Time Series

Emerson Amirhosein Azarbakht

Time Series (TS)

Used in

- control of inventory, based on demand trends
- airline's decision to buy airplanes bc of passenger trends and decision to increase/maintain market share
- climate change decisions based on temperature change trends
- business/sales forecasting
- everyday operational decisions
- long-term effects of proposed water management policies by simulating daily rainfall and sea state time series
- understanding fluctuations in monthly sales
- basis for signal processing in telecommunications <?>
- disease incidence tracking, yearly rates
- census analysis
- tracking monthly unemployment rate; as an economic indicator used by decision makers

Used to

- to understand the past, and predict the future
- forcasting (predicting inference, a subset of statistical inference). assumes that present trends continue. This assumption cannot be checked empirically, but, when we identify the likely causes for a trend, we can justify the forecasting(extrapolating it) for a few time-steps at least
- anomaly detection
- clustering
- classification (assigning a time series pattern to a specific category: e.g. gesture recognition of hand movements in sign language videos)
- query by content ~ Content-based image retrieval

Data: a variable measured sequentially in time, or at a fixed [sampling] interval

serial dependence problem: observations close together in time tend to be correlated (serially dependent)

TS tries to explain this correlation (serial dependence) autocorrelation analysis examines this serial dependence <?>

conditions (assumptions of TS)

- Stationary process:
- if there's no systematic change in mean (no trend) AND
- if there's no systematic change in variance, AND
- if strictly periodic variations have been removed
    - i.e. the properties of one section of the data are much like those of any other section

    - often we have a non-stationary TS => we need to remove trend and seasonality, to get a stationary residual, which then can be modeled using a stationary stochastic process

- Ergodic process: a stationary TS that we assume is sufficiently long time series that it characterises the hypothetical model. (~ independent of the starting point)

```r
plot(AirPassengers)
start(AirPassengers)
```

```
## [1] 1949    1
```

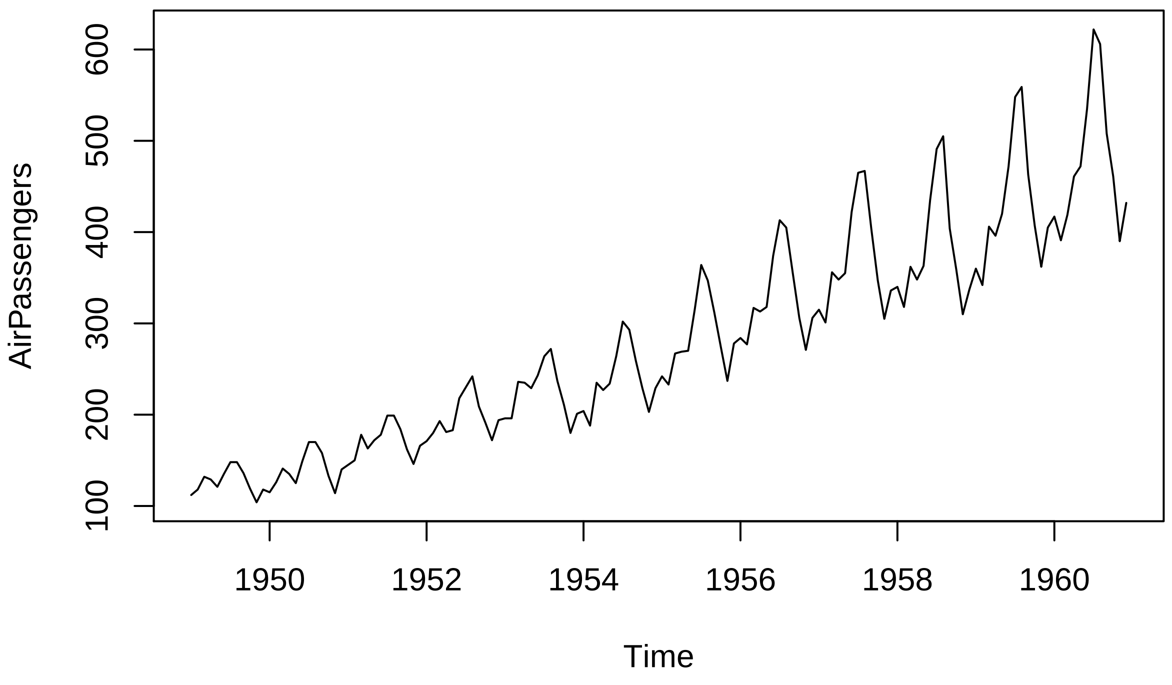
```r
end(AirPassengers)
```

```
## [1] 1960   12
```

```r
frequency(AirPassengers)
```

```
## [1] 12
```

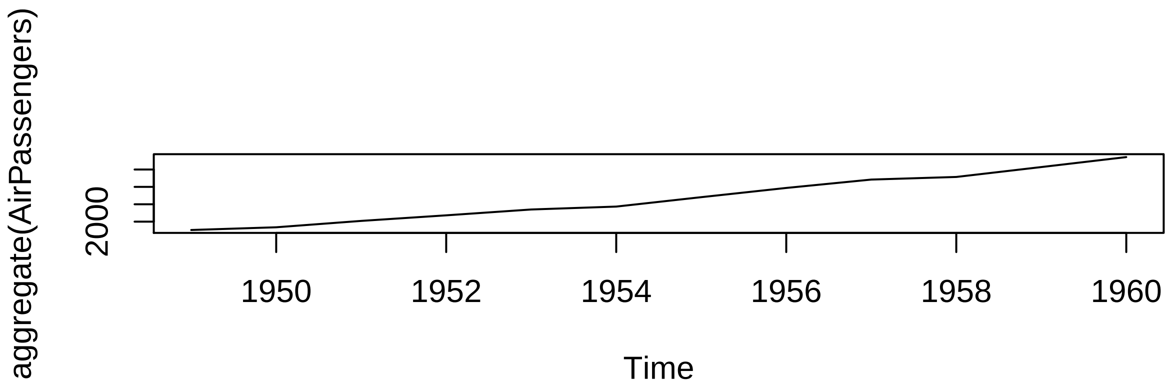
```r
plot(AirPassengers)
```

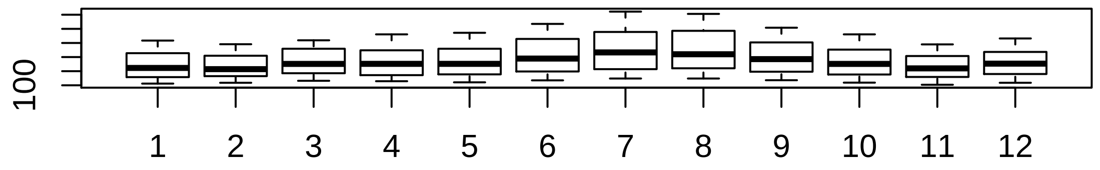


```r
summary(AirPassengers)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   104.0   180.0   265.5   280.3   360.5   622.0
```

```r
layout(1:2)
# takes an input matrix for the location of each plot in the graphics window
plot(aggregate(AirPassengers))
boxplot(AirPassengers ~ cycle(AirPassengers))
```

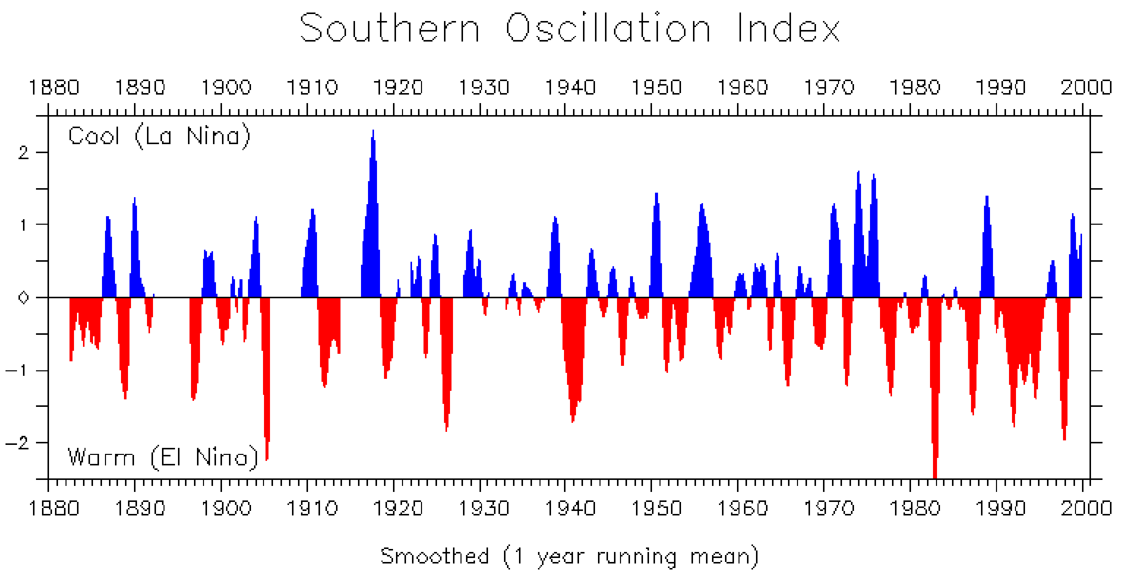




plotting shows *patterns*, and *features* of the data + *outliers* and *erroneous* values

patterns

1. trend = a non-periodic systematic change in a TS: a long-term change in the 'mean'
   - can be modeled simply by a linear increase or decrease. (only if it's deterministic ~ it's non-stochastic)
   - stochastic trend: seems to change direction at unpredictable times rather than displaying a consistent pattern (e.g. like the air passenger series)
2. seasonal variation = a repeating pattern within a fixed period (e.g. each year)
3. cycles = a non-fixed-period cycle (without a fixed period). example: El-Nino



```r
# monthly unemployment rate for the US state of Maine from January 1996 until August 2006
Maine.month <- read.table("http://staff.elena.aut.ac.nz/Paul-Cowpertwait/ts/Maine.dat", header = TRUE)
# header TRUE means treat first row as column names
attach(Maine.month)
str(Maine.month)
```

Monthly unemployment rate for a state

```
## 'data.frame':    128 obs. of  1 variable:
##  $ unemploy: num  6.7 6.7 6.4 5.9 5.2 4.8 4.8 4 4.2 4.4 ...
```

```r
head(Maine.month)
```

```
##   unemploy
## 1      6.7
## 2      6.7
## 3      6.4
## 4      5.9
## 5      5.2
## 6      4.8
```

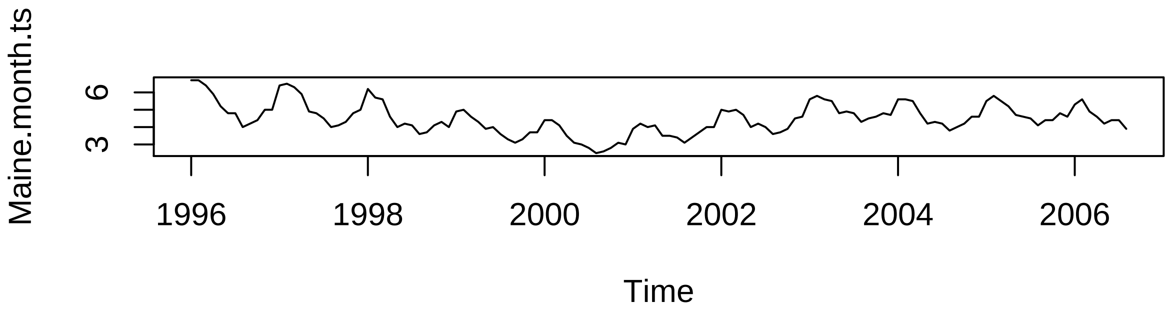
```r
class(Maine.month)
```

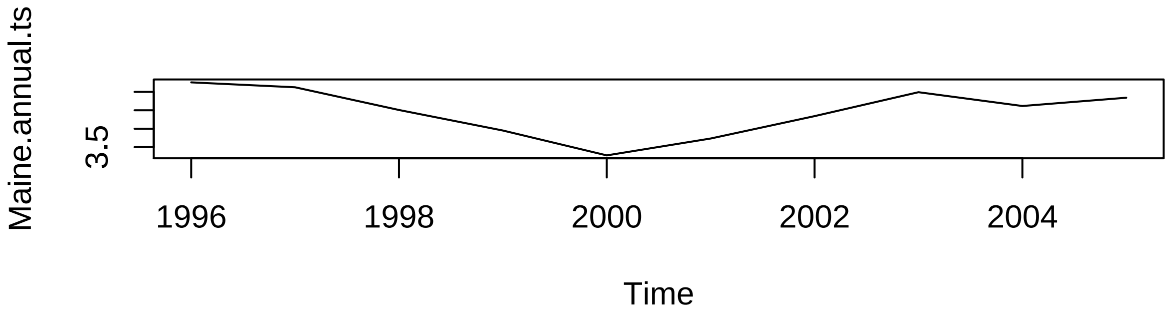
```
## [1] "data.frame"
```

```r
# it's a data.frame, not a ts object. So, we need to convert it to ts
Maine.month.ts <- ts(unemploy, start = c(1996, 1), freq = 12)
Maine.month.ts
```

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 1996 6.7 6.7 6.4 5.9 5.2 4.8 4.8 4.0 4.2 4.4 5.0 5.0
## 1997 6.4 6.5 6.3 5.9 4.9 4.8 4.5 4.0 4.1 4.3 4.8 5.0
## 1998 6.2 5.7 5.6 4.6 4.0 4.2 4.1 3.6 3.7 4.1 4.3 4.0
## 1999 4.9 5.0 4.6 4.3 3.9 4.0 3.6 3.3 3.1 3.3 3.7 3.7
## 2000 4.4 4.4 4.1 3.5 3.1 3.0 2.8 2.5 2.6 2.8 3.1 3.0
## 2001 3.9 4.2 4.0 4.1 3.5 3.5 3.4 3.1 3.4 3.7 4.0 4.0
## 2002 5.0 4.9 5.0 4.7 4.0 4.2 4.0 3.6 3.7 3.9 4.5 4.6
## 2003 5.6 5.8 5.6 5.5 4.8 4.9 4.8 4.3 4.5 4.6 4.8 4.7
## 2004 5.6 5.6 5.5 4.8 4.2 4.3 4.2 3.8 4.0 4.2 4.6 4.6
## 2005 5.5 5.8 5.5 5.2 4.7 4.6 4.5 4.1 4.4 4.4 4.8 4.6
## 2006 5.3 5.6 4.9 4.6 4.2 4.4 4.4 3.9
```

```
Maine.annual.ts <- aggregate(Maine.month.ts)/12
layout(1:2)
plot(Maine.month.ts)
plot(Maine.annual.ts)
```





```
Maine.Feb <- window(Maine.month.ts, start = c(1996,2), freq = TRUE)
Maine.Aug <- window(Maine.month.ts, start = c(1996,8), freq = TRUE)
Feb.ratio <- mean(Maine.Feb) / mean(Maine.month.ts)
Aug.ratio <- mean(Maine.Aug) / mean(Maine.month.ts)
```

Multiple TS

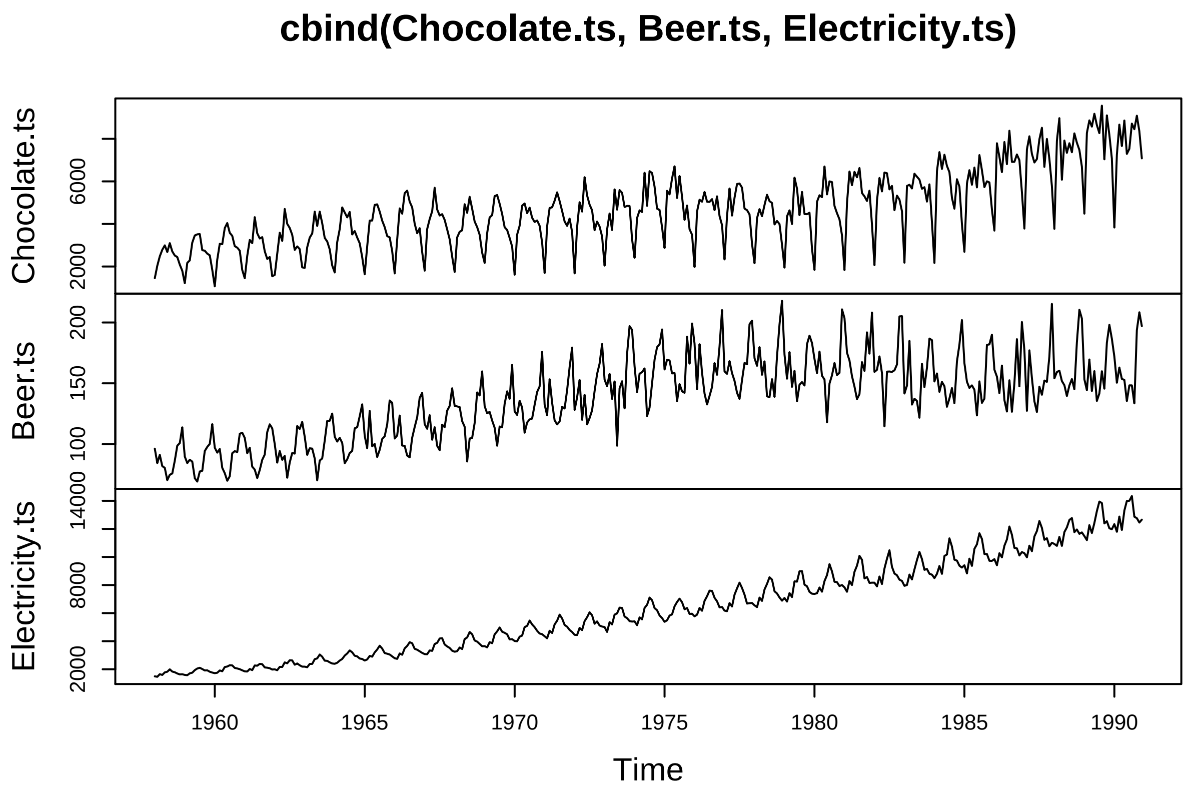
```
ChocolateBeerElectricity <- read.table("http://staff.elena.aut.ac.nz/Paul-Cowpertwait/ts/cbe.dat", head
class(ChocolateBeerElectricity)
```

```
## [1] "data.frame"
```

```
str(ChocolateBeerElectricity)
```

```
## 'data.frame':    396 obs. of  3 variables:
##  $ choc: int  1451 2037 2477 2785 2994 2681 3098 2708 2517 2445 ...
##  $ beer: num  96.3 84.4 91.2 81.9 80.5 70.4 74.8 75.9 86.3 98.7 ...
##  $ elec: int  1497 1463 1648 1595 1777 1824 1994 1835 1787 1699 ...
```

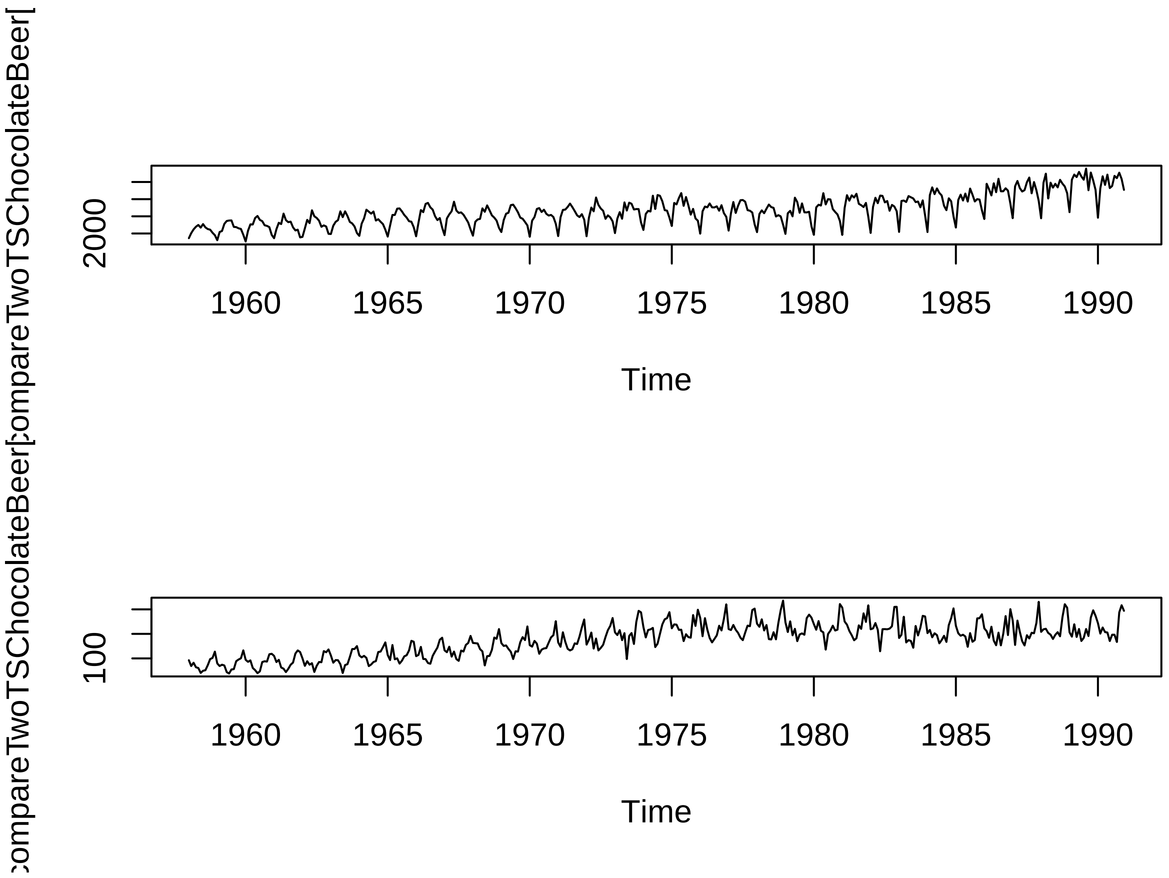
```
Chocolate.ts <- ts(ChocolateBeerElectricity[,1], start = 1958, frequency = 12)
Beer.ts <- ts(ChocolateBeerElectricity[,2], start = 1958, frequency = 12)
Electricity.ts <- ts(ChocolateBeerElectricity[,3], start = 1958, frequency = 12)

plot(cbind(Chocolate.ts, Beer.ts, Electricity.ts))
```



```
# intersection of multiple TS
compareTwoTSChocolateBeer <- ts.intersect(Chocolate.ts, Beer.ts)
# bind time series which have a common frequency
?ts.intersect()
head(compareTwoTSChocolateBeer)
```

```
##      Chocolate.ts Beer.ts
## [1,]         1451    96.3
## [2,]         2037    84.4
## [3,]         2477    91.2
## [4,]         2785    81.9
## [5,]         2994    80.5
## [6,]         2681    70.4
```

```
layout(1:2)
plot(compareTwoTSChocolateBeer[,1])
plot(compareTwoTSChocolateBeer[,2])
```



```
cor(Chocolate.ts, Beer.ts)
```

```
## [1] 0.3774314
```

```
# correlation
```

```
# Global temperature anomalies from the monthly means over the period
GlobalTemperatures <- scan("http://staff.elena.aut.ac.nz/Paul-Cowpertwait/ts/global.dat")
```

```
# scan to read data into a vector or list from the console or file.
?scan
str(GlobalTemperatures)
```

Climate change

```
##  num [1:1800] -0.384 -0.457 -0.673 -0.344 -0.311 -0.071 -0.246 -0.235 -0.38 -0.418 ...
```

```
GlobalTemperatures.ts <- ts(GlobalTemperatures, st = c(1856,1), end = c(2005,1), frequency = 12)
head(GlobalTemperatures.ts)
```

```
## [1] -0.384 -0.457 -0.673 -0.344 -0.311 -0.071
```

```
tail(GlobalTemperatures.ts)
```

```
## [1] 0.436 0.452 0.494 0.586 0.385 0.502
```

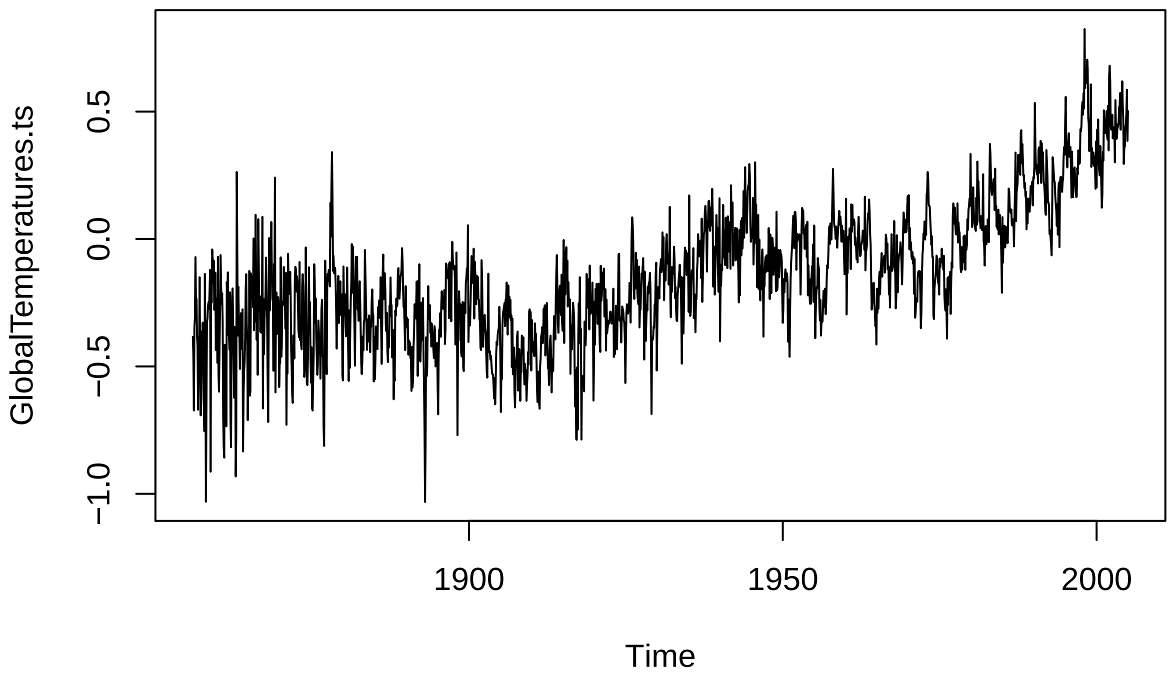
```
Global.annual <- aggregate(GlobalTemperatures.ts, FUN = mean)
head(Global.annual)
```

```
## [1] -0.3812500 -0.4611667 -0.4153333 -0.2252500 -0.3697500 -0.4003333
```

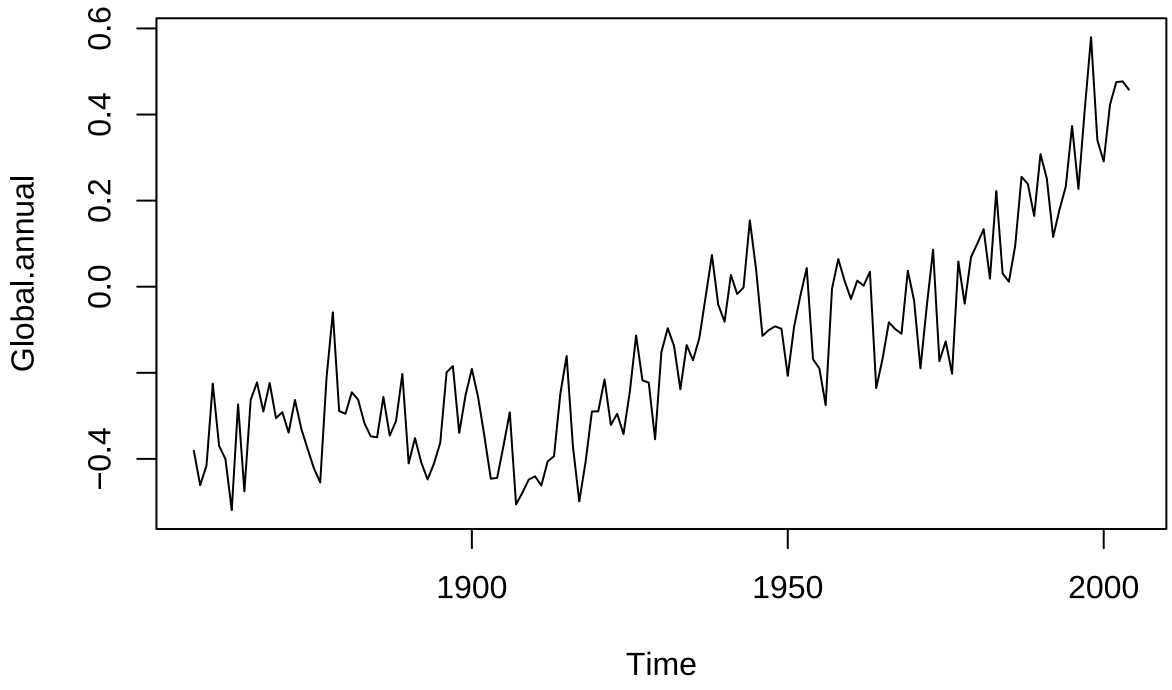
```
Global.annual
```

```
## Time Series:
## Start = 1856
## End = 2004
## Frequency = 1
##   [1] -0.381250000 -0.461166667 -0.415333333 -0.225250000 -0.369750000
##   [6] -0.400333333 -0.518916667 -0.273333333 -0.475333333 -0.262583333
##  [11] -0.222416667 -0.289916667 -0.223583333 -0.305500000 -0.291333333
##  [16] -0.339000000 -0.263083333 -0.329583333 -0.376500000 -0.421583333
##  [21] -0.454750000 -0.212166667 -0.059500000 -0.288916667 -0.295166667
##  [26] -0.245333333 -0.262416667 -0.317166667 -0.347583333 -0.349916667
##  [31] -0.256083333 -0.345833333 -0.311916667 -0.202750000 -0.410416667
##  [36] -0.351750000 -0.408583333 -0.447833333 -0.411083333 -0.362666667
##  [41] -0.199083333 -0.184250000 -0.339416667 -0.252416667 -0.190916667
##  [46] -0.257583333 -0.348583333 -0.446000000 -0.444166667 -0.370000000
##  [51] -0.291916667 -0.505750000 -0.478750000 -0.448333333 -0.440500000
##  [56] -0.461833333 -0.405916667 -0.393916667 -0.249333333 -0.161000000
##  [61] -0.371416667 -0.498666667 -0.407333333 -0.289833333 -0.289750000
##  [66] -0.215500000 -0.321333333 -0.295083333 -0.342250000 -0.244333333
##  [71] -0.113416667 -0.217416667 -0.222916667 -0.354416667 -0.151833333
##  [76] -0.096500000 -0.137500000 -0.238333333 -0.135666667 -0.170750000
##  [81] -0.119416667 -0.023666667  0.073583333 -0.041916667 -0.081166667
##  [86]  0.027250000 -0.017166667 -0.001833333  0.154083333  0.038583333
##  [91] -0.114250000 -0.100666667 -0.092083333 -0.097666667 -0.207083333
##  [96] -0.093000000 -0.020416667  0.043250000 -0.168416667 -0.189583333
## [101] -0.275166667 -0.004750000  0.064166667  0.013250000 -0.028416667
## [106]  0.014166667  0.002166667  0.034500000 -0.235583333 -0.167583333
## [111] -0.082750000 -0.098416667 -0.109333333  0.037000000 -0.032500000
## [116] -0.189500000 -0.045833333  0.085833333 -0.173083333 -0.127000000
## [121] -0.202250000  0.058416667 -0.039416667  0.068416667  0.100333333
## [126]  0.133583333  0.018666667  0.222416667  0.030666667  0.011666667
## [131]  0.095500000  0.255166667  0.238666667  0.164416667  0.307916667
## [136]  0.251250000  0.115666667  0.178666667  0.232583333  0.373166667
## [141]  0.227000000  0.411000000  0.579666667  0.340500000  0.291083333
## [146]  0.422916667  0.475416667  0.476916667  0.457750000
```

```r
plot(GlobalTemperatures.ts)
```



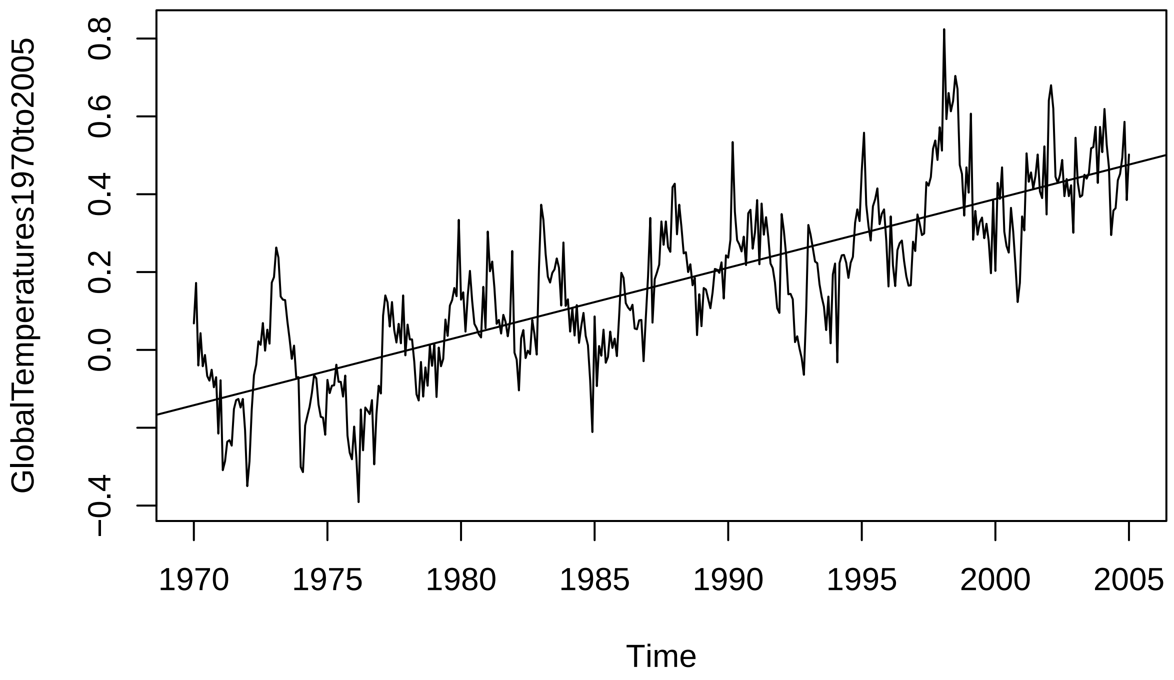
```r
plot(Global.annual)
```



```
#
GlobalTemperatures1970to2005 <- window(GlobalTemperatures.ts, start=c(1970, 1), end=c(2005, 12))
```

```
## Warning in window.default(x, ...): 'end' value not changed
```

```
# subset a time window
?window
GlobalTemperatures1970to2005.time <- time(GlobalTemperatures1970to2005)
# create the vector of times at which a time series was sampled
?time
plot(GlobalTemperatures1970to2005)
abline(reg = lm(GlobalTemperatures1970to2005 ~ GlobalTemperatures1970to2005.time))
```



—

In statistics, usually, first thing, we compute the ‘mean’ and ‘variance/standard deviation’, which show ‘location’ and ‘dispersion’:

- mean ~ location
- variance ~ dispersion

or

- mean ~ central location
- variance ~ the spread

Transformations

Transform the data (e.g. log or square-root) when:

1. if variance appears to increase with the mean.
   - if standard deviation is directly proportional to the mean (a trend), LOG transform!
   - if variance changes through time, but without a trend, transformation won’t help. use a model that accommodates variance change
2. if there’s additive seasonal effect, i.e. size of seasonal effect increases with the mean, transform! to make seasonal effect constant.
   - If there’s multiplicative seasonal effect, i.e. size of seasonal effect proportional to the mean, LOG transform! to make the seasonal effect additive. (variance gets stable, but error term will still remain unstable)
3. if there are spikes in the data, (skewness), transform to normalize the data distribution.

   avoid transformations, except where the transformed has a direct physical interpretation

Sample autocorrelation coefficients

- a series of quantities, that measure the correlation between observations at different distances apart
- show us which probability model to use for that data
- negative correlation? shows high values of x tend to go with low values of y
- zero correlation? shows two variables are independent

serial correlation coefficients (e.g. at lag 1), or autocorrelation coefficients

- measures correlation between successive observations
- serial correlation coefficients at lag k
- if you graph it, it's called 'correlogram'
- ac.f and correlogram is meaningful ONLY IF data is STATIONARY
- ac.f and correlogram is meaningless for non-stationary

how to interpret a Correlogram

- if r_0 = 1, r_1 = large value, r_2, r_3 = diminishingly successively smaller values, r_k = almost zero then, TS: one observation 'above' the mean tends to be followed by more observations 'above' the mean, and one observation 'below' the mean tends to be followed by more observations 'below' the mean
- if r_0 = 1, r_1 = negative value, r_2 = positive value, r_3 = negative, … (diminishingly successively smaller values) then, TS: one observation 'above' the mean tends to be followed by more observations 'below' the mean, aka alternating: successive observations on both sides of the overall mean
- if r_0…k are all positive and large values, TS is non-stationary, and correlogram is meaningless
- if r_0…k oscillate, TS is 'seasonal', e.g. sinusoidal. check at least 3 seasons worth of r_k

Covariance

$$Cov(x,y) = E(xy) - E(x)E(y)$$
$$Var(x+y) = Var(x) + Var(y) + 2Cov(x,y)$$

Basic stochastic models

- residual error series = observed data - fitted values (from the model)
- if our model is good (i.e. captures the deterministic features of the TS)
- then residual TS should be a realization of independent random variables from a probability distribution

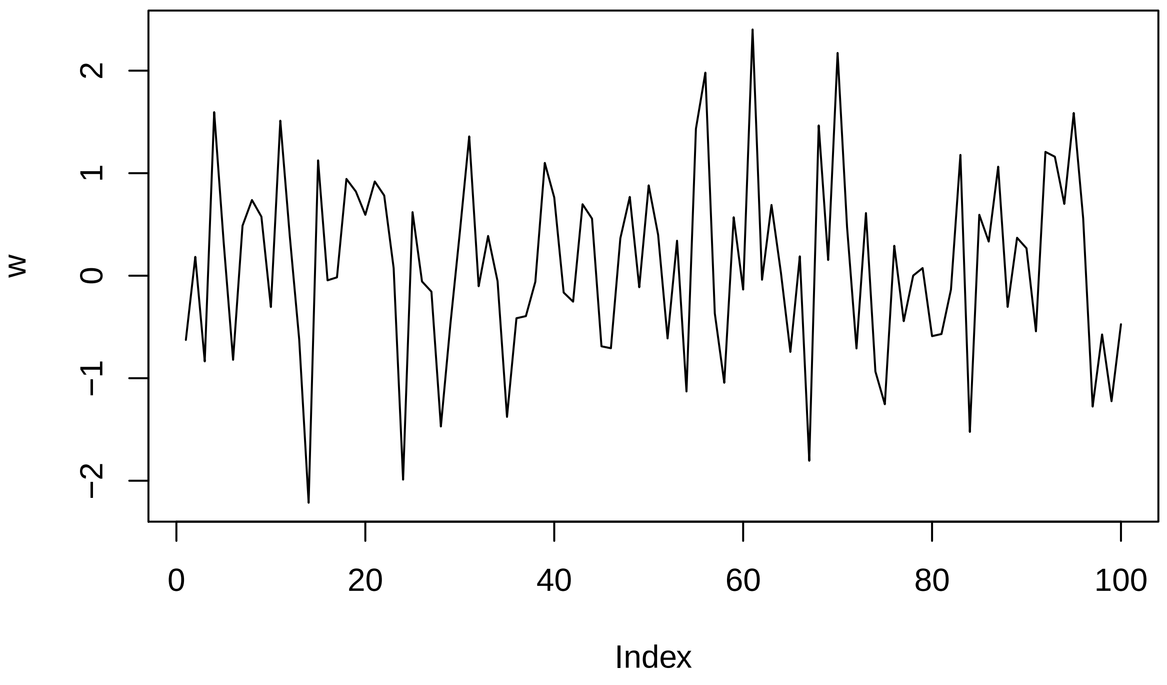
White Noise (WN) [stochastic model for modeling residual TS]

- a TS is discrete WN, if the observations (variables) $w_1, ..., w_t$ are I.I.D. with mean = zero, and all variables have same variance σ^2 and their pairwise Covariance is zero.
- if, additionally, they are normally distributed, then it's Gaussian WN.
- mean = 0
- $Cov(w_i, w_j) = 0 for all i \neq j$

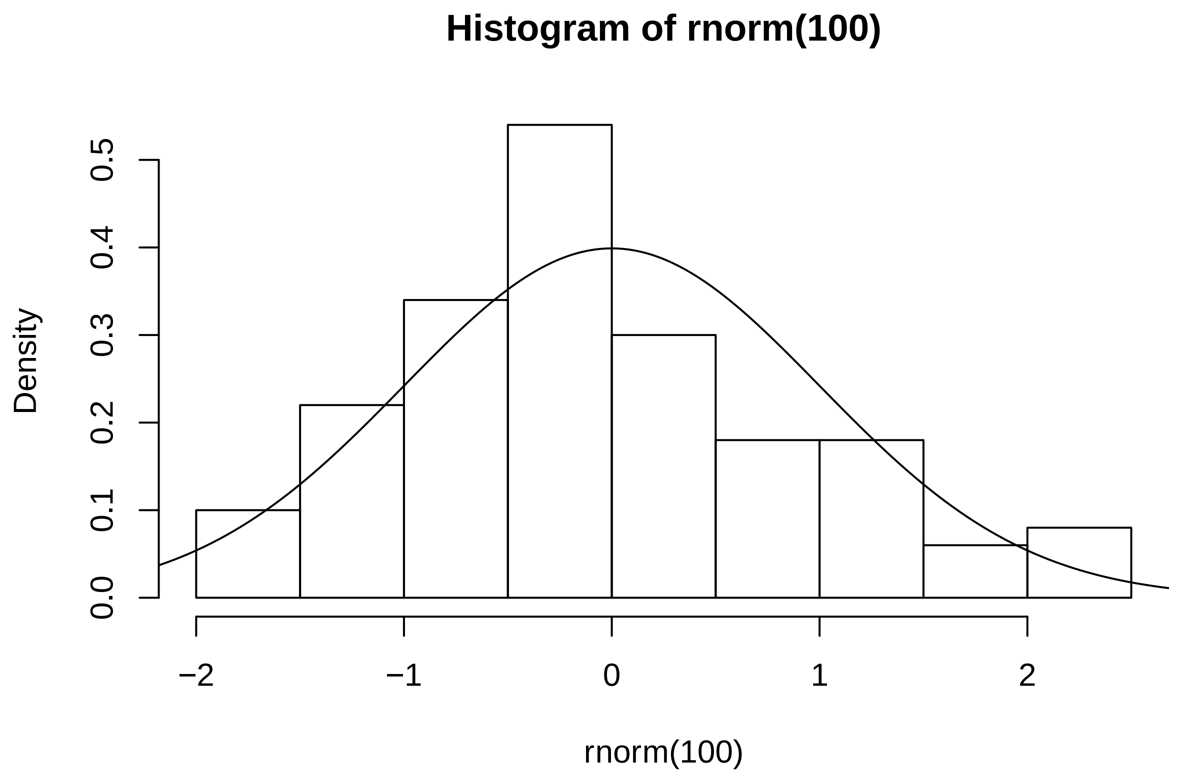
Simulation: to make a synthetic TS (vs. observed TS) Why simulate?

- generate plausible future scenarios
- bootstrapping: constructing confidence intervals for the model parameters

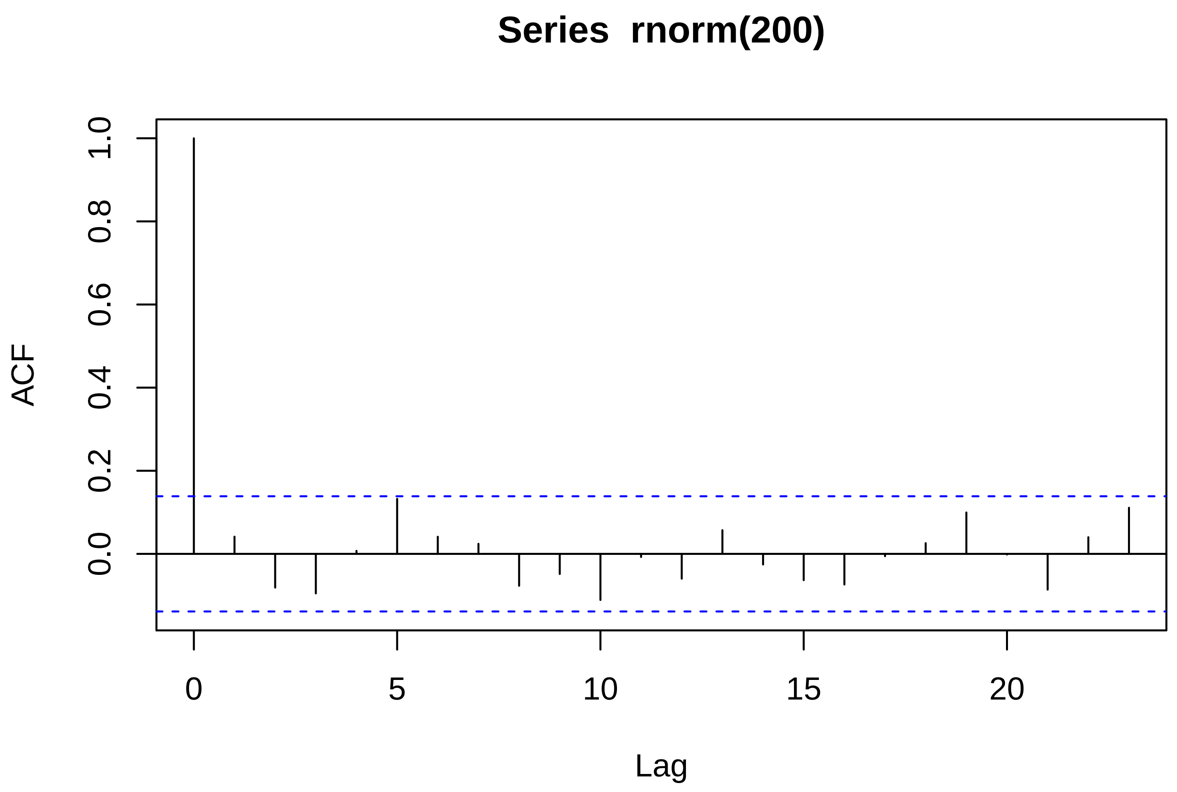
```r
# provide a starting point for the random generation function, to make sure the random generation can b
set.seed(1)

# rnorm simulates 100 random variables that are standard normal and independent
# which can be used as a Gaussian WN TS of length 100
w <- rnorm(100)

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



```r
# ?dnorm
# w2 <- dnorm(w, mean = 0, sd = 1, log = FALSE)
# plot(w2, type = "l")


x <- seq(-3,3, length = 1000)
hist(rnorm(100), prob = T)
points(x, dnorm(x), type = "l")
```



```r
# correlogram of WN TS
set.seed(200)
acf(rnorm(200))
```



```
# the correlations (for lag > 1) is (almost because of sampling variation) zero for almost all (95%)
```

Random Walk (RW) TS [stochastic model for modeling residual TS]

is a good model to fit to data with stochastic trends (not as good as ARIMA though)

TS x_t is a RW, if

$$x_t = x_{t-1} + w_t$$

, where w_t is a WN TS. So, we can back substitute x_{t-1} with $x_{t-2} + w_{t-1}$ and so on, and get:

$$x_t = w_1 + w_2 + w_3 + ... + w_t$$

(i.e. a finite sum of WN, each with mean = zero and var = σ^2)

- mean = 0
- $Cov(x_t, x_{t+k}) = t\sigma^2$ => TS is non-stationary because the Covariance depends on time t. variance increases as t increases => RW model is good ONLY for short-term predictions

backshift(lag) operator (B) $Bx_t = x_{t-1}$ $B^n x_t = x_{t-n}$

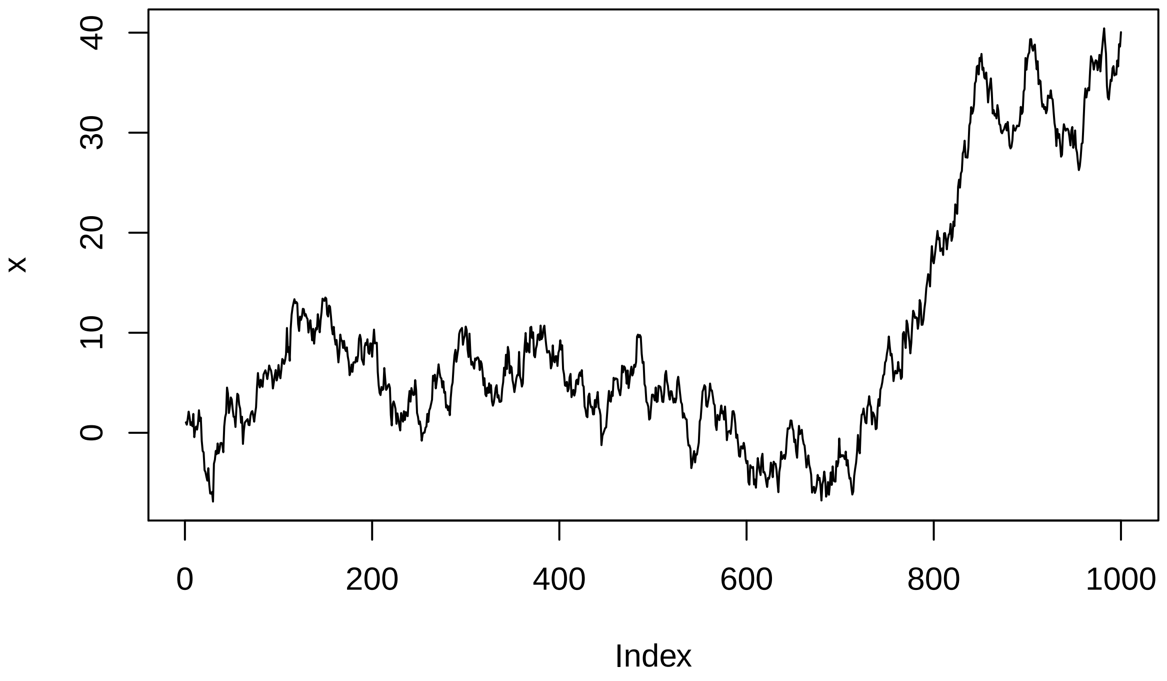
difference operator (to make a stationary TS from a non-stationary TS)

$$\Delta x_t = x_t - x_{t-1}$$

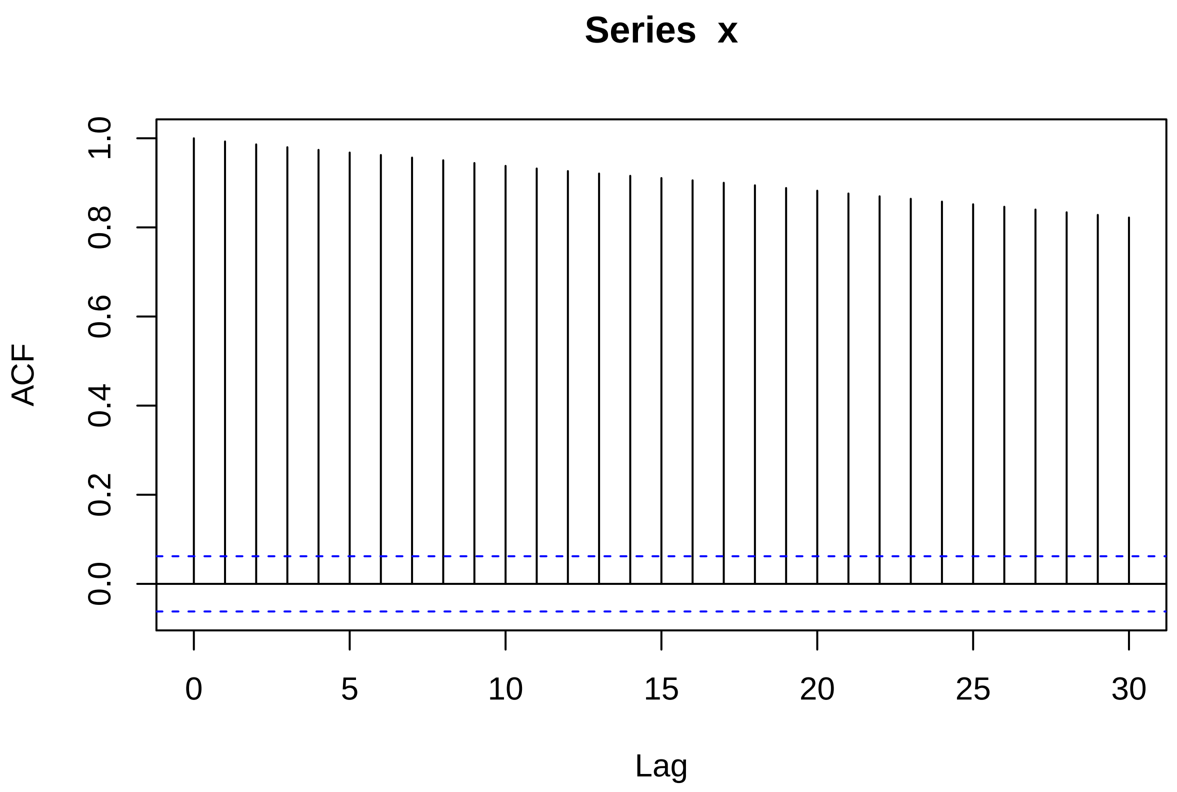
$$\Delta x_t = (1 - B)x_t$$

$$\Delta^n = (1 - B)^n$$

```r
# generate WN TS as w
x <- w <- rnorm(1000)
# make a RW TS using backshift and WN TS
for (t in 2:1000) x[t] <- x[t - 1] + w[t]
# plot the RW TS with lines
plot(x, type = "l")
```



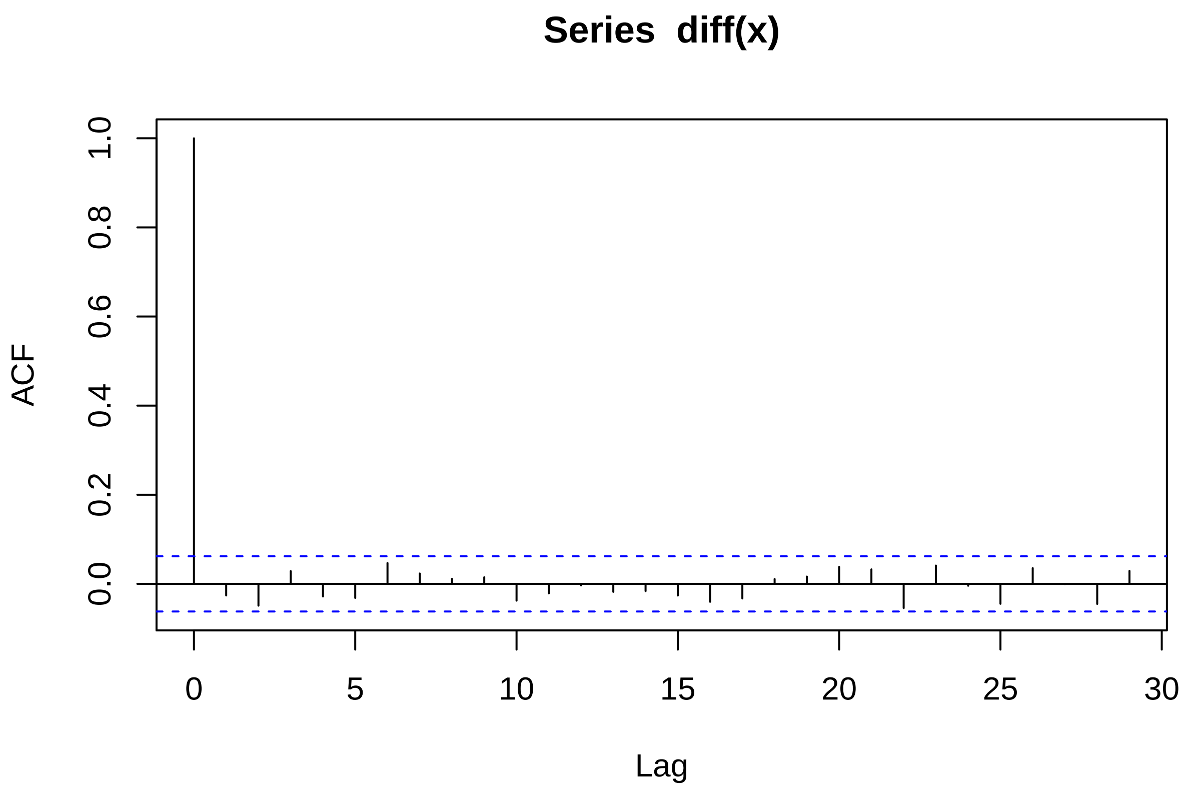
```r
# draw the RW TS correlogram
acf(x)
```



#### diagnosis of a RW TS:

its correlogram should look like this:

```
?diff
# this correlogram shows a RW TS (as a diagnostic tool)
acf(diff(x))
```

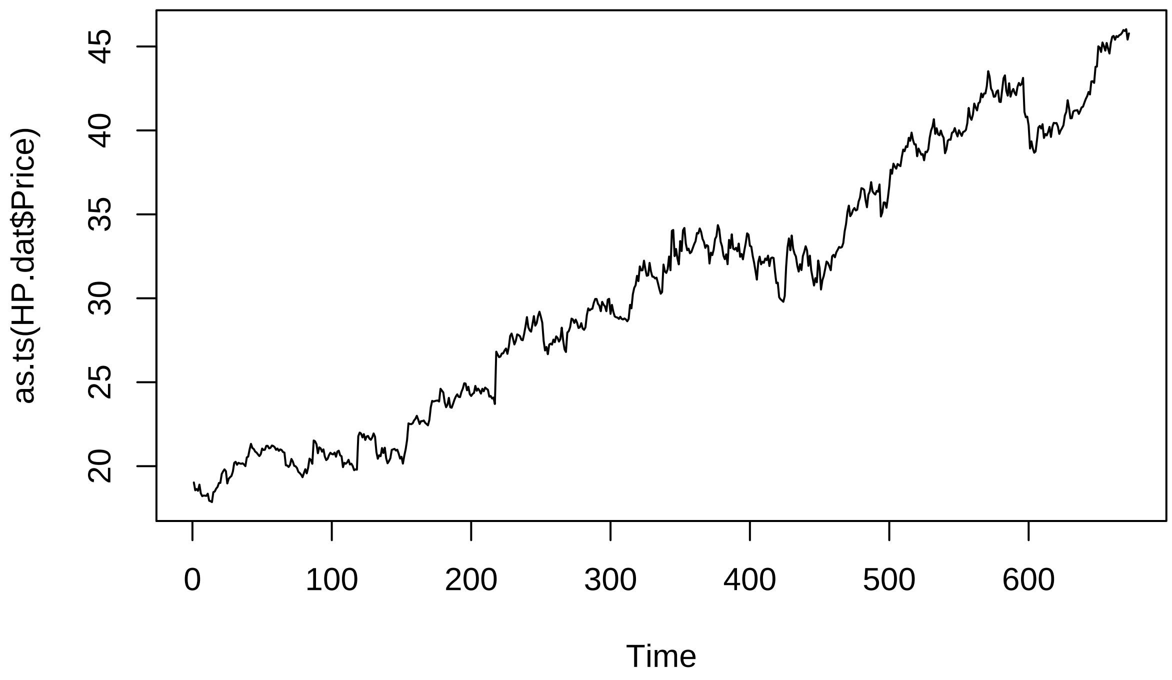


### Random Walk with drift TS [stochastic model for modeling residual TS]

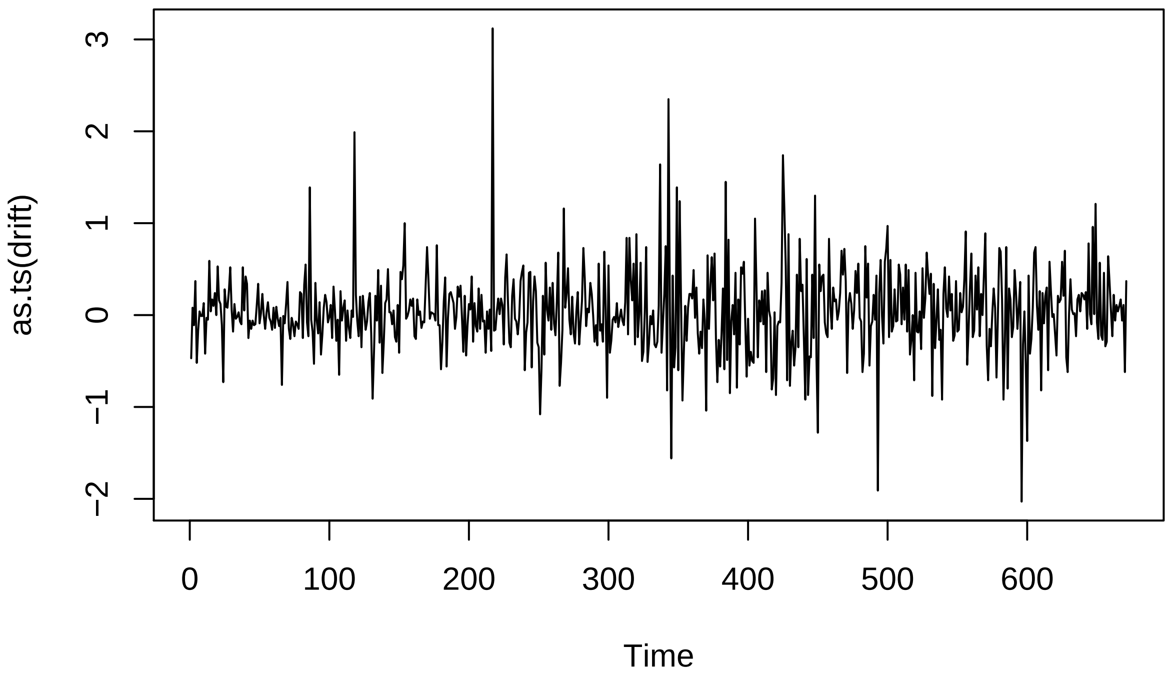
e.g. for stocks, stockholders expect the value of their investment to increase despite the volatility of financial markets. (this increase can be mapped by a drift (upwards) parameter (delta))

$$x_t = x_{t-1} + \delta + w_t$$

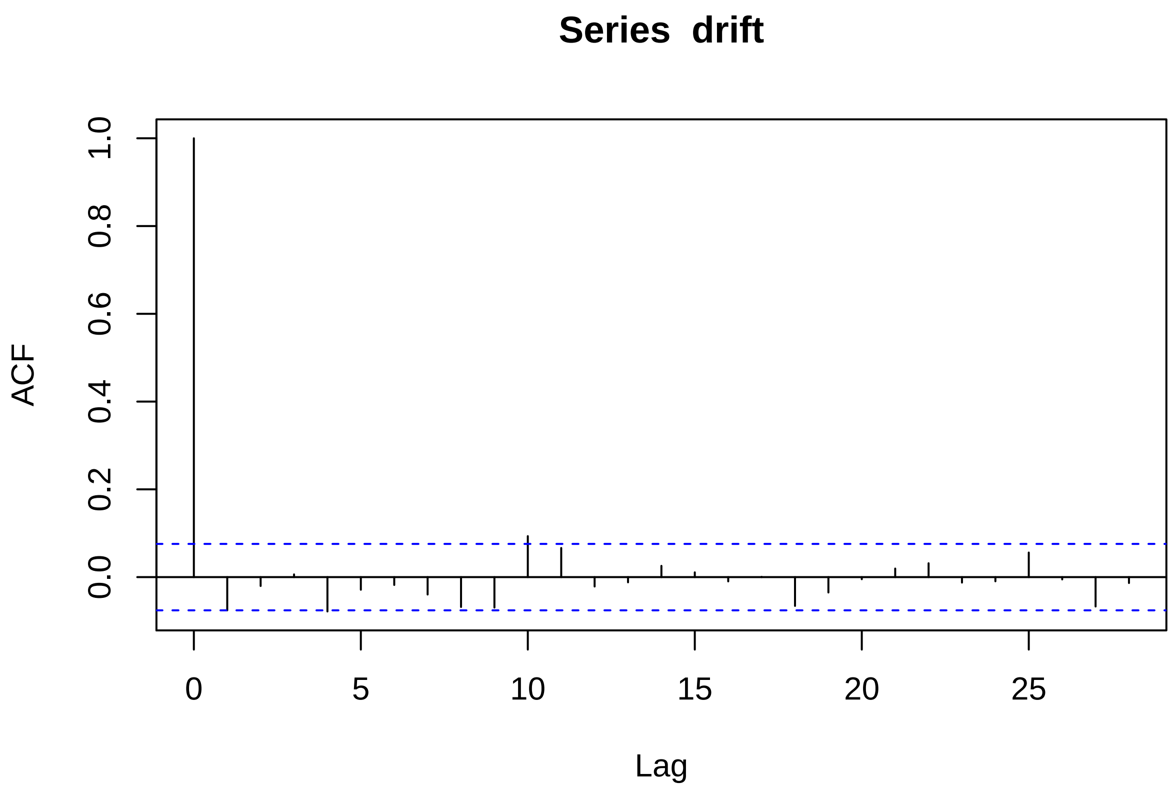
```
# HP stock prices
HP.dat <- read.table("http://staff.elena.aut.ac.nz/Paul-Cowpertwait/ts/HP.txt", header = T)
attach(HP.dat)
# plot the stock price TS
plot (as.ts(HP.dat$Price))
```



```r
# calculate drift parameter \delta
drift <- diff(Price)
plot (as.ts(drift))
```



```r
# check correlogram of drift to make sure it's a WN TS
acf(drift)
```



```
# calculate confidence interval
mean(drift) + c(-2, 2) * sd(drift)/sqrt(length(drift))
```

```
## [1] 0.004378275 0.075353468
```

AutoRegressive TS of order p AR(p)[**stochastic model for modeling residual TS**]

when the model is a regression of x_t on its past terms from the same TS.

$$x_t = \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + ... + \alpha_p x_{t-p} + w_t$$

where

- w_t is WN TS
- α_i are model parameters
- $\alpha_p \neq 0$ for an AR(p) TS

i.e. value of TS at time t depends not only on t-1 but t-2 and … t-p (i.e. p steps back)

$$\theta(B)x_t = (1 - \alpha_1 B^1 - \alpha_2 B^2 - ... - \alpha_p B^p)x_t = w_t$$

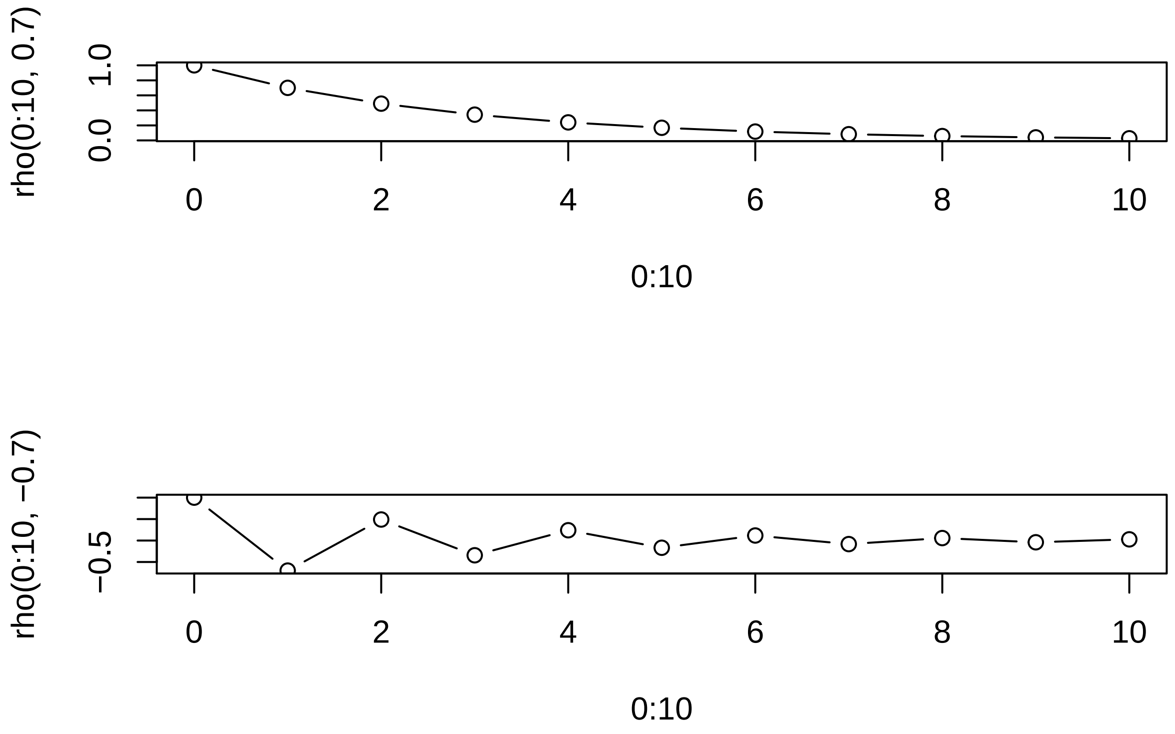
how to tell if an AR(p) TS is stationary or non-stationary? $\theta(B) = 1 - \alpha_1 B^1 - \alpha_2 B^2 - ... - \alpha_p B^p$ is called characteristic equation (CE).

- If all roots of CE for an AR(p) are > 1, (their absolute value is > 1), AR(p) is stationary.
- If even one of the roots of CE is $<= 1$ in absolute value, then AR(p) is non-stationary.

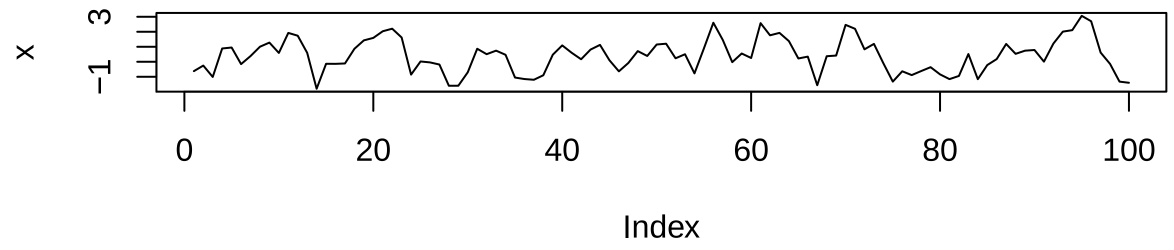
```
# find roots of a polynomial
polyroot(c(1, 2, 1))
```

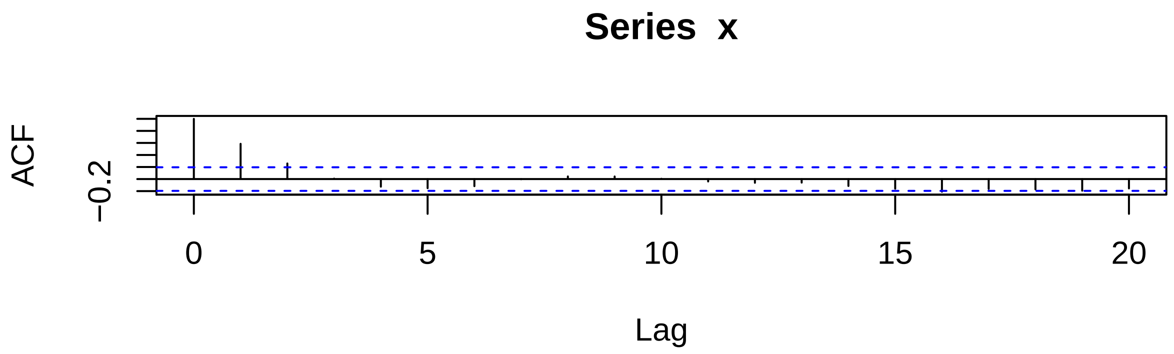
```
## [1] -1-0i -1+0i
```

```
rho <- function(k, alpha) alpha^k
layout(1:2)
plot(0:10, rho(0:10, 0.7), type = "b")
plot(0:10, rho(0:10, -0.7), type = "b")
```

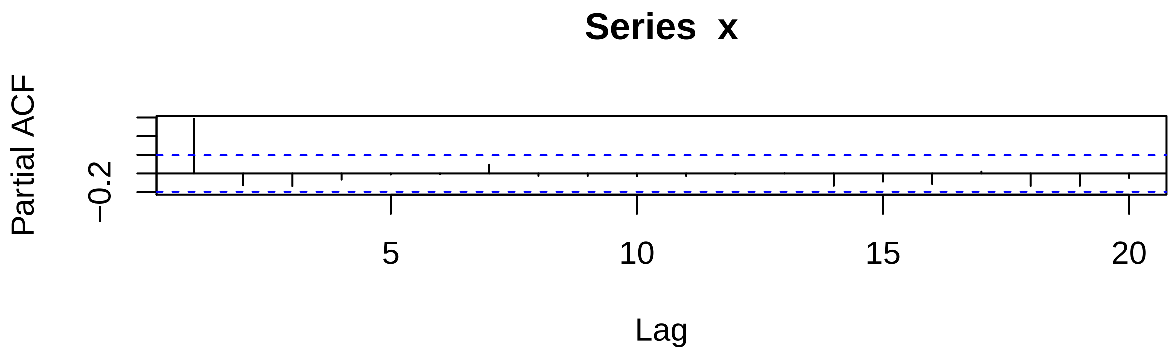


```
set.seed(1)
x <- w <- rnorm(100)
for (t in 2:100) x[t] <- 0.7 * x[t - 1] + w[t]
plot(x, type = "l")
acf(x)
```





```r
pacf(x)
```



#### AR(p) TS model fitting in R

```r
# fit an AutoRegression AR(p) model
x.ar <- ar(x, method = "mle")

x.ar$order
```

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

```r
# parameter estimate (alpha bar)
x.ar$ar
```

```
## [1] 0.6009459
```

```r
x.ar$asy.var
```

```
##              [,1]
## [1,] 0.006443494
```

```
x.ar$ar + c(-2, 2) * sqrt(x.ar$asy.var)
```

```
## [1] 0.4404031 0.7614886
```

```
# str(x.ar)
```

ARMA(p,q) models: AR(p) + MA(q) combined together: AutoRegressive Moving Average

- AR(p):
$$x_t = \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + ... + \alpha_p x_{t-p} + w_t$$
- MA(q):
$$x_t = w_t + \beta_1 w_{t-1} + ... + \beta_q w_{t-q}$$
- ARMA(p,q):
$$x_t = \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + ... + \alpha_p x_{t-p} + w_t + \beta_1 w_{t-1} + ... + \beta_q w_{t-q}$$

$$\theta_p(B)x_t = \phi_q(B)w_t$$

- ARMA(p,q) TS is STATIONARY if roots of $\theta > 1$ in absolute value
- ARMA(p,q) TS is INVERTIBLE if roots of $\phi > 1$ in absolute value
- ARMA(p, 0) = AR(p)
- ARMA(0,q) = MA(q)
- ARMA(p,q) is better (parameter efficient) than either a single AR(p) or MA(q)

```
set.seed(1)
# simulate an ARMA data with alpha = -0.6 and beta = 0.5
x <- arima.sim(n = 10000, list(ar = -0.6, ma = 0.5))
# fit an ARMA(1,1) model to the simulated data using c(p,0,q)
coef(arima(x, order = c(1, 0, 1)))
```

```
##          ar1          ma1    intercept
## -0.596966371  0.502703368 -0.006571345
```

Now, let's compare AR(p), MA(q), and ARMA(p,q)

```
x.ma <- arima(x.ts, order = c(0, 0, 1))
x.ar <- arima(x.ts, order = c(1, 0, 0))
x.arma <- arima(x.ts, order = c(1, 0, 1))
AIC(x.ma)
```

```
## [1] -3.526895
```

```
AIC(x.ar)
```

```
## [1] -37.40417
```

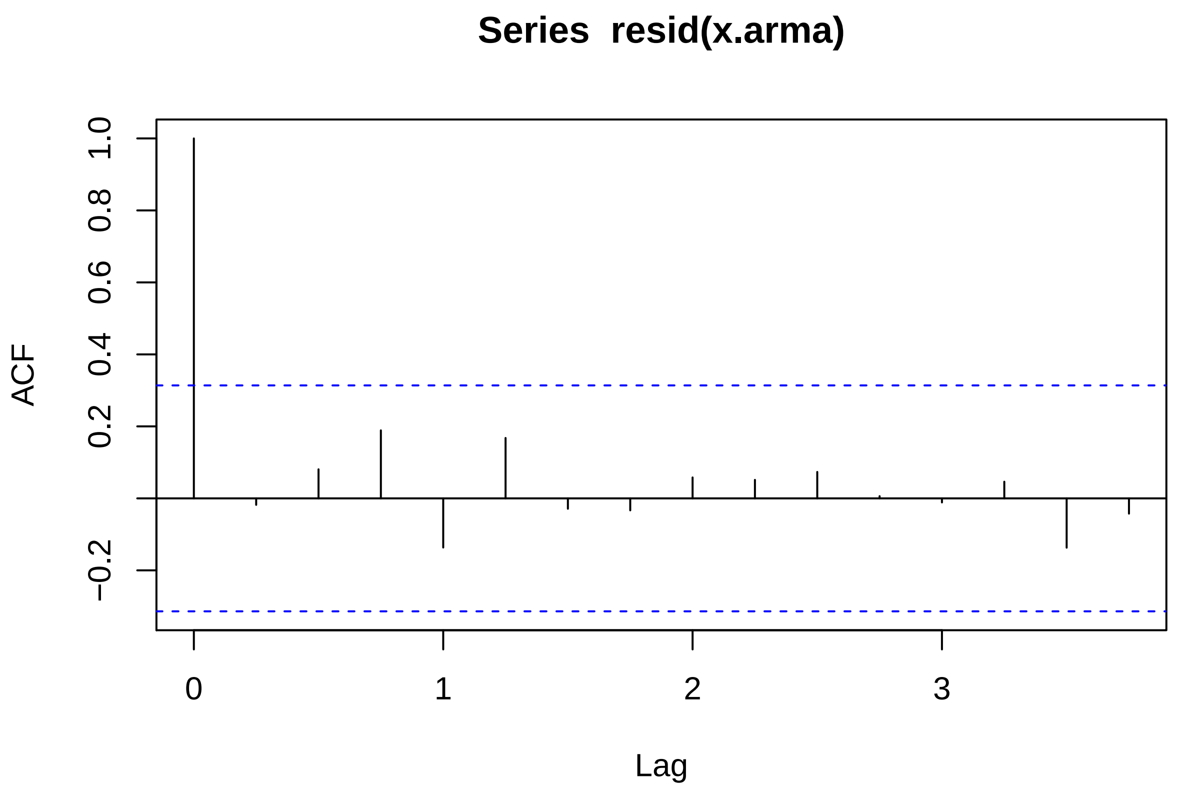
```
AIC(x.arma)
```

```
## [1] -42.27357
```

```
# the smaller the AIC or BIC, the better the fit => ARMA(1,1) provides the best fit
x.arma
```

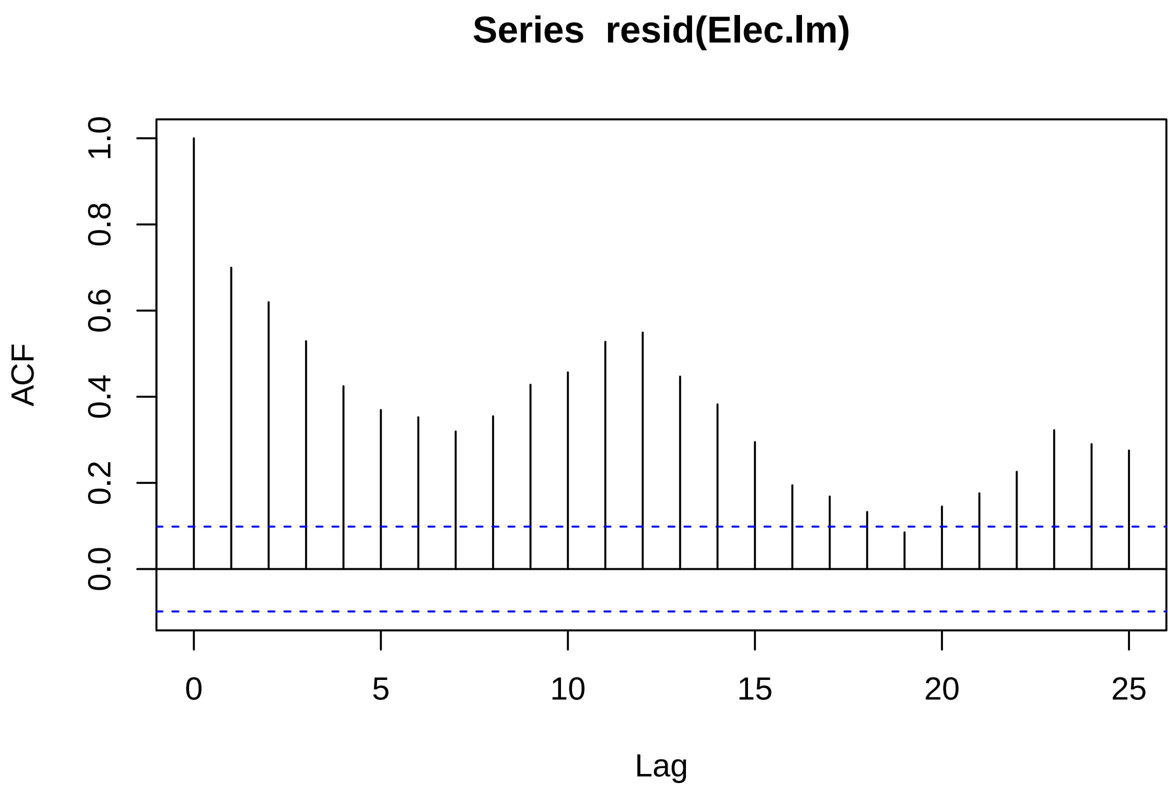
```
##
## Call:
## arima(x = x.ts, order = c(1, 0, 1))
##
## Coefficients:
##          ar1     ma1  intercept
##       0.8925  0.5319     2.9597
## s.e.  0.0759  0.2021     0.2435
##
## sigma^2 estimated as 0.01505:  log likelihood = 25.14,  aic = -42.27
```

```
# correlogram of residuals of fitted ARMA(1,1) model
acf(resid(x.arma))
```



example: electricity production series

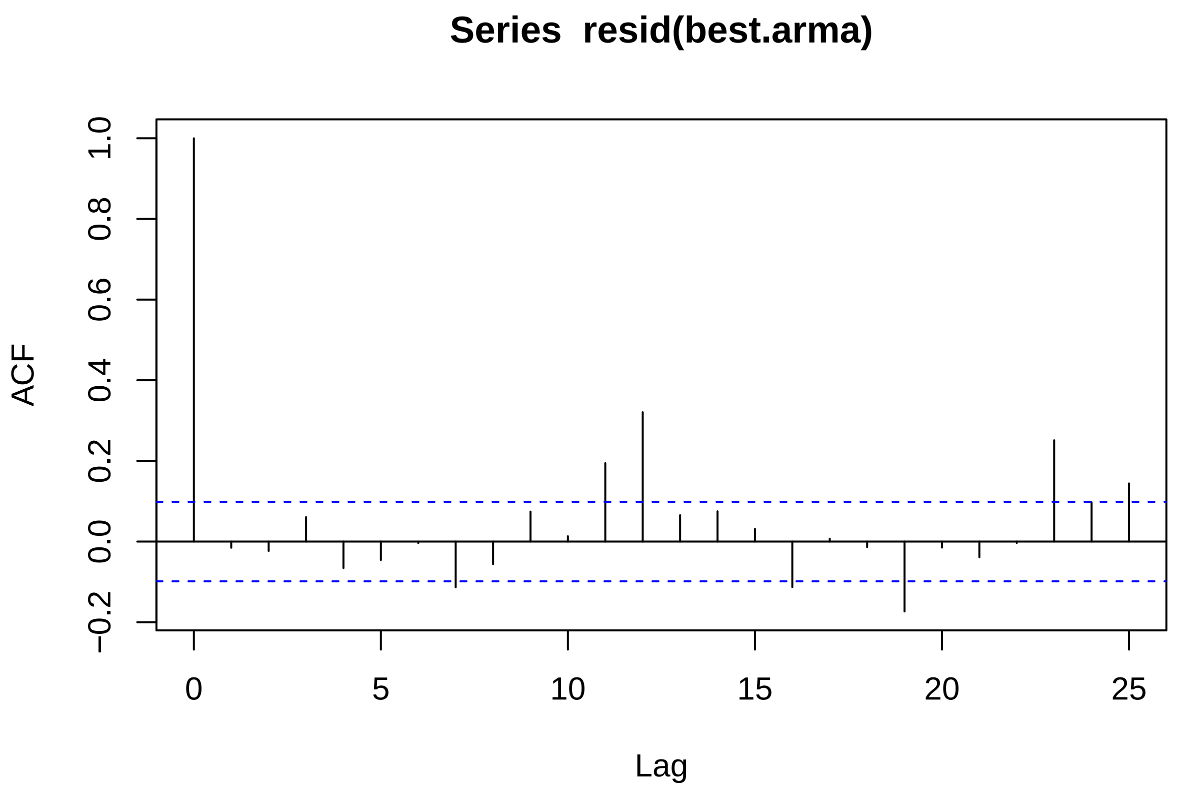
```
CBE <- read.table("http://staff.elena.aut.ac.nz/Paul-Cowpertwait/ts/cbe.dat", header = T)
Elec.ts <- ts(CBE[, 3], start = 1958, freq = 12)
Time <- 1:length(Elec.ts)
Imth <- cycle(Elec.ts)
Elec.lm <- lm(log(Elec.ts) ~ Time + I(Time^2) + factor(Imth))
acf(resid(Elec.lm))
```



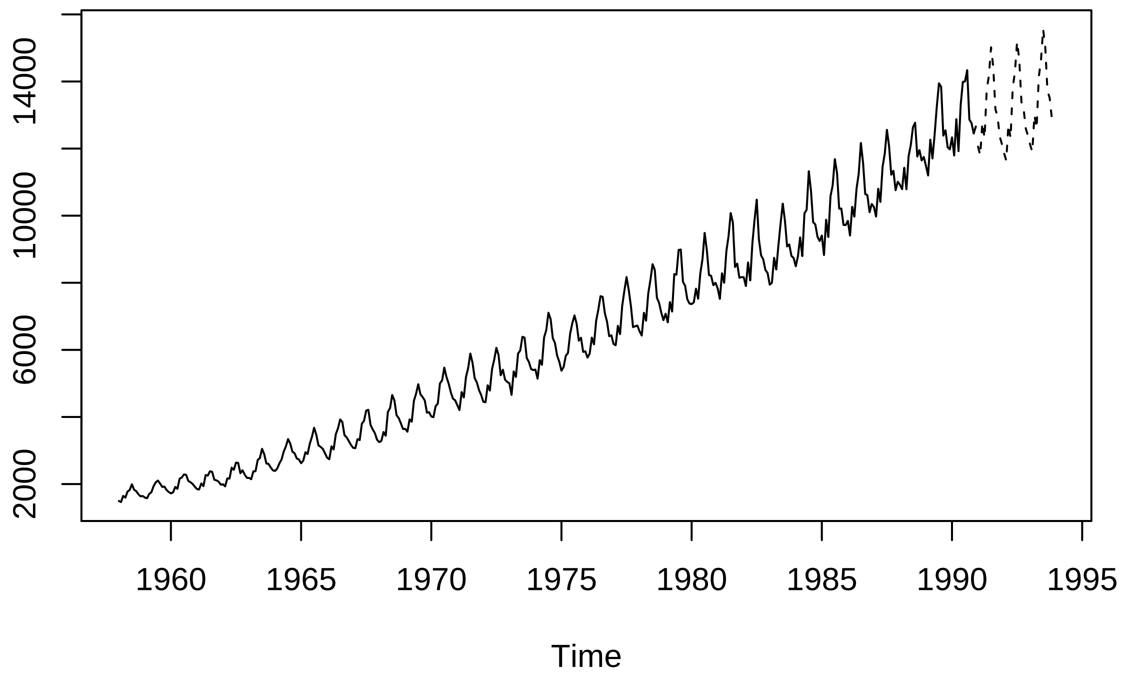
```
# find the best p and q for ARMA(p,q)
best.order <- c(0, 0, 0)
best.aic <- Inf
for (i in 0:2) for (j in 0:2) {
  fit.aic <- AIC(arima(resid(Elec.lm), order = c(i, 0,j)))
  if (fit.aic < best.aic) {
    best.order <- c(i, 0, j)
    best.arma <- arima(resid(Elec.lm), order = best.order)
    best.aic <- fit.aic
  }
}
# what's the best order found:
best.order
```

```
## [1] 2 0 0
```

```
# correlogram
acf(resid(best.arma))
```



```
## prediction
new.time <- seq(length(Elec.ts), length = 36)
new.data <- data.frame(Time = new.time, Imth = rep(1:12,3))
# predict using the fitted regression model of class lm
predict.lm <- predict(Elec.lm, new.data)
# predict using the fitted ARMA model, number of timesteps ahead to predict; here = 36 ~ predict for th
predict.arma <- predict(best.arma, n.ahead = 36)
# convert the predicted values to TS object
elec.pred <- ts(exp(predict.lm + predict.arma$pred), start = 1991, freq = 12)
ts.plot(cbind(Elec.ts, elec.pred), lty = 1:2)
```


14000
10000
6000
2000
1960
1965
1970
1975
1980
1985
1990
1995
Time