```
# TS studio ------------------------------------------------------------
#https://cran.r-project.org/web/packages/TSstudio/vignettes/Plotting_Time_Series.html


pacman::p_load(TSstudio)

data(USgas)
USgas %>% class
```

```
## [1] "ts"
```

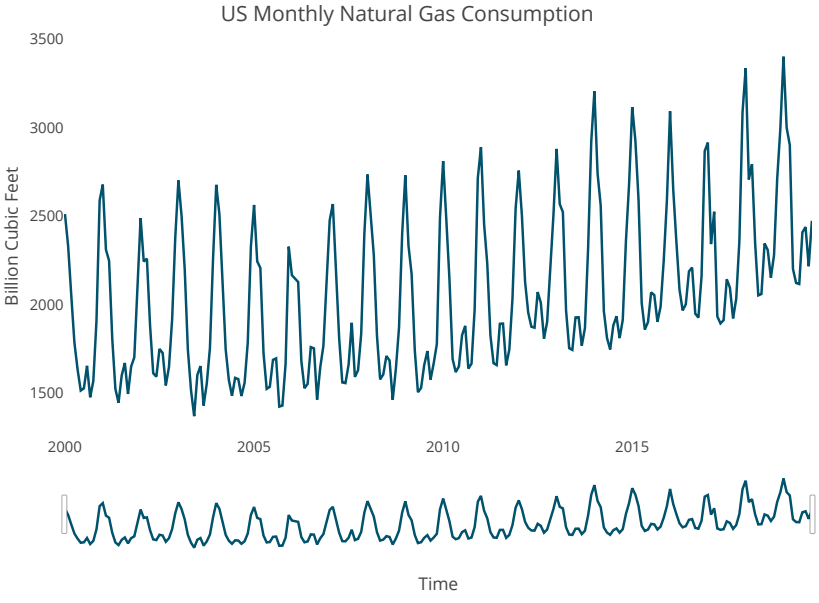
```
ts_info(USgas)
```

```
##  The USgas series is a ts object with 1 variable and 238 observations
##  Frequency: 12
##  Start time: 2000 1
##  End time: 2019 10
```

```
ts_plot(USgas)
```




```
ts_plot(USgas,
        title = "US Monthly Natural Gas Consumption",
        Xtitle = "Time",
        Ytitle = "Billion Cubic Feet",
        slider = TRUE)
```



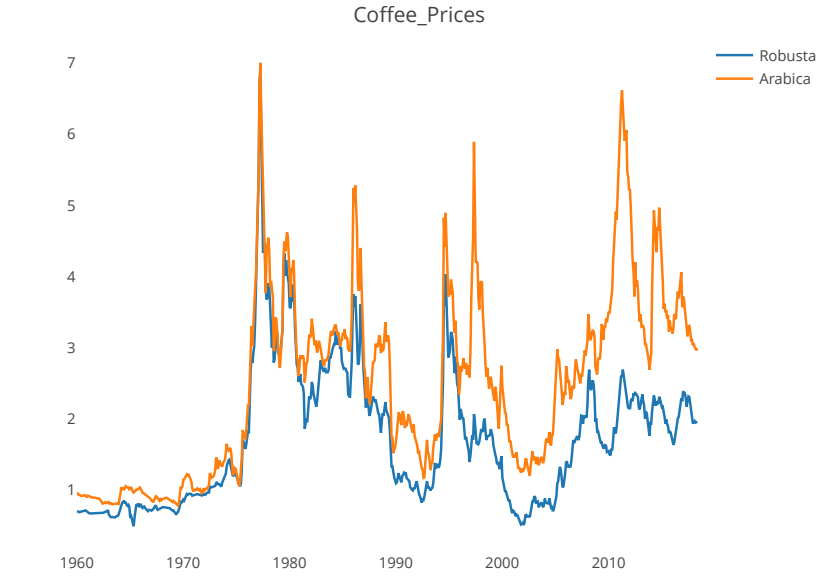
```
## multiple ts plot
data("Coffee_Prices")
Coffee_Prices %>% head
```

```
##        Robusta Arabica
## [1,] 0.6968643  0.9409
## [2,] 0.6887074  0.9469
## [3,] 0.6887074  0.9281
## [4,] 0.6845187  0.9303
## [5,] 0.6906915  0.9200
## [6,] 0.6968643  0.9123
```

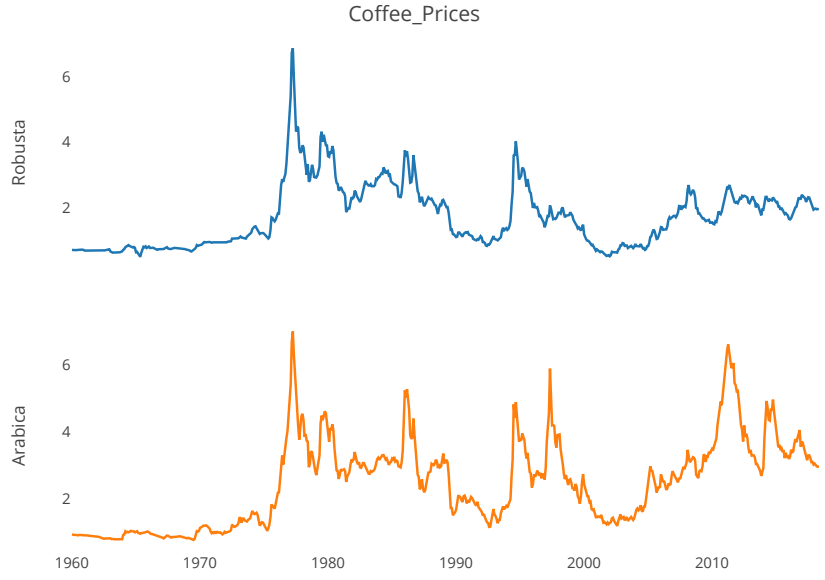
```
ts_info(Coffee_Prices)
```

```
##  The Coffee_Prices series is a mts object with 2 variables and 701 observations
##  Frequency: 12
##  Start time: 1960 1
##  End time: 2018 5
```

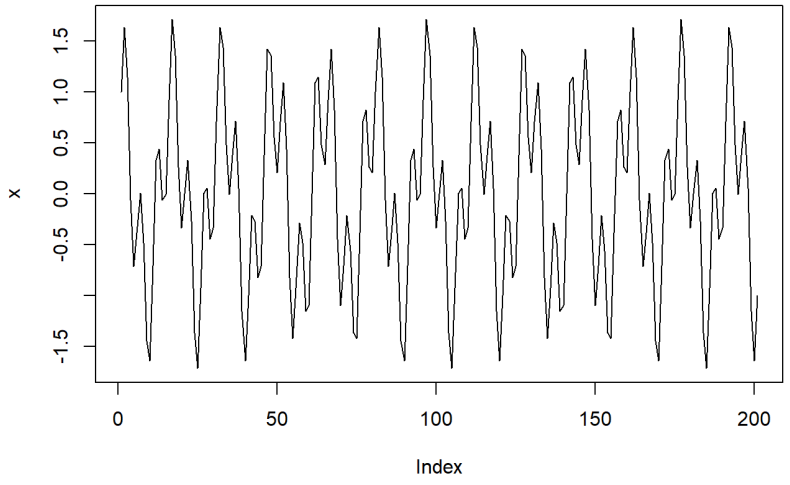
```
ts_plot(Coffee_Prices)
```



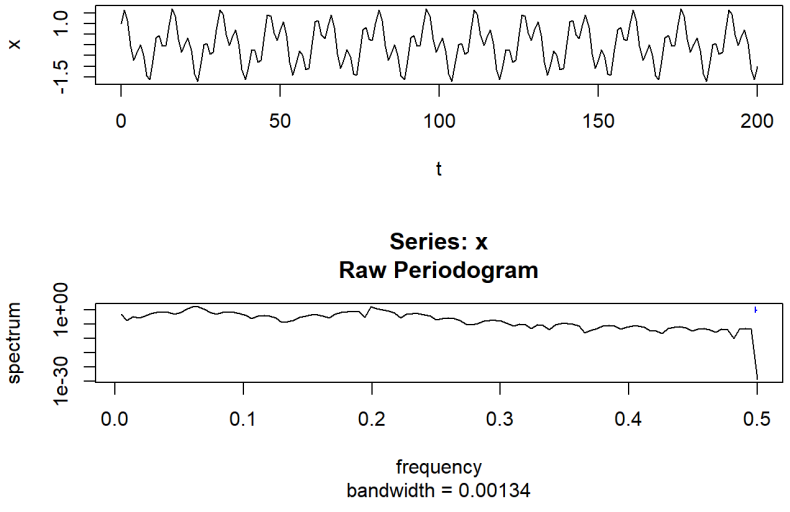

```
## Plot on different plots

ts_plot(Coffee_Prices,
        type = "multiple")
```



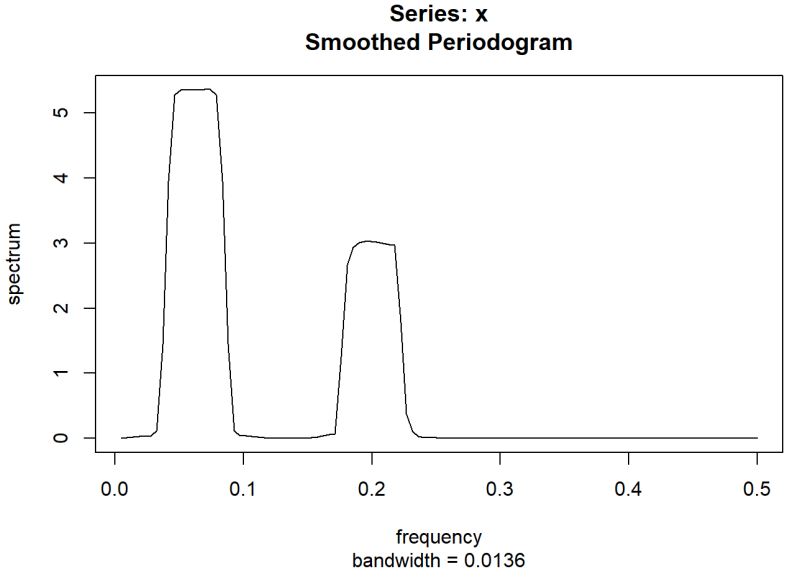
```
# Spectral analysis ---------------------------------------------------------


#https://math.mcmaster.ca/~bolker/eeid/2010/Ecology/Spectral.pdf


#Simple Example

t <- seq(0,200,by=1)
x <- cos(2*pi*t/16) + 0.75*sin(2*pi*t/5)

plot(x,type="l")
```



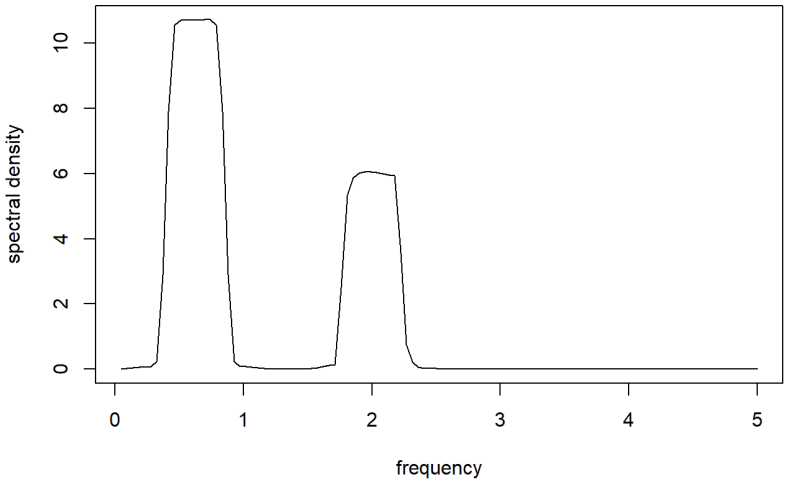

```
par(mfrow=c(2,1))
plot(t,x,'l')
spectrum(x)
```



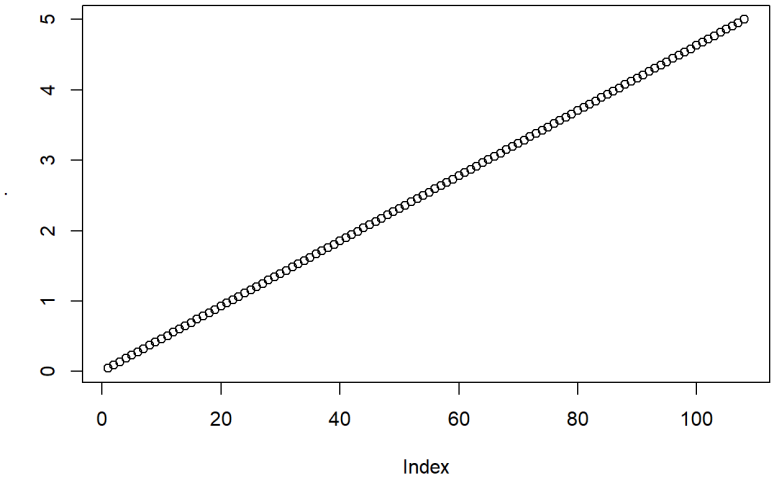
```
par(mfrow=c(1,1))
#Usually, we want to subtract the mean from the time series.

#the spectrum function goes further and automatically removes a linear trend from the series before
#calculating the periodogram

#Smoothing the periodogram

#Spectral analysis in R

#The default frequency axis is in cycles per sampling interval

#It is
#more intuitive to convert the frequency axis to cycles per unit time, we can do this by extracting the
#frequency values that R returns and dividing by the length of the sampling interval. We should also
#multiply the spectral density by 2 so that the area under the periodogram actually equals the variance
#of the time series
spectrum(x,log="no",span=10)
```




```
del<-0.1 # sampling interval

x.spec <- spectrum(x,log="no",span=10,plot=FALSE)

spx <- x.spec$freq/del
spy <- 2*x.spec$spec
plot(spy~spx,xlab="frequency",ylab="spectral density",type="l")
```



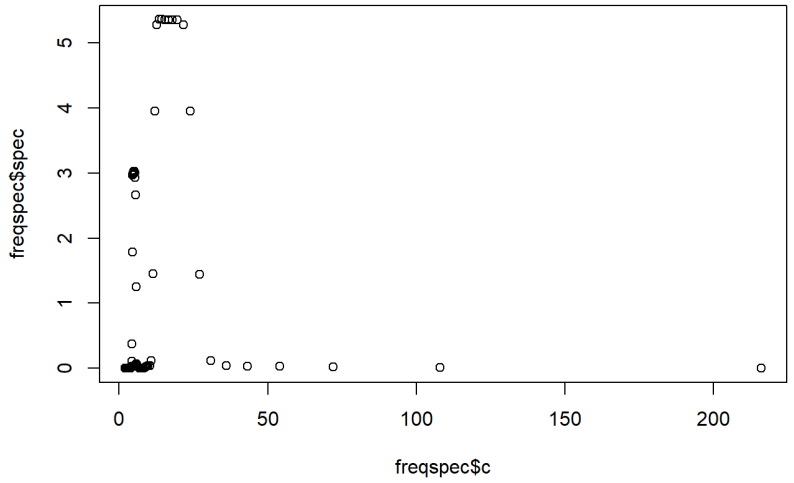

```
spx %>% plot
```



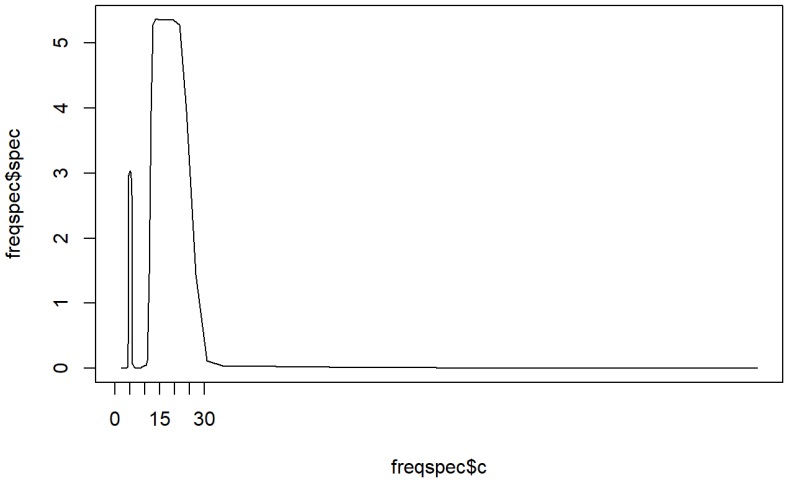

```
attributes(x.spec)
```

```
## $names
##  [1] "freq"      "spec"      "coh"       "phase"     "kernel"    "df"
##  [7] "bandwidth" "n.used"    "orig.n"    "series"    "snames"    "method"
## [13] "taper"     "pad"       "detrend"   "demean"
##
## $class
## [1] "spec"
```

```
freqspec=data.frame(freq=x.spec$freq,spec=x.spec$spec)
freqspec=freqspec%>% mutate(c=1/freq)

plot(freqspec$spec~freqspec$c)
```




```
plot(freqspec$spec~freqspec$c,type="l",xaxt="n")
axis(1, at = seq(0,30, by = 5))
```




```
## so it seems that we have a cycle every 5 periods, and 1 more every 10-30 periods

freqspec %>% filter(c>=10 & c<=30) %>% summarize(mean=mean(c))
```

```
##      mean
## 1 16.2386
```

```
#so one more at 16 which is the dominant one
forecast::findfrequency(x)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
## [1] 16
```

```
quantile(freqspec$c)
```

```
##         0%        25%        50%        75%       100%
##   2.000000   2.658537   3.963636   7.785714 216.000000
```

```
q=quantile(freqspec$c,probs=c(0.1,0.9))

freqspec %>% filter(between(c,q[1],q[2])) %>% head #filter for 10% 90% values -->trimming
```

```
##         freq     spec        c
## 1 0.05555556 5.359152 18.00000
## 2 0.06018519 5.356081 16.61538
## 3 0.06481481 5.356592 15.42857
## 4 0.06944444 5.360568 14.40000
## 5 0.07407407 5.360787 13.50000
## 6 0.07870370 5.283165 12.70588
```

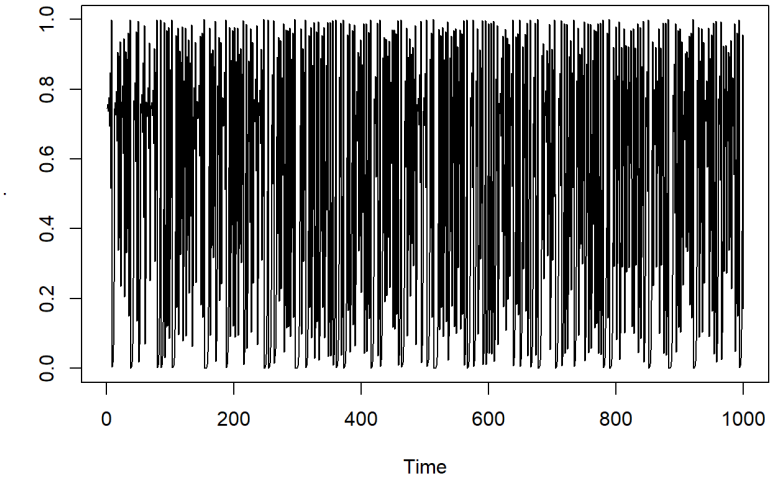
```
freqspec %>% filter(between(c,q[1],q[2])) %>% tail
```

```
##         freq         spec        c
## 81 0.4259259 4.408458e-08 2.347826
## 82 0.4305556 3.067506e-08 2.322581
## 83 0.4351852 2.296704e-08 2.297872
## 84 0.4398148 2.164951e-08 2.273684
## 85 0.4444444 2.175318e-08 2.250000
## 86 0.4490741 2.207253e-08 2.226804
```

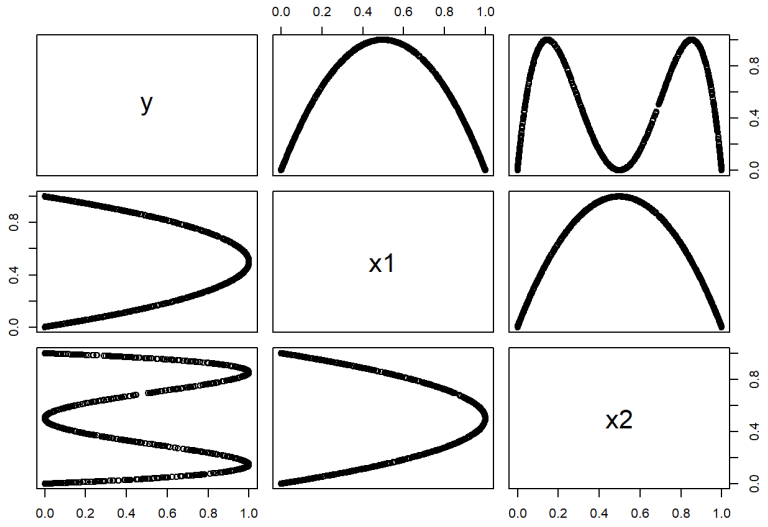
```
# DChaos ------------------------------------------------------------------



pacman::p_load(DChaos)

set.seed(34)
#Simulates time-series data from the Logistic map with chaos
ts <- DChaos::logistic.sim(n=1000, a=4)
show(head(ts, 5))
```

```
## Time Series:
## Start = 1
## End = 5
## Frequency = 1
## [1] 0.7466701 0.7566155 0.7365940 0.7760930 0.6950905
```

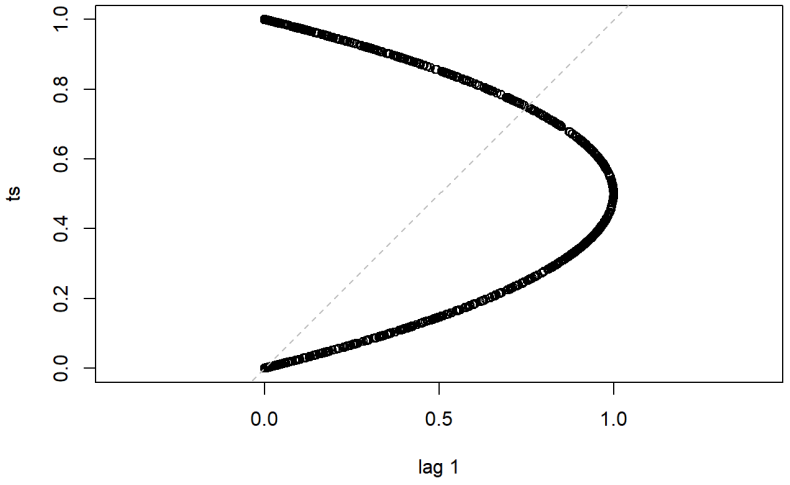
```
ts %>% plot(type="l")
```



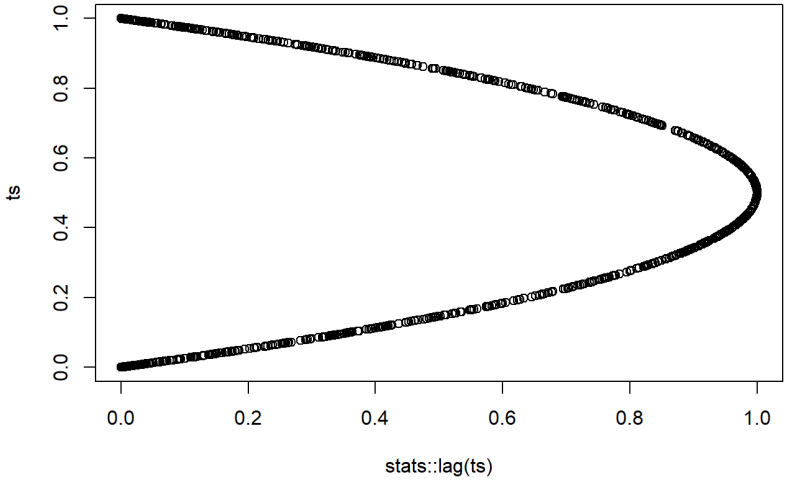

```
#Provides the uniform delayed-coordinate embedding vectors (Backward)
 data <- DChaos::embedding(ts, m=3, lag=1, timelapse="FIXED")
data %>% plot
```



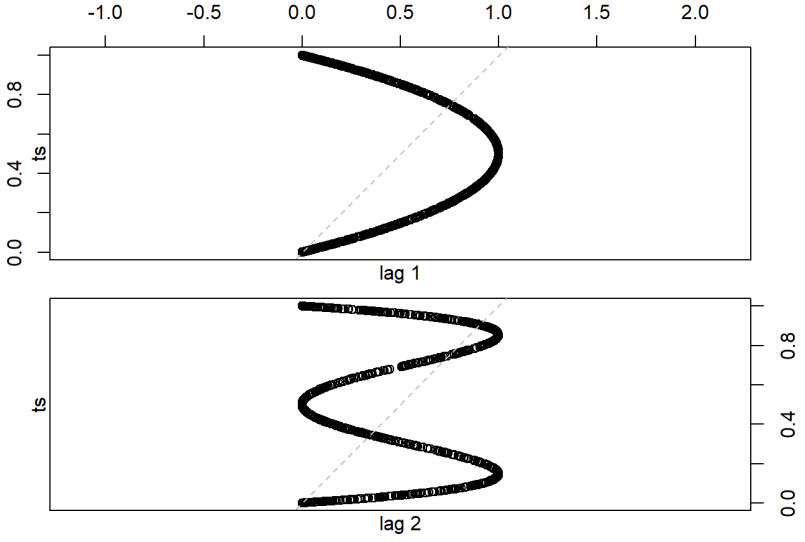

```
#same as
lag.plot(ts)
```



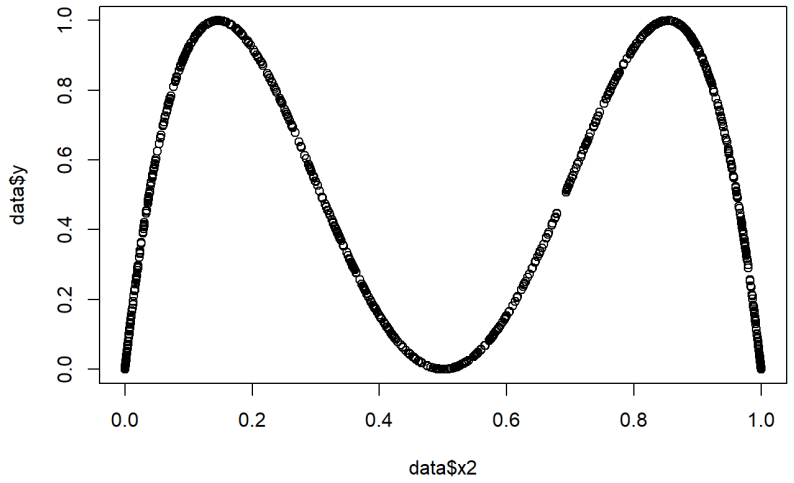

```
plot(ts~stats::lag(ts))
```



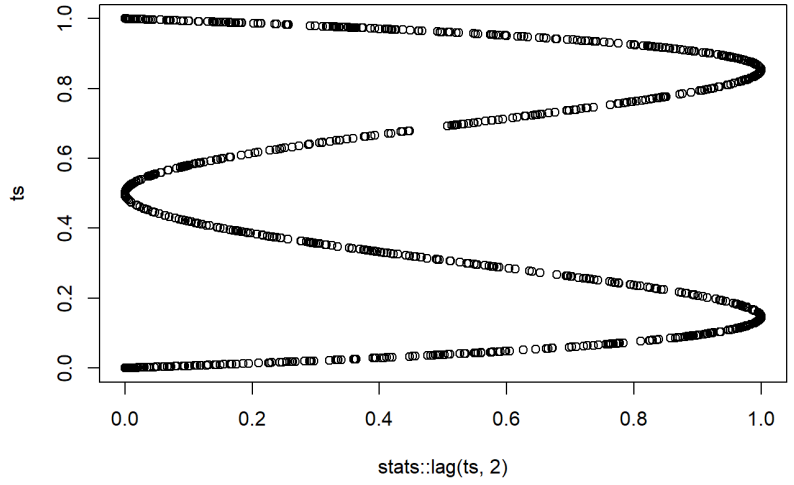

```
lag.plot(ts,2)
```



```
plot(data$x2,data$y)
```




```
#same as
plot(ts~stats::lag(ts,2))
```




```
stats::lag(ts)==data$x2
```

```
## Warning in `==.default`(stats::lag(ts), data$x2): longer object length is not a
## multiple of shorter object length
```

```
## Time Series:
## Start = 0
## End = 999
## Frequency = 1
##    [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##   [13]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##   [25]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##   [37]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##   [49]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##   [61]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##   [73]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##   [85]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##   [97]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [109]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [121]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [133]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [145]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [157]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [169]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [181]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [193]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [205]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [217]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [229]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [241]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [253]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [265]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [277]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [289]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [301]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [313]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [325]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [337]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [349]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [361]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [373]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [385]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [397]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [409]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [421]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [433]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [445]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [457]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [469]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [481]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [493]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [505]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [517]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [529]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [541]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [553]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [565]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [577]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [589]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [601]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [613]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [625]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [637]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [649]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [661]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [673]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [685]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [697]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [709]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [721]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [733]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [745]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [757]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [769]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [781]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [793]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [805]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [817]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [829]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [841]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [853]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [865]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [877]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [889]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [901]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [913]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [925]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [937]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [949]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

```
##  [961]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [973]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [985]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [997]  TRUE  TRUE FALSE FALSE
```

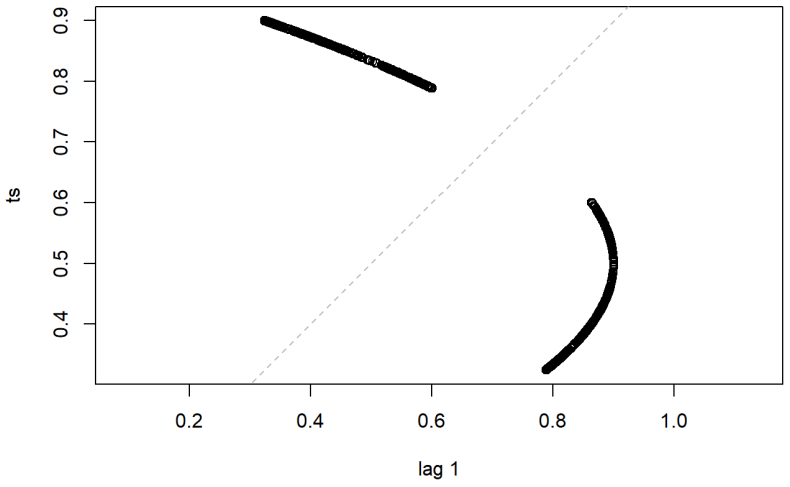
```
?plot3d
```

```
## No documentation for 'plot3d' in specified packages and libraries:
## you could try '??plot3d'
```

```
library(rgl)
```

```
## Warning: package 'rgl' was built under R version 4.1.2
```

```
#https://en.wikipedia.org/wiki/Logistic_map phase spase for logistic a=4
plot3d(
  x=data$x1, y=data$x2, z=data$y,
  type = 's',
  col="red")
#estimates lyapunov exponent
#https://math.libretexts.org/Bookshelves/Scientific_Computing_Simulations_and_Modeling/Book%3A_Introduction_to_the_Modeling_and_Analysis_of_Complex_Systems_(Sayama)/09%3A_Chaos/9.03%3A_Lyapunov_Exponent

#https://hypertextbook.com/chaos/lyapunov-1/

#If Lyapunov's exponent is positive, then the behavior of the dynamical system is chaotic.
#if negative the orbit attracts to a stable point or stable periodoc orbit
#if 0 the orbit is a neutral fixed point

#lyapunov(ts) too many computations


#jacobian <- DChaos::jacobian.net(data=ts, m=3:3, lag=1:1, timelapse="FIXED", h=2:10)

#lyapunov.max(jacobian)



ts <- DChaos::logistic.sim(n=1000, a=3.6)
data <- DChaos::embedding(ts, m=3, lag=1, timelapse="FIXED")

lag.plot(ts,type="p")
```




```
?lag.plot
```

```
## starting httpd help server ...
```

```
##  done
```

```
plot3d(
  x=data$x1, y=data$x2, z=data$y,
  type = 's',
  col="red")




# ForeCA ----------------------------------------------------------------

#https://cran.r-project.org/web/packages/ForeCA/vignettes/Introduction.html

pacman::p_load(ForeCA)

# spectrum control
sc <- list(method = "mvspec")
# entropy control
ec <- list(prior.weight = 1e-2)
data("EuStockMarkets")

# log-returns in %
ret <- diff(log(EuStockMarkets)) * 100
cor.ret <- cor(ret)
knitr::kable(format(cor.ret, digits = 2),
      caption = "Correlation matrix")
```

Correlation matrix

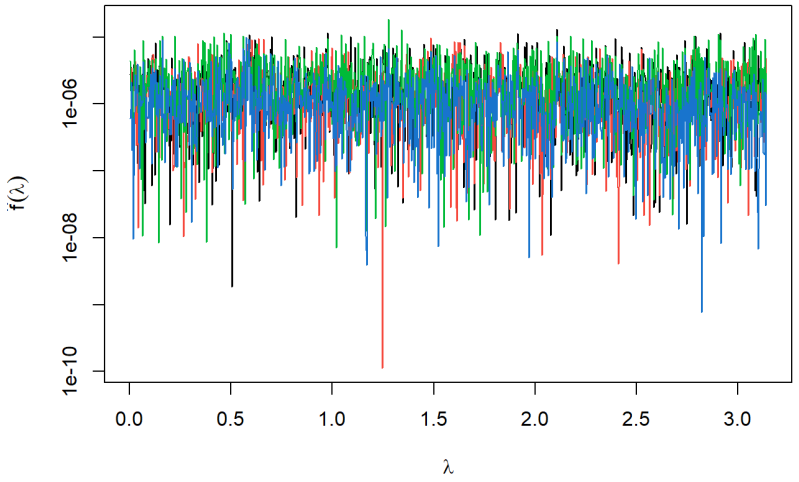
	DAX	SMI	CAC	FTSE
DAX	1.00	0.70	0.73	0.64
SMI	0.70	1.00	0.62	0.58
CAC	0.73	0.62	1.00	0.65
FTSE	0.64	0.58	0.65	1.00

```
#the meaning of the inverted correlation matrix
#http://www2.tulane.edu/~PsycStat/dunlap/Psyc613/RI2.html

knitr::kable(format(solve(cor.ret), digits = 2),
             caption = "Conditional covariance given other variables")
```

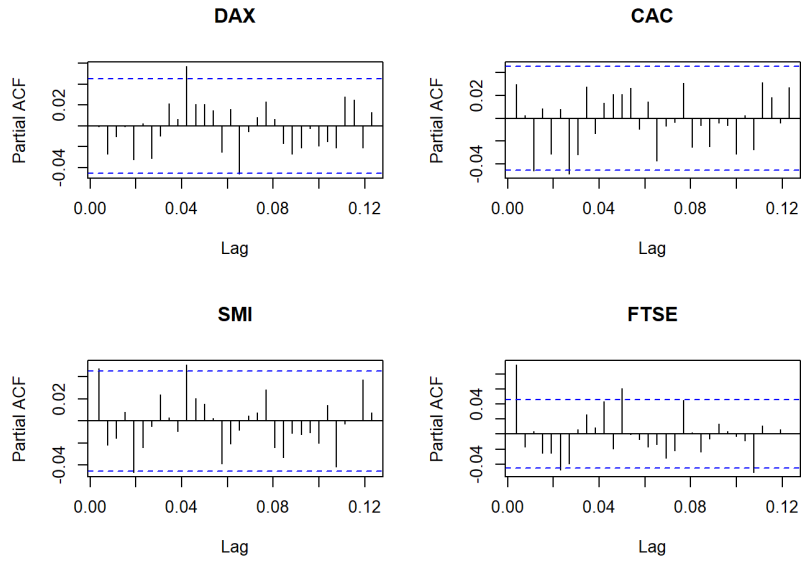
Conditional covariance given other variables

	DAX	SMI	CAC	FTSE
DAX	2.90	-1.03	-1.18	-0.49
SMI	-1.03	2.14	-0.31	-0.40
CAC	-1.18	-0.31	2.51	-0.69
FTSE	-0.49	-0.40	-0.69	1.99

```
ret.spec <- mvspectrum(ret, method = sc$method)
plot(ret.spec)
```




```
?mvspectrum


layout(matrix(seq_len(ncol(ret)), ncol = 2))
for (nn in colnames(ret)) {
  pacf(ret[, nn], main = nn)
}
```



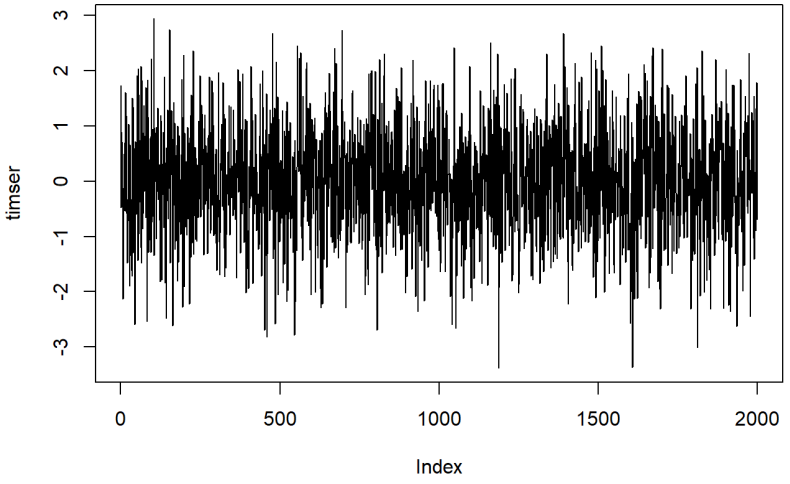

```
#More specifically, we can estimate ForeCA measure of forecastability, \(\Omega\), for each series:
ret.omega <- Omega(ret, spectrum.control = sc, entropy.control = ec)
ret.omega
```

```
##      DAX      SMI      CAC     FTSE
## 5.353323 5.135365 4.966076 5.253096
## attr(,"unit")
## [1] "%"
```

```
#According to the estimates CAC is the least forecastable, DAX is the most forecastable stock market.

#However, we can ask if there are linear combinations of stock markets, i.e.,
#a portfolio, that are even easier to forecast. That's exactly what ForeCA is doing.

mod.foreca.ret <- foreca(ret, n.comp = ncol(ret),
                         spectrum.control = sc,
                         entropy.control = ec)
#mod.foreca.ret  # this uses the print.foreca method




# Entropy -------------------------------------------------------------

pacman::p_load(TSEntropies)
#https://en.wikipedia.org/wiki/Approximate_entropy

# A time series containing many repetitive patterns has a relatively small ApEn;
#a less predictable process has a higher ApEn., where ApEn is approximiate entropy
par(mfrow=c(1,1))
timser <- rnorm(2000)
ApEn(timser)
```

```
## [1] 1.92855
```

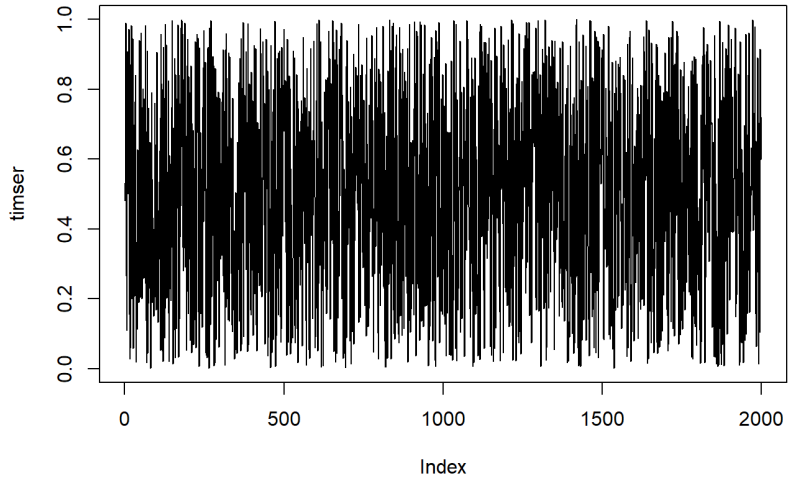
```
plot(timser,type="l")
```




```
timser=runif(2000)

ApEn(timser)
```

```
## [1] 1.998601
```

```
plot(timser,type="l")
```



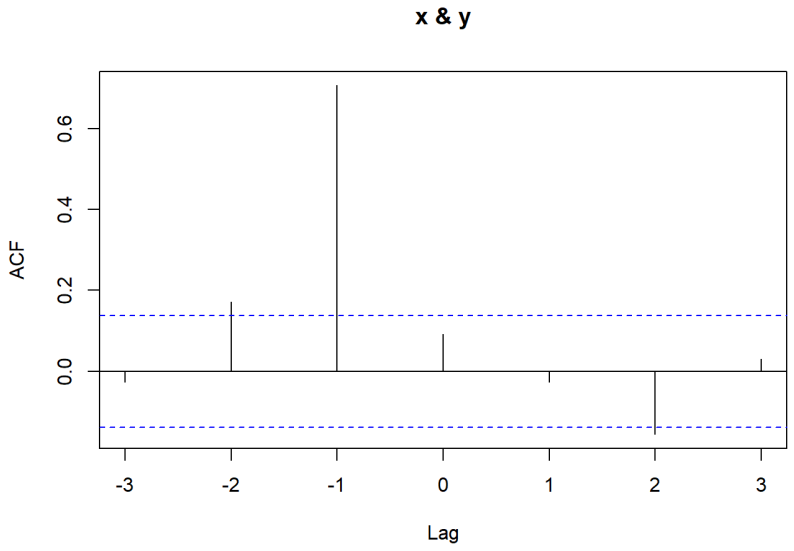

```
timser <- rnorm(2000)
ApEn(timser)
```

```
## [1] 1.895816
```

```
ApEn(ts) # the chaotic one
```

```
## [1] 0.2031768
```

```
#### Transfrer entropy
pacman::p_load(RTransferEntropy)
#https://en.wikipedia.org/wiki/Transfer_entropy


#simulated series

set.seed(12345)
n <- 200
x <- rep(0, n + 1)
y <- rep(0, n + 1)

for (i in 2:(n + 1)) {
  x[i] <- 0.2 * x[i - 1] + rnorm(1, 0, 2)
  y[i] <- x[i - 1] + rnorm(1, 0, 2)
}

x <- x[-1]
y <- y[-1]


ccf(x=x,y=y,lag=3)
print(ccf(y=y, x=x,lag=3))
```




```
##
## Autocorrelations of series 'X', by lag
##
##     -3     -2     -1      0      1      2      3
## -0.026  0.171  0.706  0.092 -0.026 -0.155  0.031
```

```
ApEn(x)
```

```
## [1] 0.8690917
```

```
ApEn(y)
```

```
## [1] 0.8657891
```

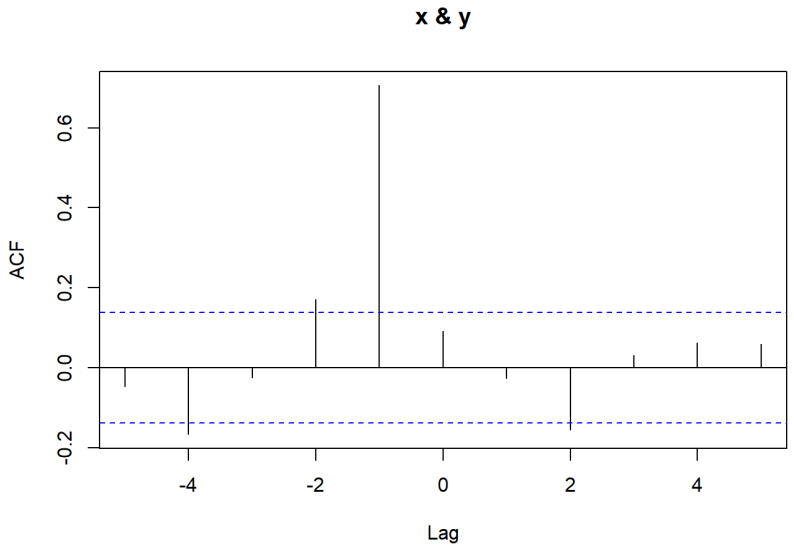
```
set.seed(12345)
shannon_te <- transfer_entropy(x, y)
```

```
## Shannon's entropy on 1 core with 100 shuffles.
##   x and y have length 200 (0 NAs removed)
##   [calculate] X->Y transfer entropy
##   [calculate] Y->X transfer entropy
##   [bootstrap] 300 times
## Done - Total time 7.37 seconds
```

```
shannon_te
```

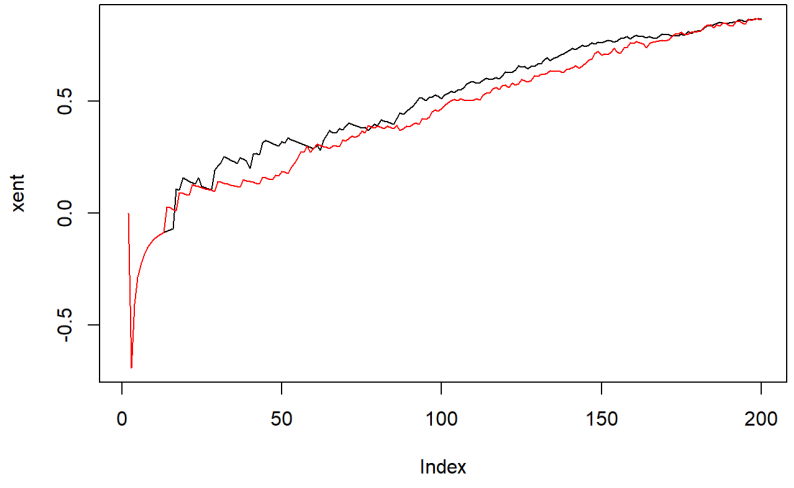
```
## Shannon Transfer Entropy Results:
## ------------------------------------------------------------
## Direction        TE   Eff. TE  Std.Err.   p-value    sig
## ------------------------------------------------------------
##      X->Y    0.1030    0.0865    0.0115    0.0000    ***
##      Y->X    0.0141    0.0000    0.0082    0.5233
## ------------------------------------------------------------
## Bootstrapped TE Quantiles (300 replications):
## ------------------------------------------------------------
## Direction      0%     25%     50%     75%    100%
## ------------------------------------------------------------
##      X->Y  0.0018  0.0114  0.0144  0.0200  0.0614
##      Y->X  0.0040  0.0125  0.0162  0.0258  0.0787
## ------------------------------------------------------------
## Number of Observations: 200
## ------------------------------------------------------------
## p-values: < 0.001 '***', < 0.01 '**', < 0.05 '*', < 0.1 '.'
```

```
## from the above we can see that there is a significant flow of information from x to y
## but not vice versa


#the cross correlation agrees that x causes y and not vice versa (that much)
ccf(x=x,y=y,lag=3)
print(ccf(y=y, x=x,lag=5))
```




```
##
## Autocorrelations of series 'X', by lag
##
##     -5     -4     -3     -2     -1      0      1      2      3      4      5
## -0.047 -0.167 -0.026  0.171  0.706  0.092 -0.026 -0.155  0.031  0.062  0.059
```

```
yent=vector()
xent=vector()
for (i in 2:200) {
  xent[i]=ApEn(x[1:i])
  yent[i]=ApEn(y[1:i])
}

plot(xent,type="l")
lines(yent,col="red")
```




```
cor(yent,xent,use="complete.obs")
```

```
## [1] 0.9886563
```

```
r1=rnorm(100)
r2=rnorm(100)

transfer_entropy(r1, r2)
```

```
## Shannon's entropy on 1 core with 100 shuffles.
##   x and y have length 100 (0 NAs removed)
##   [calculate] X->Y transfer entropy
##   [calculate] Y->X transfer entropy
##   [bootstrap] 300 times
## Done - Total time 5.38 seconds
```

```
## Shannon Transfer Entropy Results:
## ------------------------------------------------------------
## Direction        TE   Eff. TE  Std.Err.   p-value    sig
## ------------------------------------------------------------
##      X->Y    0.0146    0.0000    0.0191    0.7467
##      Y->X    0.0400    0.0055    0.0164    0.1033
## ------------------------------------------------------------
## Bootstrapped TE Quantiles (300 replications):
## ------------------------------------------------------------
## Direction      0%     25%     50%     75%    100%
## ------------------------------------------------------------
##      X->Y  0.0059  0.0146  0.0204  0.0273  0.1353
##      Y->X  0.0032  0.0144  0.0237  0.0394  0.1063
## ------------------------------------------------------------
## Number of Observations: 100
## ------------------------------------------------------------
## p-values: < 0.001 '***', < 0.01 '**', < 0.05 '*', < 0.1 '.'
```