Discrete Response, Time Series and Panel Data - Notes

Nils Rechberger

2026-02-26

W-01-Timeseries-Introduction-Visualisation-Stationarity

> **Note**
>
> Coming soon...

W-02-Timeseries-Introduction-Visualisation-Stationarity

Log-transformations

Instead of the original time series , consider the log-transformed time series with the elements.

$$g_t = \log(X_t)$$

When to apply it?:

- When the distribution of the time series elements is right-skewed.
- When a time series could be produced by the product of model outcomes.

Differencing

Let's take a linear function (defined on a set of integer numbers)

$$X_t = m \cdot t + b$$

Then, the backward difference

$$X_t - X_{t-1} = (m \cdot t + b) - (m \cdot (t-1) + b) = m$$

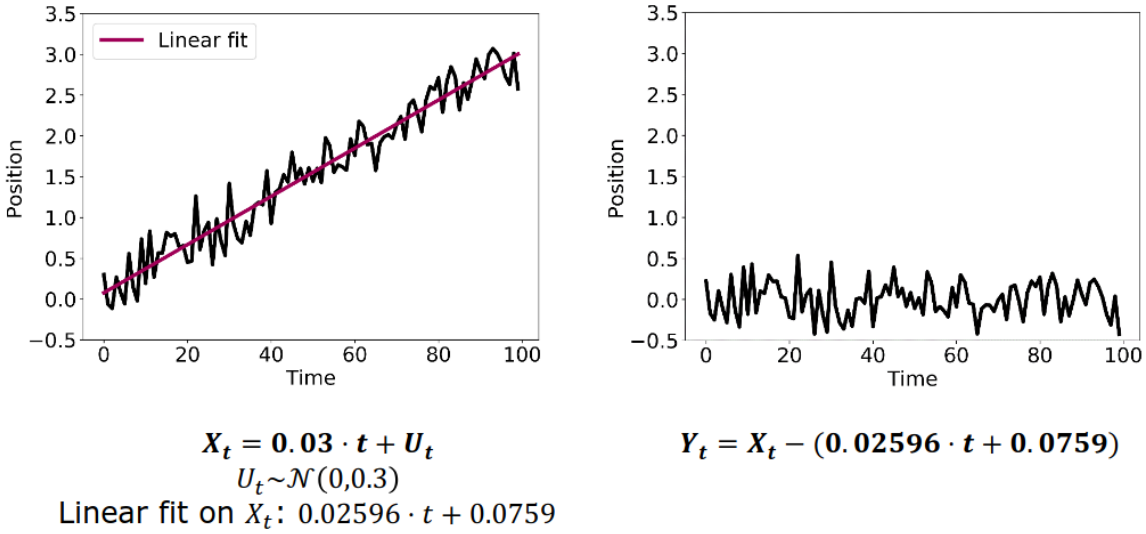


Figure 1: Differencing

Higher Order Differencing

Use multiple derivatives to eliminate lower-order terms. Trend order is reduced, but the stationary part may be more complex. After trend is eliminated, maybe a new dependencies in the stationary part.

Example

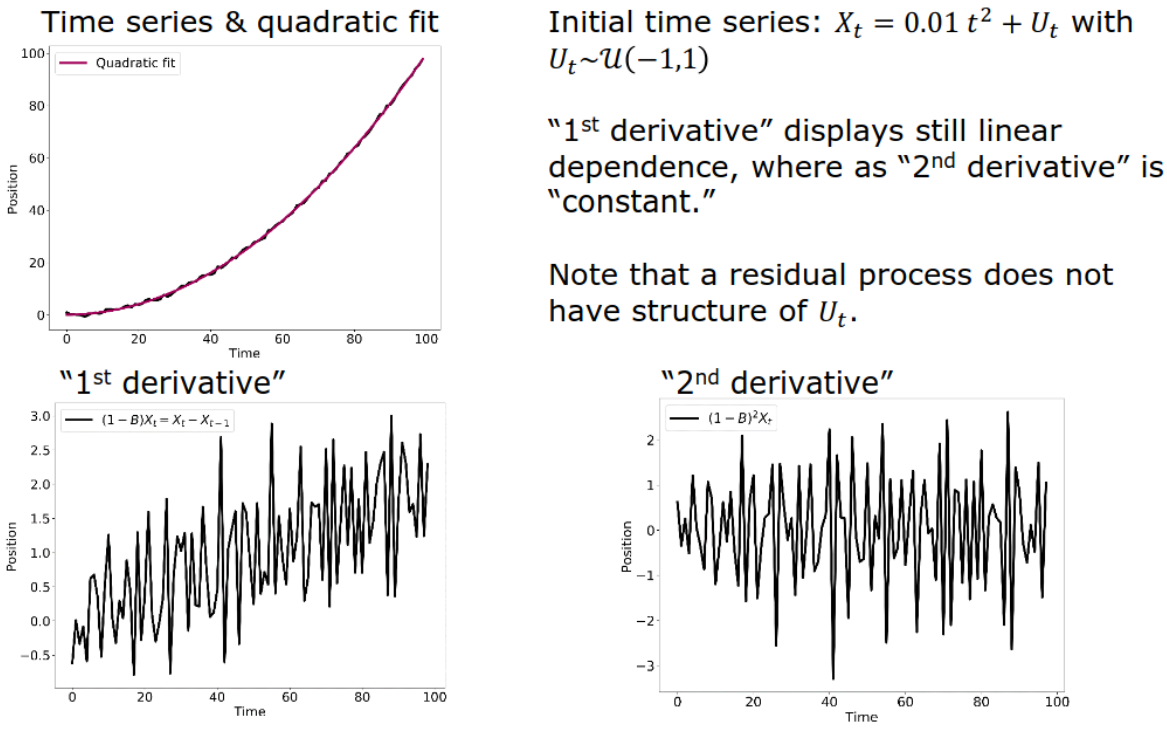
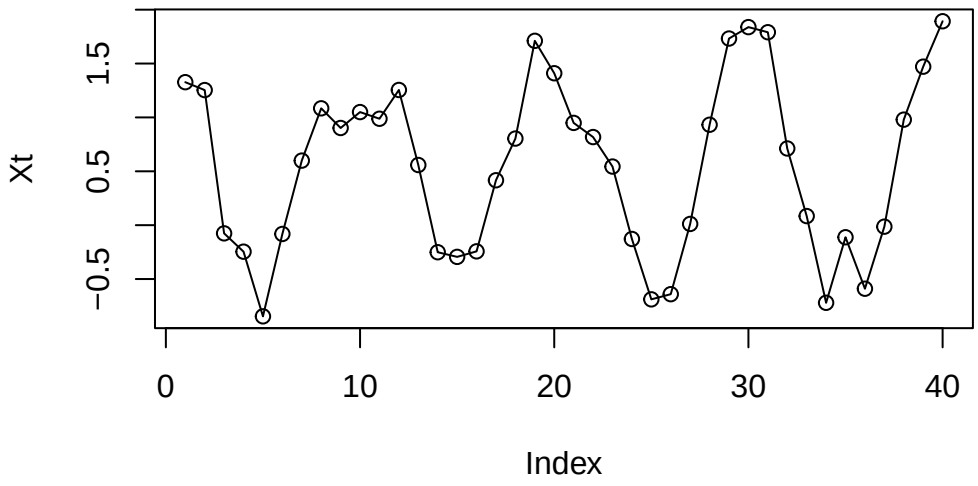


Figure 2: Remove Higher Order Trend

Diferencing Seasonality

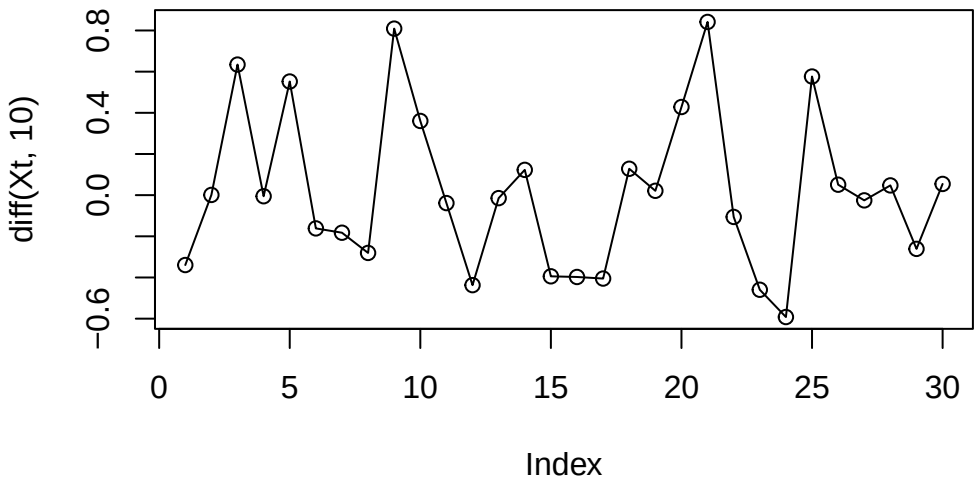
Seasonality manifests itself by a repetitive behavior after a fixed lag time p.

```
tRange<-1:40
Xt<-cos(2*pi/10*tRange)+runif(40)
plot(Xt,type="o")
```



What happens if we differentiate now with lag 10?

```
plot(diff(Xt,10),type="o")
```



Summary of differencing

Advantages:

- can remove trends and seasonality effects
- is easy to use

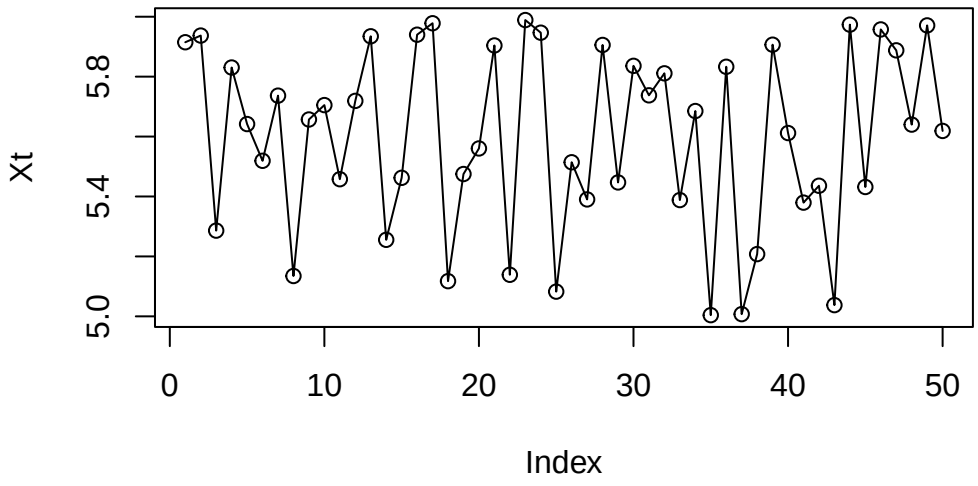
Disadvantages:

- the resulting time series is shorter than the original one
- the results of differencing may display dependencies among different elements
- the original decomposition elements are unknown

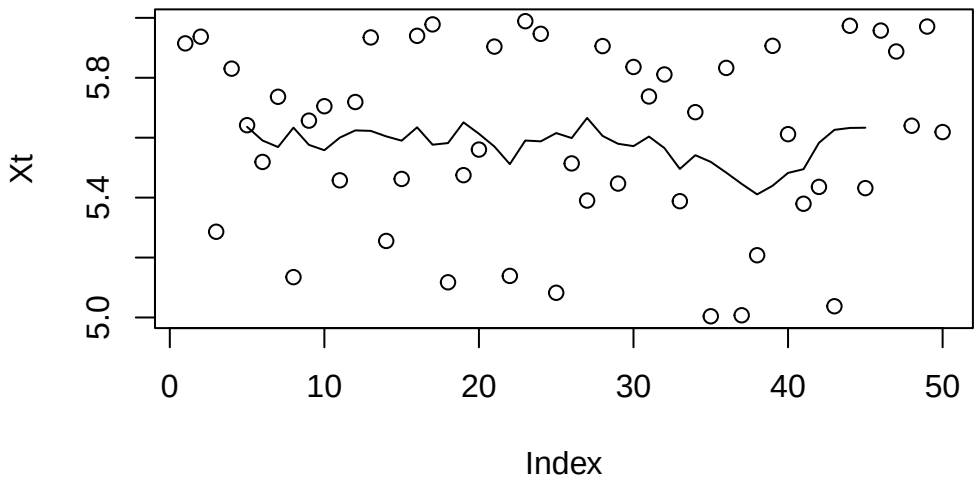
Filtration

To reduce the noise element in an experiment, you can repeat the measurement multiple times and then calculate the average.

```
set.seed(42)
Xt<-5+runif(50)
plot(Xt,type="o")
```



```r
filtered<-filter(Xt,filter=rep(1,10)/10)
plot(Xt)
points(filtered,type="l")
```



Reasons for using Filtration

By averaging measurements, we obtain a new averaged time series. By increasing the number of averaged measurements, the parts converges to the expectation value of the measurement error, i.e. 0.

Note: The short-term variations are smoothed out and the trend is confirmed.

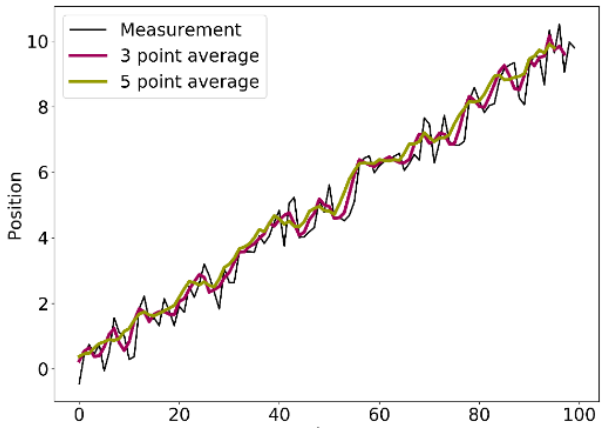


Figure 3: Filtering

Linear filters to quantify seasonality effects

In case of seasonality effects, the filter has to expand over the full period to extract the trend. To quantify the seasonal impact, the values of the same time in the season have to be averaged.

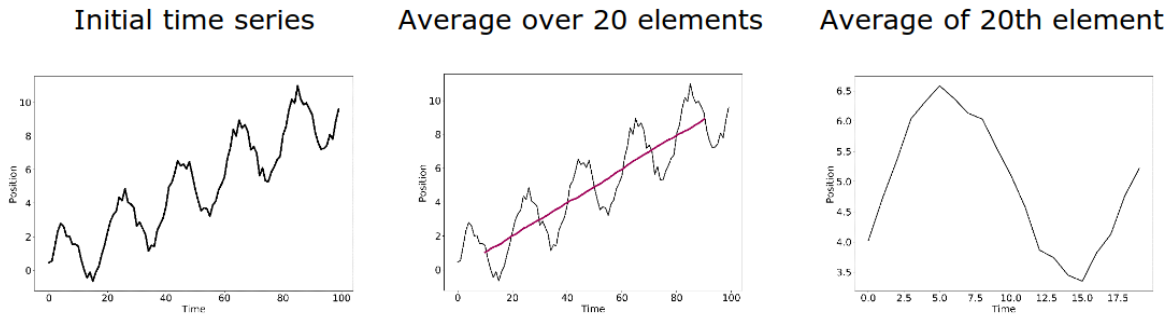


Figure 4: Quantify Seasonality Effects

Seasonal-trend decomposition by Loess Algorithm (STL)

Linear filters are great if periodicity is obvious; but in less obvious cases, extracting the individual components may be laborious. An alternative is the seasonal-trend decomposition procedure by Loess: An iterative, non-parametric smoothing algorithm that is more robust.

Fitting of Periodic Function

If you know the generative model, you can use fitting to determine the contributions of trend and seasonality. Fitting methods (among others):

- Ordinary least square (OLS)
- Robust fitting (to prevent the impact of outliers)
- GLS (generalised least square) / time series regression methods

```r
set.seed(42)
#Generate the sample data set
t<-seq(1,100,length=100)
data<-0.1*t+cos(2*pi/10*t)+runif(100)
ts<-ts(data)
#Fit the model
fit<-lm(ts~t+cos(2*pi/10*t))
summary(fit)
```

```
Call:
lm(formula = ts ~ t + cos(2 * pi/10 * t))

Residuals:
     Min       1Q   Median       3Q      Max
-0.58266 -0.22271  0.04478  0.25411  0.49285

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)        0.639744   0.059840   10.69   <2e-16 ***
t                  0.097718   0.001029   94.98   <2e-16 ***
cos(2 * pi/10 * t) 0.973247   0.041999   23.17   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2969 on 97 degrees of freedom
Multiple R-squared:  0.9901,    Adjusted R-squared:  0.9899
F-statistic:  4836 on 2 and 97 DF,  p-value: < 2.2e-16
```

Autocorrelation

The autocorrelation depends only on the lag for weakly or strongly stationary time series.

$$p(k) := Cor(X_{t+k}, X_t)$$

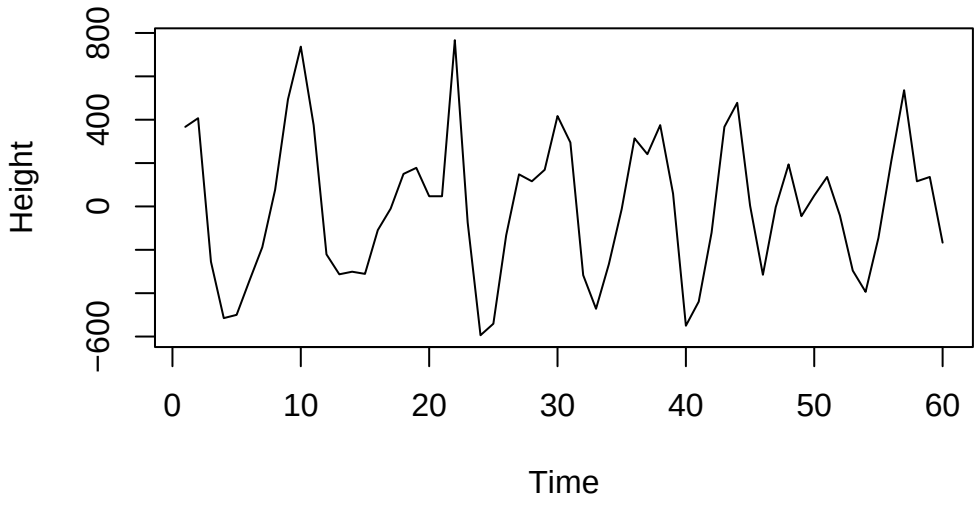
Interpretation of autocorrelation

The correlation $p(k)^2$ squared corresponds to the percentage of variability explained by the linear association between X_t and X_{t+k}.

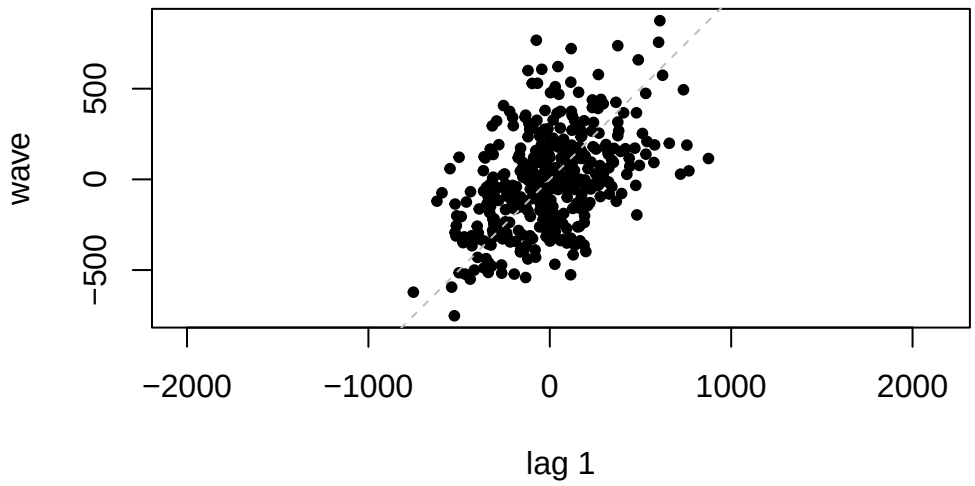
Lagged Scatter Plot

Exchange average over realizations with average over time.

```
address <- "https://raw.githubusercontent.com/AtefOuni/ts/master/Data/wave.dat"
dat <- read.table(address,header=T)

#Generate ts object
wave<-ts(dat$waveht)

#Visualize the data
plot(window(wave,1,60),ylab="Height")
```

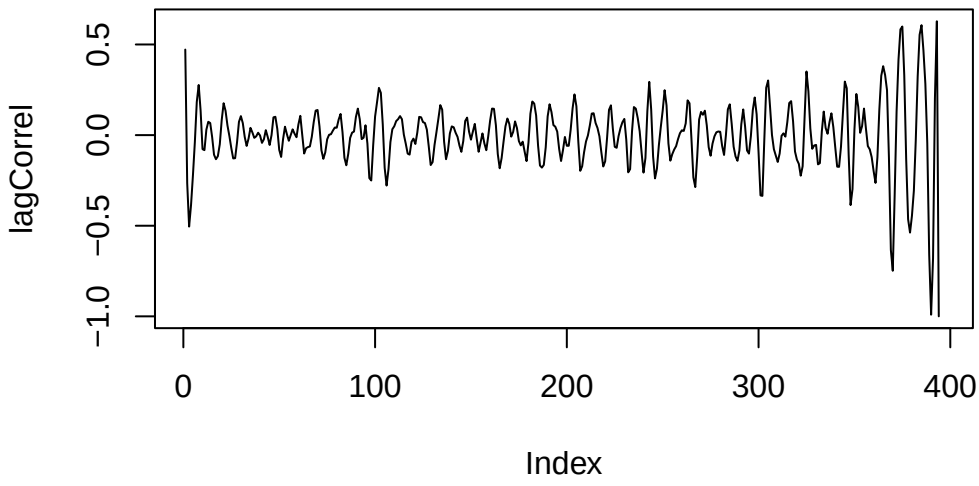


```r
#Lag plot
lag.plot(wave,do.lines=FALSE,pch=20)
```



Calculate lagged scatter plot estimator

```r
# Calculate the Pearson Correlation coefficients
n <- length(wave)
lagCorrel <- rep(0,n)

for (i in 1:(n-1)){
lagCorrel[i]=cor(wave[1:(n-i)],wave[(i+1):n])
}

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



Plug-In Estimation

The plug-in estimator is the standard approach to computation autocorrelations. Calculated with an `acf` function in R.

Summary Autocorrelation

- (Auto)correlation measures the linear association at a given lag
- For realizations, the autocorrelation is estimated by running the calculations over time instances instead of realizations (therefore stationarity is desired)
  - Lagged scatter plot estimator
  - Plug-in estimator
- For high lags, the lagged scatter plot estimator diverges (because too few time points are considered)
- The plug-in estimator corrects the diverging behavior but generates a bias.

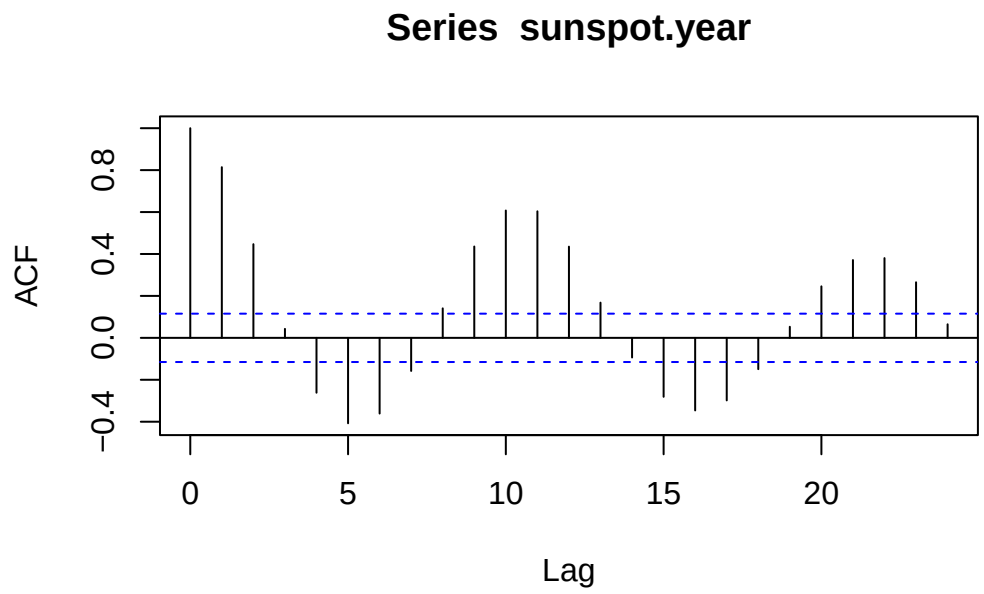
> ⚠ Warning
>
> Warning: Non-linear relations may confuse the autocorrelation.

W-03-Timeseries-ACF-AR

Correlogram

Standard command: `acf(data)` for plug-in estimator values.

```
data(sunspot.year)
acf(sunspot.year)
```



> **Caution**
>
> The first lag at lag 0 always have to be a value of 1

Bounds of Autocorrelation

Under the null hypothesis of a time series process with iid distributed random variables, a 95 % acceptance region for the null hypothesis is:

$$\pm \frac{1.96}{\sqrt{n}}$$

For stationary processes, plug-in estimator value $\hat{p}(k)$ within the confidence band $\pm\frac{1.96}{\sqrt{n}}$ are considered to be different from 0 only by chance.

Ljung-Box Test

The Ljung-Box test assesses the null hypothesis that the random variables are independent when the h autocorrelation coefficients $\hat{p}(k)$ for $k = 1, \ldots, h$ are equal to zero.

```
Box.test(sunspot.year, lag=10, type="Ljung-Box")
```

```
    Box-Ljung test

data:  sunspot.year
X-squared = 542.41, df = 10, p-value < 2.2e-16
```

Characteristic Patterns in ACF

Patterns in ACF for non-stationary time series

For non-stationary time series, the ACF decays only slowly.

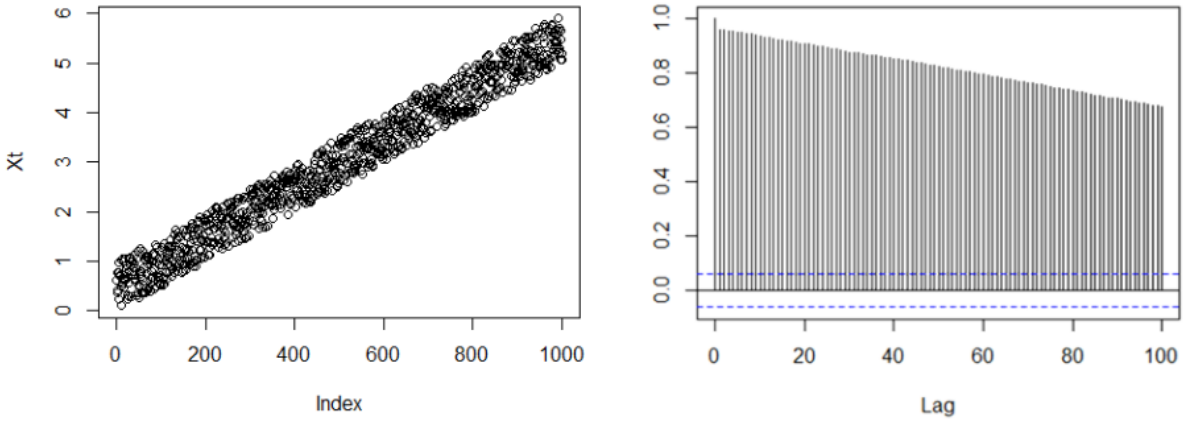


Figure 5: Non-Statiionary time seires

ACF of time series with seasonal pattern

For time series with seasonality, the ACF displays periodic structure.

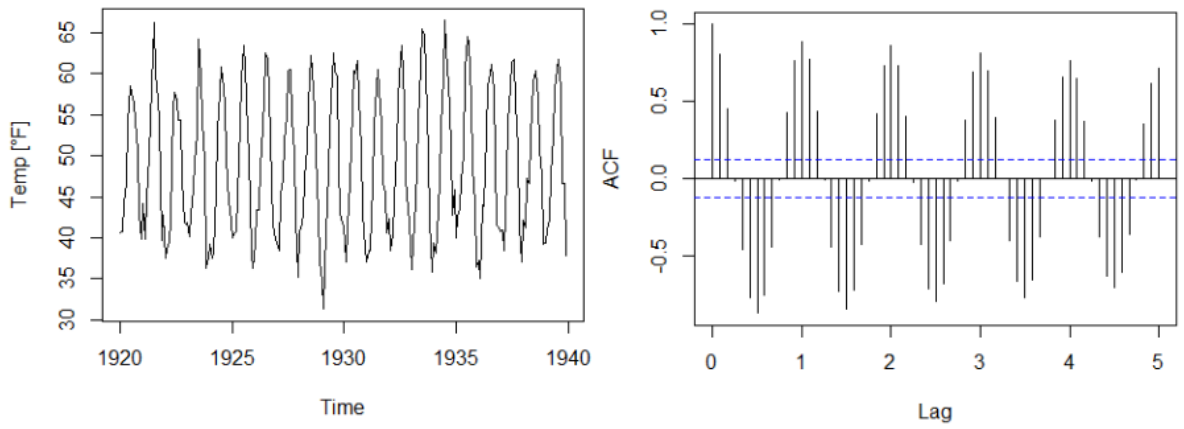


Figure 6: Time series with seasonal pattern

ACF of time series with seasonal pattern and trend

Time series with seasonality and trend, the ACF displays periodicity and slow decay.

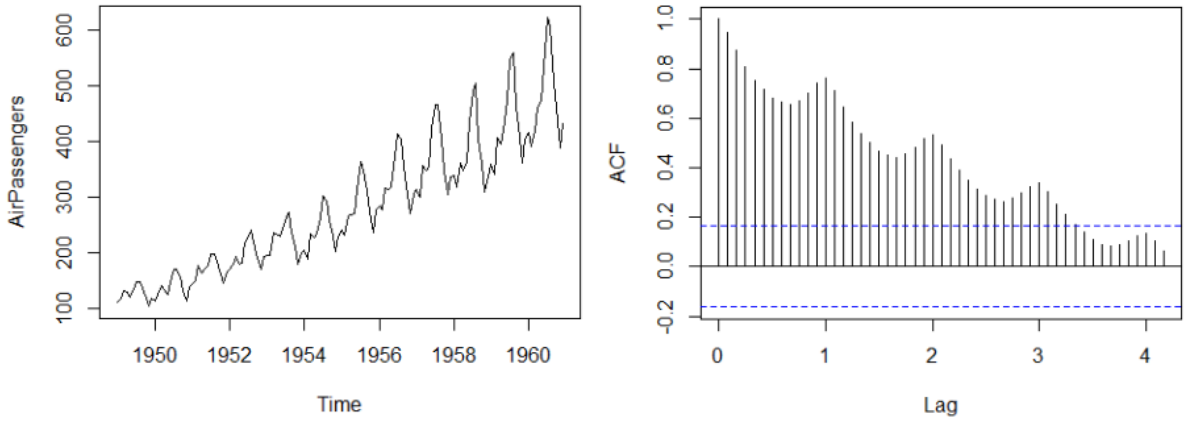


Figure 7: Time series with seasonal pattern and trend

ACF of time series with outliers

Outliers may be diagnosed by lagged scatter plots. Note that each outlier causes other outliers.

> **!** Important
>
> The plug-in estimator $\hat{p}(k)$ is sensitive to outliers.

Properties of ACF

Autocorrelation of process vs. estimated values of realization

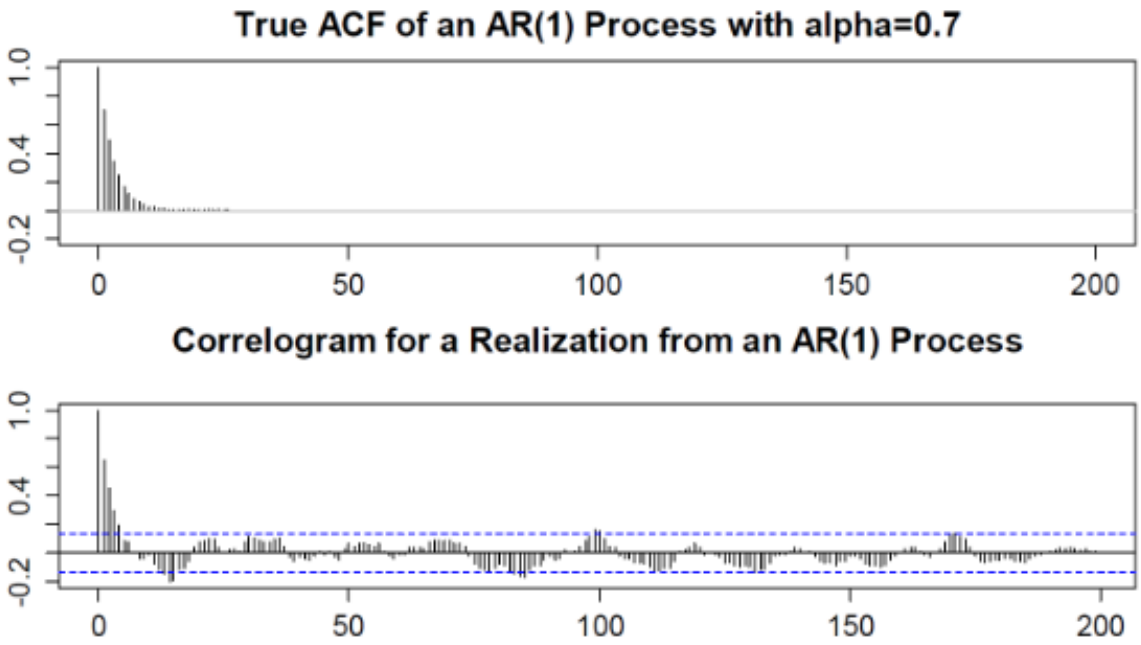
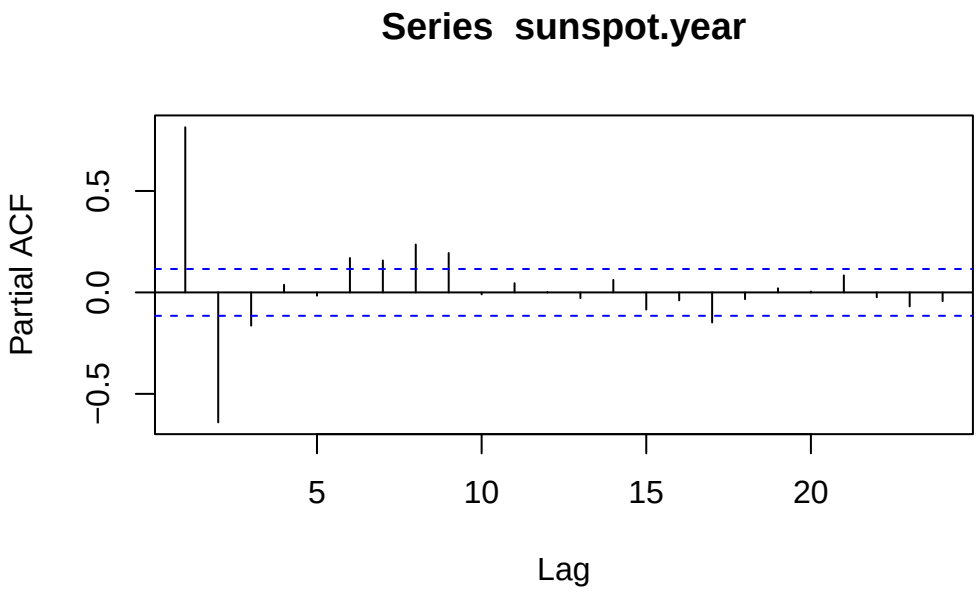


Figure 8: Correlogram of one realization compare with the true plot

Partial Autocorrelation

A challenge in the interpretation of ACFs is that the effect of smaller lags remains. The partial autocorrelation function (PACF) gives the partial correlation of a stationary time series with its own lagged values, regressed the values of the time series at all shorter lags.

```
pacf(sunspot.year)
```



W-04-Timeseries-ACF-AR

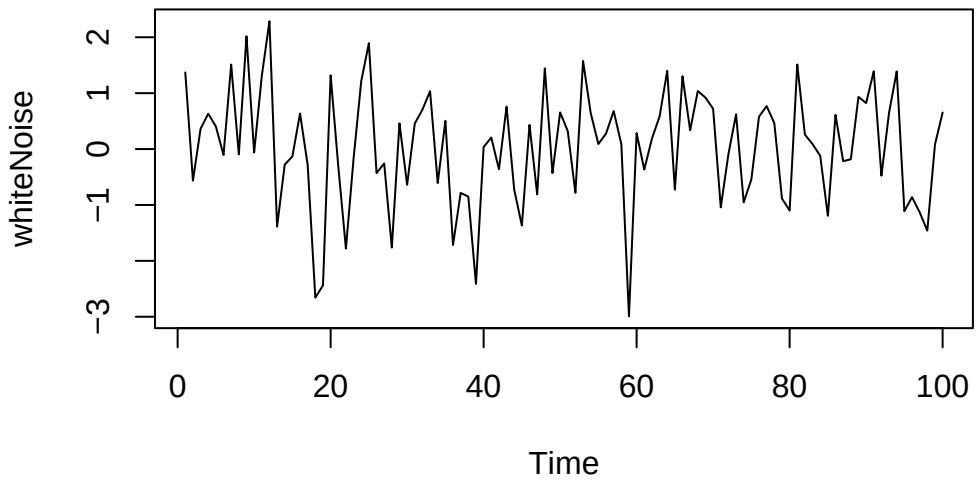
Fundamentals of Modeling

White noise

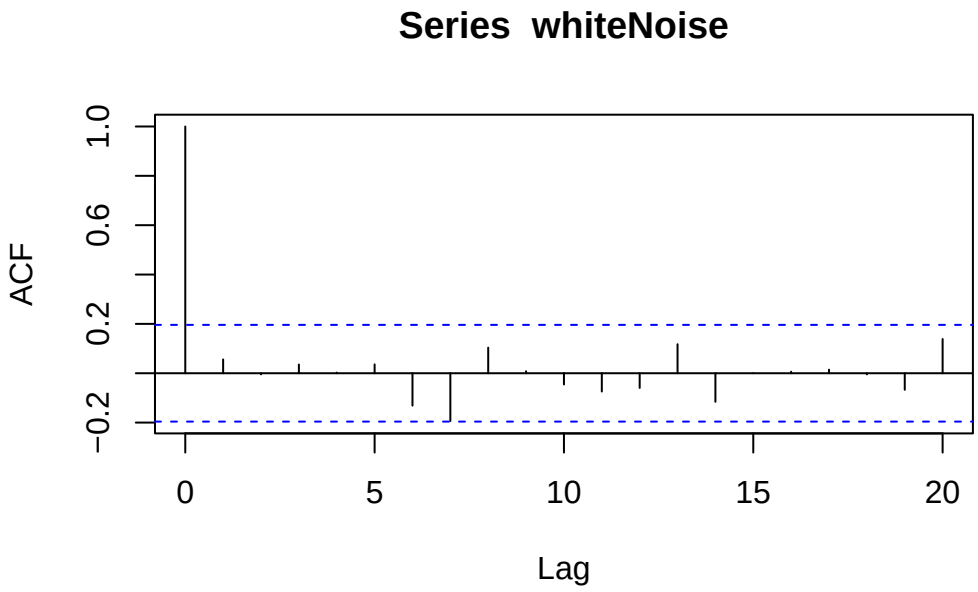
A time series process $Wt, t \in \mathbb{N}$ is called white noise if it is a sequence of independent and identically distributed random variables with mean 0.

> Note: The name "white" indicates that all frequencies are included, as in the case of "white light."

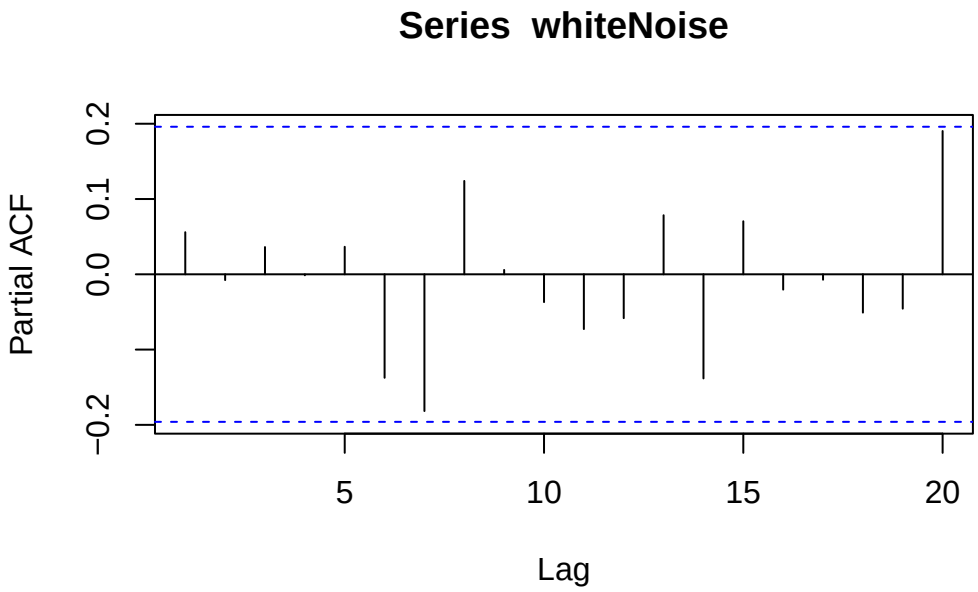
```
set.seed(42)
whiteNoise <- ts(rnorm(100,mean=0,sd=1))
plot(whiteNoise)
```




```
acf(whiteNoise)
```



```
pacf(whiteNoise)
```



Causal White Noise

A series of *iid* random variables e_t, which are individually independent of the proceeding time series elements X_t

Basic Time Series Models

The following models are often used when there is limited information about how time series data is generated:

- AR: auto-regressive model
- MA: moving average model
- ARMA: combination of AR and MA model
- ARIMA: non-stationary ARMA model (I=integration)
- SARIMA: seasonal ARIMA model

Auto-Regression Models (AR)

Express the current random variable as a linear combination of the last p time steps and a "completely independent" term e_t called innovation.

$$X_t = \alpha_1 X_{t-1} + \cdots + \alpha_p X_{t-p} + e_t$$

where α_x is a set of finite parameters and e_t a series of independent random variables.

$AR(1)$ as Random Walk Model

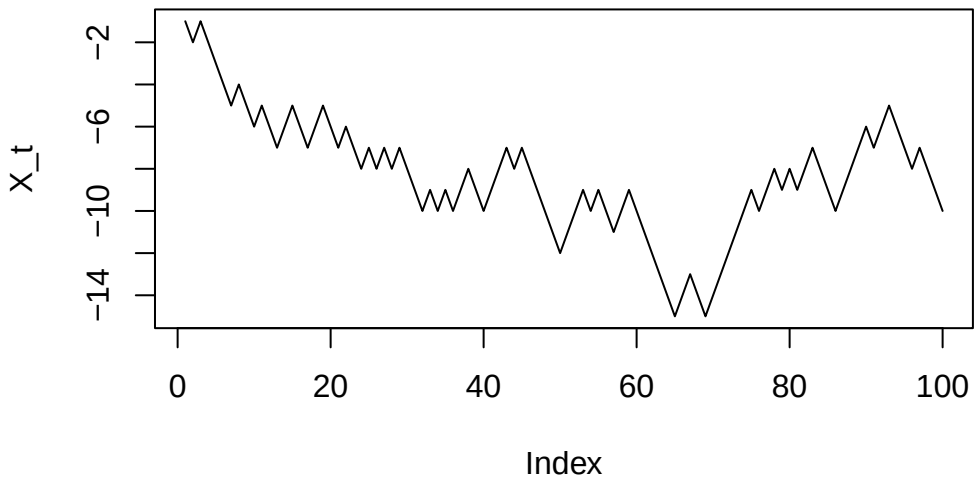
Trajectory of an object moving along a line. At each moment, the object takes randomly the decision in which direction to move on.

$$X_t = X_{t-1} + e_t$$

where e_t is random with equal probability for -1 and 1.

```
set.seed(42)
step<- 1 - 2 * round(runif(100))
X_t<-cumsum(step)
plot(X_t, type="lines")
```

```
Warning in plot.xy(xy, type, ...): plot type 'lines' will be truncated to first
character
```

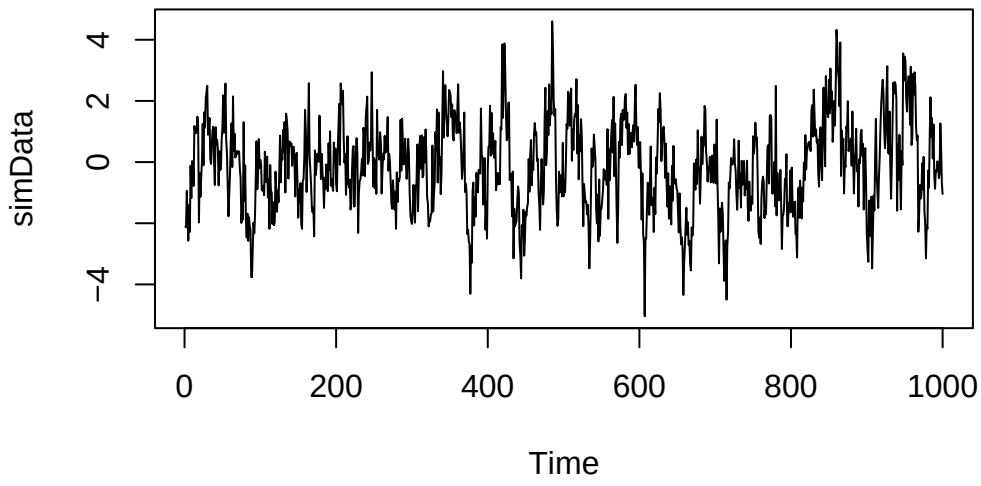


Simulation $AR(p)$ Processes

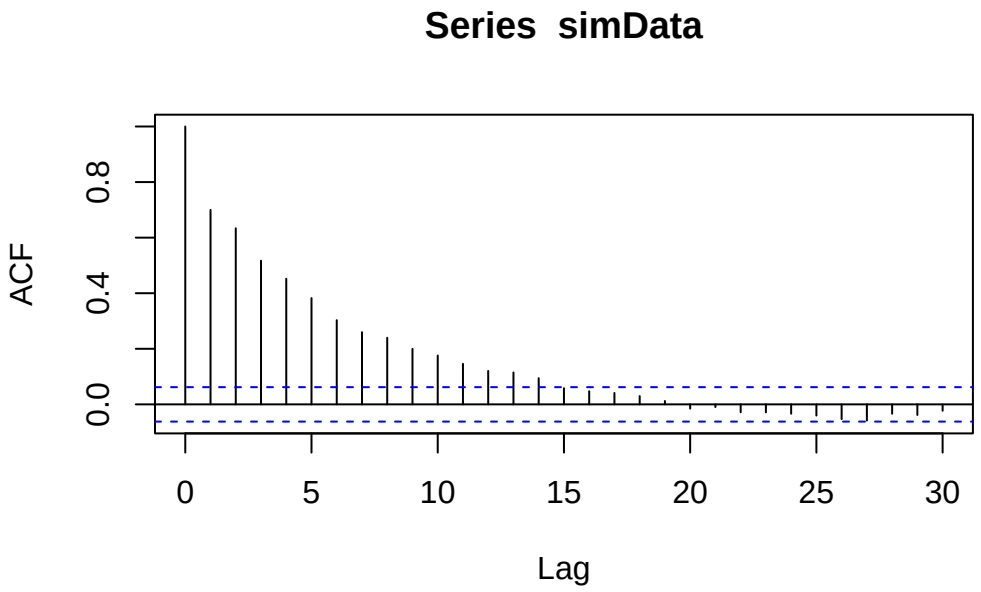
We would like to simulate:

$$X_t = 0.5X_{t-1} + 0.3X_{t-2} + e_t \text{ with } e_t \sim N(0,1) \text{ iid.}$$

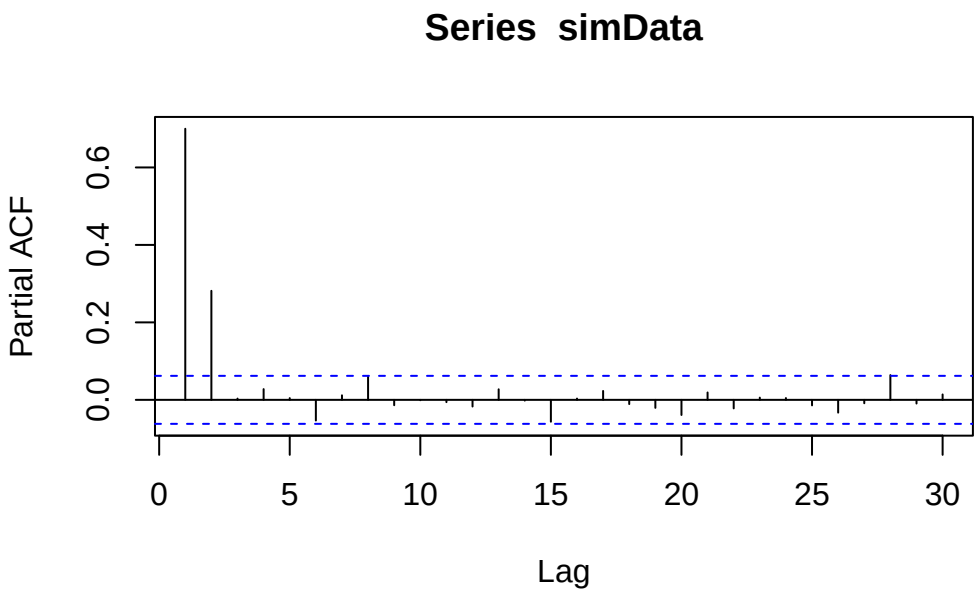
```
set.seed(42)
ar.coefficients <- c(0.5, 0.3)
simData <- arima.sim(n=1000,model=list(ar=ar.coefficients))
plot(simData)
```


simData
4
2
0
−4
0
200
400
600
800
1000
Time

```
acf(simData)
```


Series simData
ACF
0.8
0.4
0.0
0
5
10
15
20
25
30
Lag

```
pacf(simData)
```



Stationarity of $AR(p)$ Models

AR(p) models are stationary when the:

- mean is 0
- absolute value of the roots of the characteristic polynomial is all larger than 1

```
polyroot(c(1,-0.8,-0.4))
```

```
[1]  0.8708287+0i -2.8708287+0i
```

As the first value has an absolute value below 1, the time series is not stationary.

Key Properties of $AR(p)$ Models

The AR(1) process:

$$X_t = 0.75 * X_t + e_t$$

is stationary because the characteristic polynomial:

```
polyroot(c(1,-0.75))
```

```
[1] 1.333333+0i
```

For the model identification of AR(p) processes, two properties are key:

- Autocorrelation decays quickly or displays a damped sinusoid
- Partial autocorrelation function has a clear cut-off at p

Model Order Identification

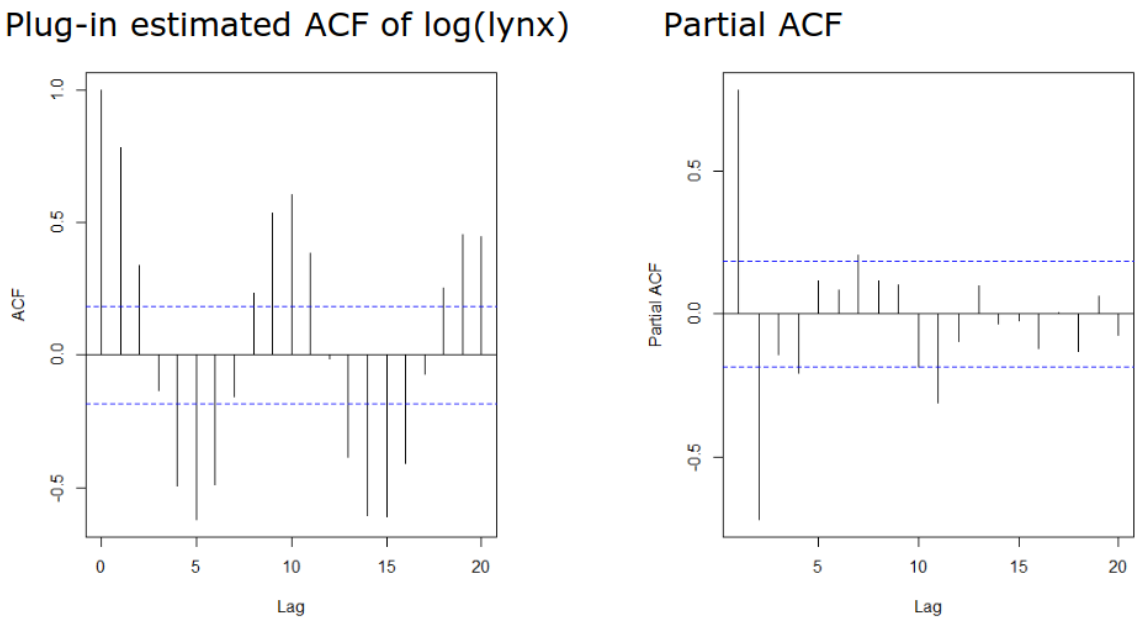


Figure 9: Plug-in estimated ACF and PACF

- ACF displays damped sinusoid
- PACF has clear cut-off after lag 11, 7, 4 or 2 (depending on the threshold).

Try fitting an $AR(2)$, $AR(4)$, $AR(7)$, $AR(11)$

W-05-Timeseries-AR-MA

Model Parameter Identification

Time series are often not centered: I.e. instead of a classical AR(p) process, a centered version has to be fitted.

$$Y_t = m + X_t = m + \alpha_1 X_{t-1} + \cdots + \alpha_p X_{t-p} + e_t$$

Fitting parameters are: - the global mean m - the AR-coefficients $\alpha_1, \ldots, \alpha_p$ - the parameters defining the distribution of the innovation e_t. Here, we assume a normal distribution.

Model Parameter from Ordinary Least Square (OLS)

We want to estimate the parameters from the model equation, we have a total od $n-p$ points for this, because we have a total of n time segments and each equation links p time instance.

The OLS procedure is:

1. Estimate the global mean $\hat{m} = \bar{y} = \frac{1}{n}\sum_{t=1}^{n} y_t$ and determine $x_t = y_t - \hat{m}$
2. Carry out a regression on x_t without intercept. The resulting parameters are $\hat{\alpha}, \ldots, \hat{\alpha_p}$
3. Estimate the standard deviation of the innovation $\hat{\sigma}_E^2$ deviation of the residual.

```
ar.ols(data,order=p)
```

Yule-Walker Equations

Basic idea: The model coefficients and the ACF values are entangled. The values of the auto-correlation function depend on the model coefficients. A careful calculation for an AR(p) model yields the following equations, called Yule-Walker equations:

$$\sum_{i=1}^{p} \alpha_i \rho(k-i) = \rho(k) \quad \text{for } k > 0$$

$$\sum_{i=1}^{p} \alpha_i \rho(i) = \rho(0) - \sigma_E^2 \quad \text{for } k = 0$$

Note: Note that the autocorrelation function is symmetric, i.e. $p(-k) = p(k)$ for all k.

To fit the AR(p) model parameters, we must solve the above equations with the estimated values of the auto correlation function.

```
ar.yw(data,order=p)
```

Further options for fitting

Burg's algorithm

Offers another fitting cost function that exploit also the first p function values.

```
ar.burg(data,order.max=p)
```

Maximum likelihood estimator (MLE)

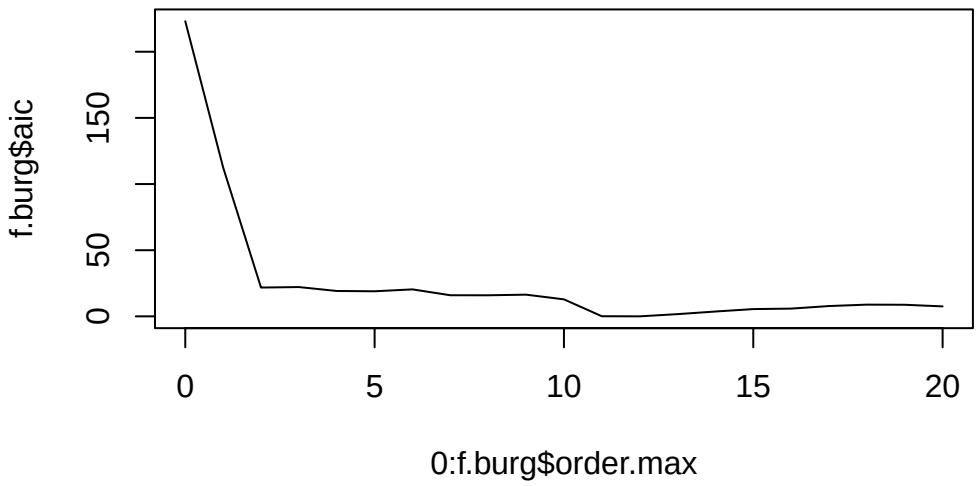
Determines the parameter based on an optimization of the maximum likelihood function

```
arima(Data,order=c(2,0,0))
```

Which method to use when?

One option is to use the Akaike information criterion (AIC), which estimates the relative amount of information lost by the considered model. The less information a model loses, the higher the quality of that model.

```
f.burg<-ar.burg(log(lynx))
plot(
    0:f.burg$order.max,
    f.burg$aic,
    type="l"
    )
```

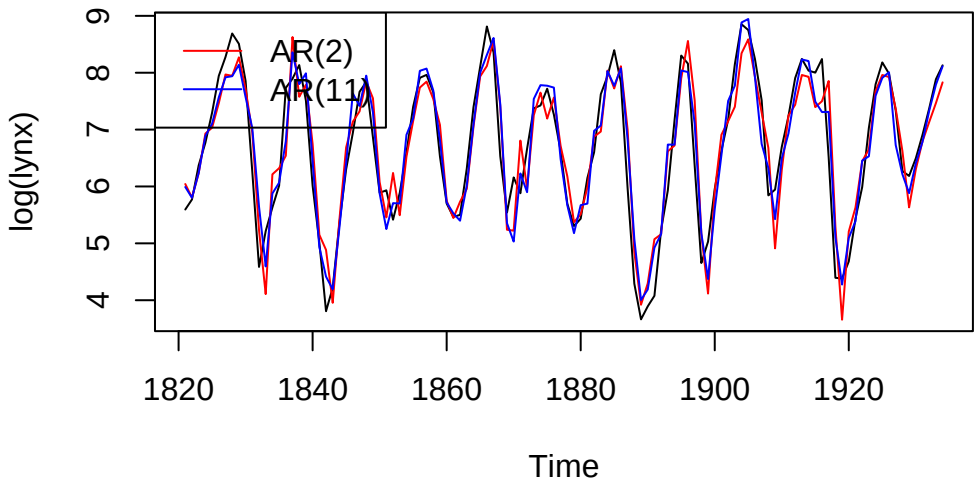


Note: No order parameter is given any more.

Model Diagnostic

Example

```
# Fitting the two models
fit.2<-arima(log(lynx),order=c(2,0,0))
fit.11<-arima(log(lynx),order=c(11,0,0))

# Display the time series and model predictions
plot(log(lynx))
lines(log(lynx) - fit.2$resid, col = "red")
lines(log(lynx) - fit.11$resid, col = "blue")
legend("topleft",
       legend = c("AR(2)", "AR(11)"),
       col = c("red", "blue"),
       lty = 1)
```

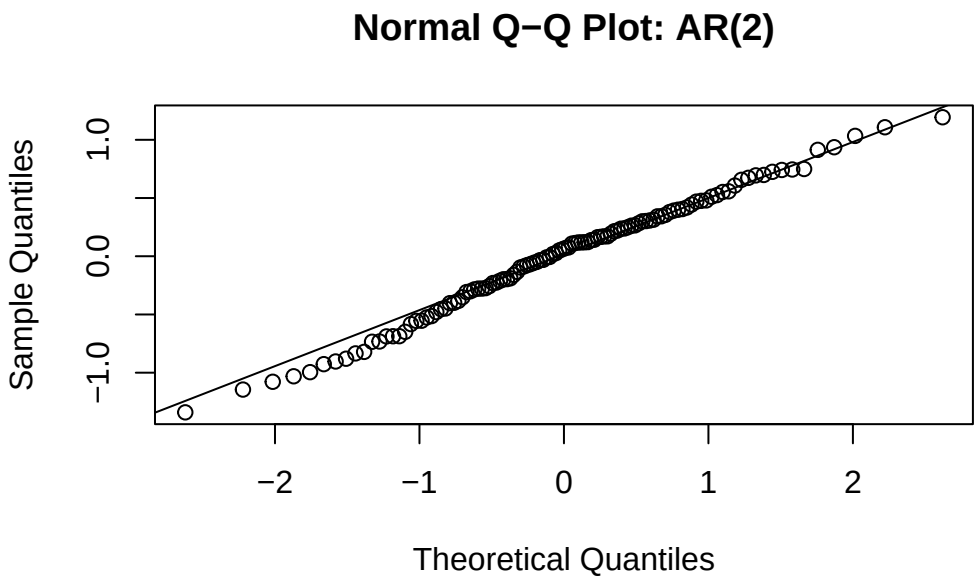


QQ-Plot

Compare quantile of measurement data with theoretical distribution

- X-Axis: Theoretical distribution
- Y-Axis: Actual data distribution

```
qqnorm(fit.2$residuals,main="Normal Q-Q Plot: AR(2)")
qqline(fit.2$residuals)
```



Moving Average Models (MA)

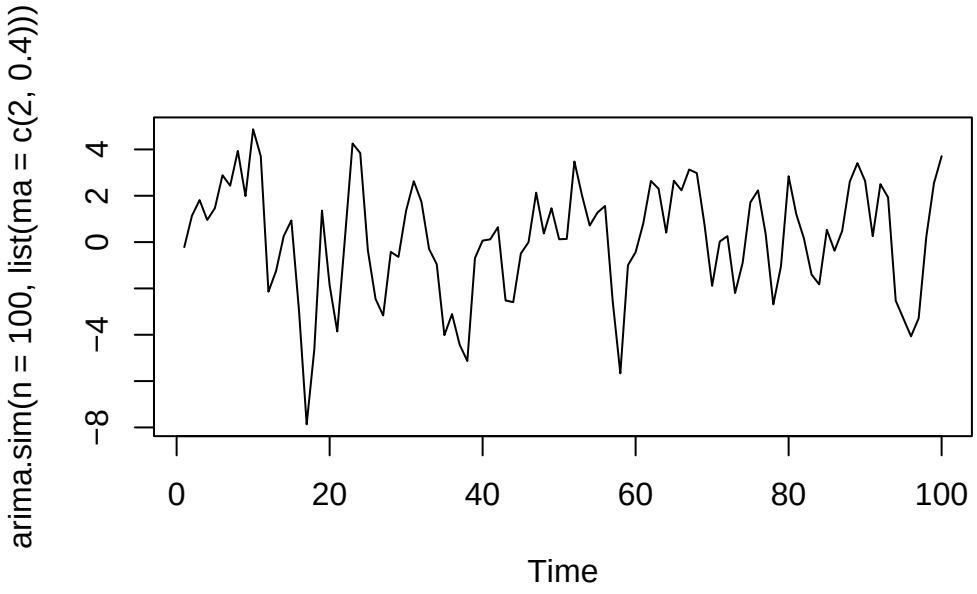
A moving average MA(q) model is a model in the form

$$X_t = E_t + \beta_1 E_{t-1} + \beta_2 E_{t-2} + \cdots + \beta_q E_{t-q}$$

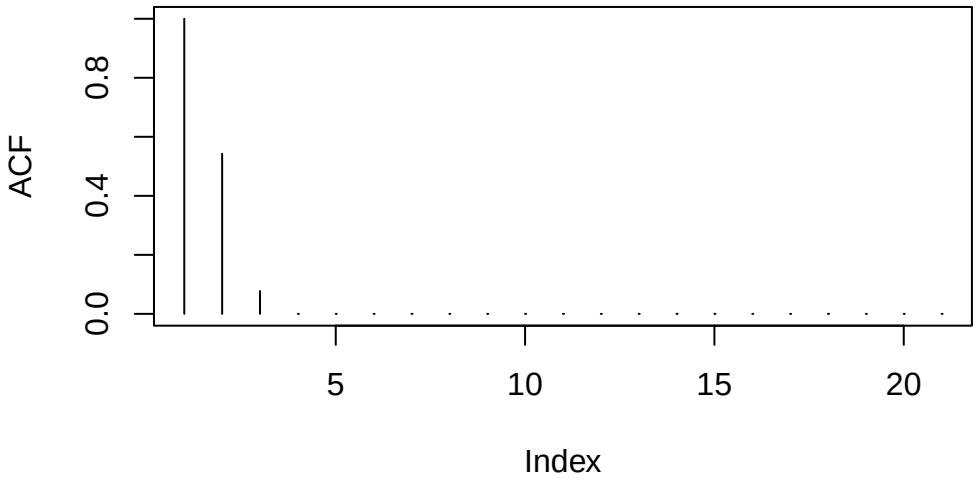
with the time series E_t of innovations.

> **Note**
>
> A moving average model does not have an explicit reference to previous values; only the innovation parts are considered.

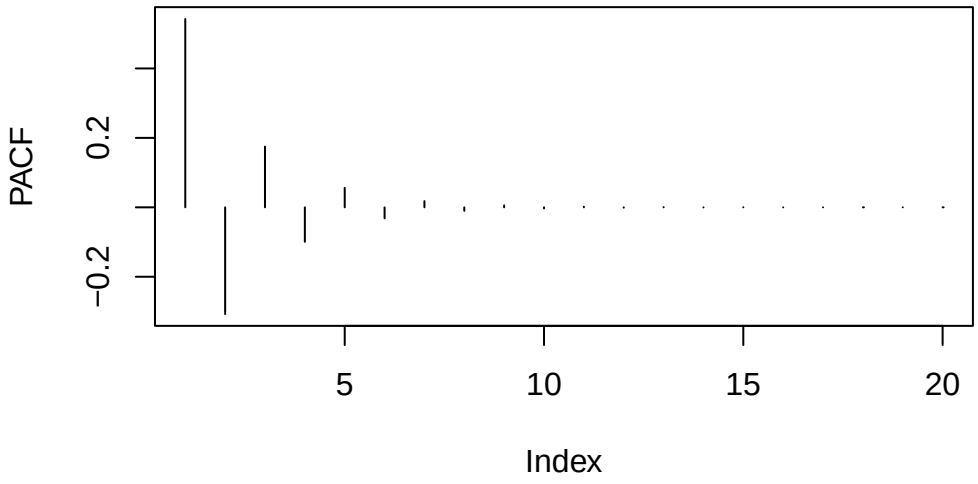
```
set.seed(42)

#MA(2)
plot(arima.sim(n=100,list(ma=c(2,0.4))))
```




```
plot(ARMAacf(ma=c(2,0.4),lag.max=20),type="h",ylab="ACF")
```



```
plot(ARMAacf(ma=c(2,0.4),pacf=T,lag.max=20),type="h",ylab="PACF")
```



Generalization: In general, we can show that the autocorrelation function of an $MA(q)$ vanishes for all lags larger than q.