Métodos de Previsão
e
Séries Temporais
(Parte II)

Mestrado em Estatística para Ciência de Dados

A. Manuela Gonçalves
mneves@math.uminho.pt

Departamento de Matemática
http://www.math.uminho.pt

Part II – Exploratory Analysis

1. Introduction

- 1.1. Time series components: trend, seasonality and remaining
- 1.2. Decomposition of a time series

2. Variance-stabilizing transformations

3. Dealing with trend (estimation/elimination)

- 3.1. Curve fitting
- 3.2. Smoothing (finite order MA filter, exponential smoothing, kernel smoothing)
- 3.3. Differencing

4. Dealing with seasonality

- 4.1. Averaging for the period
- 4.2. Seasonal differencing
- 4.3. Harmonic regression

5. Dealing with trend and seasonality using Loess: STL approach

6. Testing the noise sequence

7. Final notes and references

1. Introduction – Time series components

Times series components

Time series data can exhibit a variety of components, some of them deterministic, that can (and should) be removed before further modelling.

Classical methods consider that the observed time series is a realization of the process

$$X_t = f(m_t, s_t, Y_t)$$

comprising three components:

- m_t is a tendency aka **trend component**;
- s_t is a function with period d referred to as a **seasonal component** (day, month, …);
- Y_t is a (stationary) **random noise/error component**.

The time series X_t and its components m_t, s_t, Y_t change with time through the index t. While m_t and s_t are considered deterministic components, Y_t is random.

1. Introduction – Time series components

Times series components

$$X_t = f(m_t, s_t, Y_t)$$

- Remove the deterministic components m_t and s_t from X_t;
- The remaining corresponds to the noise/error Y_t assumed to be a zero-mean stationary process;
- If Y_t exhibits autocorrelation, a probabilistic model can be used to further analyze its properties (e.g. from the SARIMA family of models).

Dealing with trend and seasonality (2 approaches)

1) Estimate explicitly m_t and s_t;
2) Apply differencing operators repeatedly to X_t until the differenced observations resemble a realization of a stationary time series.

1. Introduction - Decomposition of a time series

Additive versus multiplicative decompositions

If assuming an additive decomposition, then

$$X_t = m_t + s_t + Y_t \Leftrightarrow Y_t = X_t - m_t - s_t$$

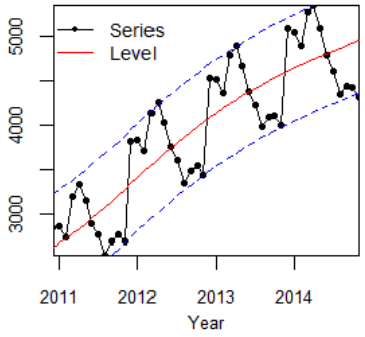
while, alternatively, a multiplicative decomposition would be written as

$$X_t = m_t \times s_t \times Y_t \Leftrightarrow Y_t = X_t/(m_t \times s_t)$$

If the magnitude of the seasonal fluctuations, or the variation around the trend, seem to vary with the level of the time series, then a multiplicative decomposition is more appropriate. Otherwise, the additive decomposition is appropriate.

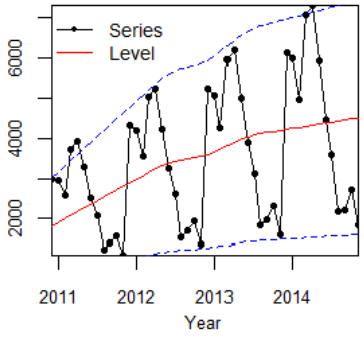
Additive model

$X_t = m_t + s_t + Y_t$



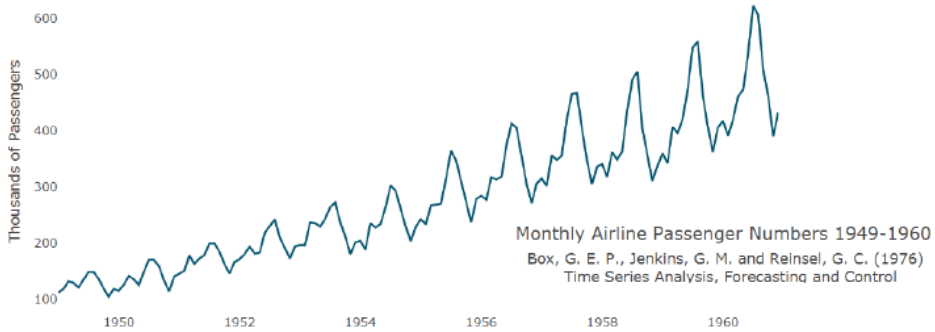
Multiplicative model

$X_t = m_t \times s_t \times Y_t$

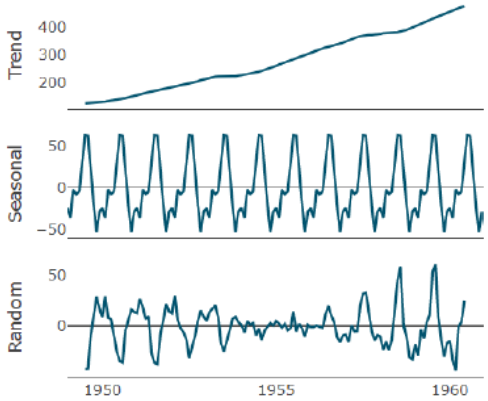


1. Introduction - Decomposition of a time series

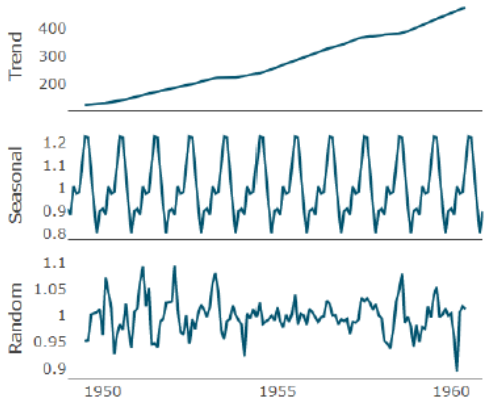
Additive versus multiplicative decompositions



Additive: $X_t = m_t + s_t + Y_t$



Multiplicative: $X_t = m_t \times s_t \times Y_t$

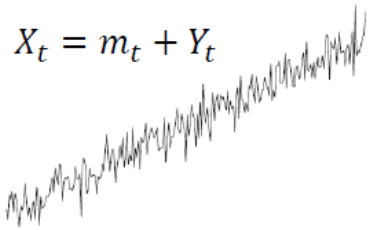


1. Introduction - Decomposition of a time series

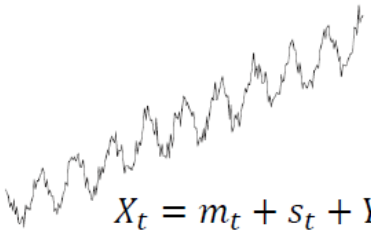
Additive versus multiplicative decompositions

Other decompositions can also be considered (mixed models)

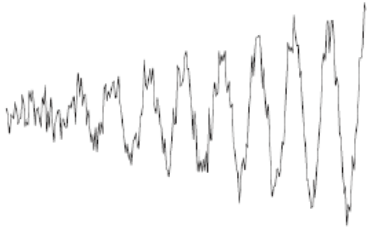
$X_t = m_t + Y_t$



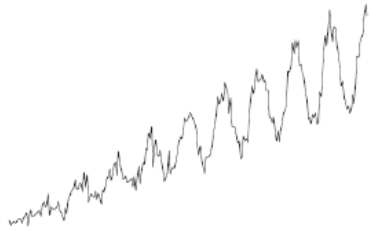
$X_t = m_t + s_t + Y_t$



$X_t = m_t \times s_t + Y_t$



$X_t = m_t + (m_t \times s_t) + Y_t$
$= m_t \times (1 + s_t) + Y_t$



A different approach with additive decomposition consists of

- applying a variance-stabilizing transformation on the original data (i.e. transform the data until the series variation appears to be stable over time),
- and then use an additive decomposition.

E.g. the additive decomposition on a log transformation of the original data is equivalent to the log of a multiplicative decomposition on the original data as

$$\log X_t = \log(m_t) + \log(s_t) + \log(Y_t) = \log(m_t \times s_t \times Y_t)$$

2. Variance - stabilization

Variance-stabilizing transformations

A transformation $Z_t = f(X_t)$ can be useful if X_t shows variation that increases or decreases with the level of the series.

But which kind of transformation?

In general, logarithms are useful because they are interpretable: changes in a log value are relative (or percentage) changes on the original scale.

E.g., for $Z_t = \log_{10} X_t$ then an increase of 1 in Z_t corresponds to a multiplication of 10 in X_t.

Other transformations are also used, although not being so interpretable. Square roots and cube roots can be used as examples of general power transformations as

$$Z_t = (X_t)^p$$

with $p = 1/2$ for the square and $p = 1/3$ for the cubic root.

2. Variance - stabilization

Variance-stabilizing transformations

The family of Box-Cox transformations (Box and Cox, 1964) includes both logarithms and power transformations. It is defined as

$$Z_t = \begin{cases} \dfrac{X_t^{\lambda} - 1}{\lambda}, \lambda \neq 0 \\ \ln(X_t), \lambda = 0 \end{cases}$$

where λ is the parameter of the function:

- power transformation and scaling ($\lambda \neq 0$)
- or natural logarithm ($\lambda = 0$).

λ	Transformation	
1	$z_t = x_t$	–
1/2	$z_t = \sqrt{x_t}$	$x_t > 0$
1/3	$z_t = \sqrt[3]{x_t}$	$x_t > 0$
1/4	$z_t = \sqrt[4]{x_t}$	$x_t > 0$
0	$z_t = \ln x_t$	$x_t > 0$
−1/2	$z_t = (\sqrt{x_t})^{-1}$	$x_t > 0$
−1	$z_t = (x_t)^{-1}$	$x_t \neq 0$

2. Variance - stabilization

Variance-stabilizing transformations

The Box-Cox modification (Bickel and Doksum, 1981) allows for non-positive X_t

$$Z_t = \begin{cases} \text{sign}(X_t)\dfrac{|X_t|^{\lambda}-1}{\lambda}, \lambda \neq 0 \\ \\ \ln(X_t), \lambda = 0 \end{cases}$$

and it is the same as the original Box-Cox transformation for positive X_t.

After choosing a proper λ value, Z_t can be used to build a model and forecast. Then, it maybe useful to reverse the Box-Cox transformation (or *back-transform*) to obtain the model or forecasts on X_t. The reverse Box-Cox transformation is given by

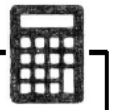
$$X_t = \begin{cases} \text{sign}(\lambda\, Z_t + 1)|\lambda\, Z_t + 1|^{1/\lambda}, \lambda \neq 0 \\ \\ \exp(Z_t), \lambda = 0 \end{cases}$$

2. Variance - stabilization

Variance-stabilizing transformations

An optimum value for λ is one which makes the size of the seasonal variation about the same across the time series. As examples, the value of λ can be chosen to

- minimize the coefficient of variation for subseries of X_t (Guerrero, 1993),
- maximize the profile log likelihood of a linear model fitted to X_t,
- other methods …



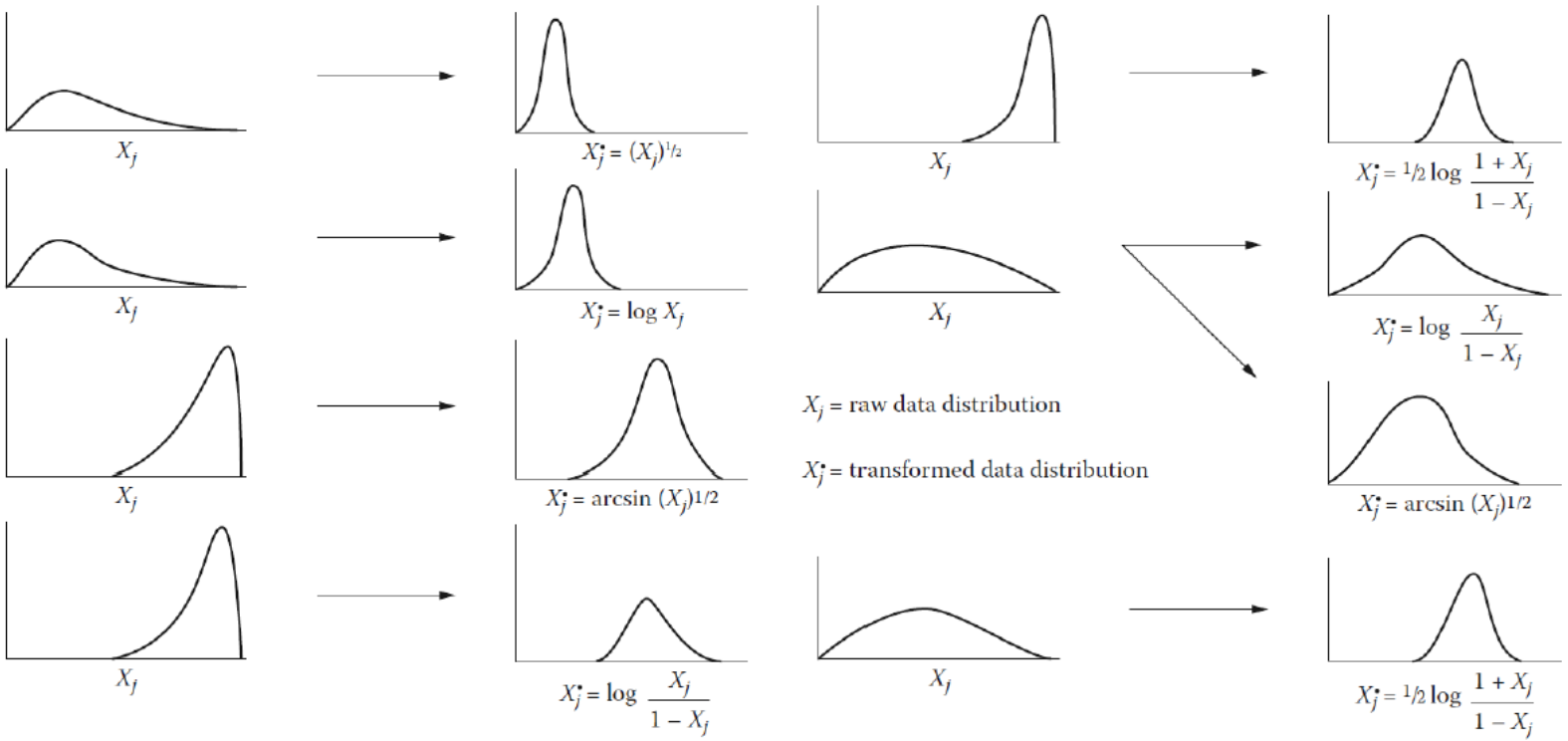
```
install.packages("forecast")
library(forecast)
seed = 123
x = rchisq(500, df=7)
z = forecast::BoxCox(x,lambda = 0.05) # boxcox transformation with lambda = 0.05
x_reversed = forecast::InvBoxCox(z,lambda = 0.05) # boxcox back-transformation with lambda = 0.05

# get optimum lambda where method = c("guerrero", "loglik")
lambda_opt = forecast::BoxCox.lambda(x,lower=-1,method = "loglik")
z_opt = forecast::BoxCox(x,lambda = lambda_opt)
par(mfrow=c(1,3)); hist(x, ylim=c(0,150)); hist(z, ylim=c(0,150)); hist(z_opt, ylim=c(0,150))
```

2. Variance - stabilization

Variance-stabilizing transformations

Common Box-Cox transformations to reduce asymmetry or kurtosis.



Adapted from Rummel RJ (1970), Applied Factor Analysis. Evanston: Northwestern University Press.

3. Trend

Dealing with trend (estimation and elimination)

In the absence of a seasonal component, the additive model becomes

$$X_t = m_t + Y_t, t = 1,2, \dots, n$$

where the trend component m_t can be

- estimated by curve fitting or by a smoothing approach,

Estimate the trend, subtract it from the original data and build an appropriate stationary time series model for the remaining part (residuals).

- removed by differentiation of order d.

Eliminate the trend by differencing and build an appropriate stationary model for the differenced series.

The latter method has the advantage of

- usually requiring the estimation of fewer parameters and
- does not rest on the assumption of a trend that remains fixed throughout the observation period.

3. Trend

Dealing with trend (estimation and elimination)

In the absence of a seasonal component, the additive model becomes

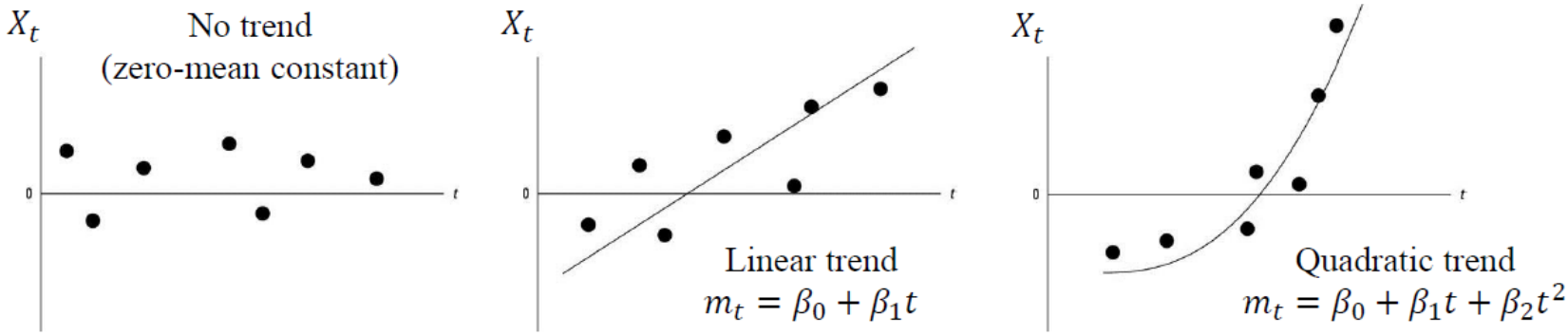
$$X_t = m_t + Y_t, t = 1,2, \dots, n$$

where the trend component m_t can be

- estimated by curve fitting (least squares)
  - linear trend $m_t = \beta_0 + \beta_1 t$,
  - quadratic trend $m_t = \beta_0 + \beta_1 t + \beta_2 t^2$, …, k-order polynomial trend,
  - concave/convex curves (no inflection point, e.g. exponential, logarithmic), sigmoidal curves, …
- estimated by a smoothing approach (non parametric)
  - linear filter e.g. finite order moving average filter,
  - exponential smoothing,
  - splines, local regression, LOESS (locally weighted polynomial regression), …
- removed by differencing the original data (differentiation of order d).

3. Trend – curve fitting

Dealing with trend via curve fitting



Considering the model $X_t = m_t + Y_t, t = 1,2, \ldots, n$ and a trend m_t, the estimators (and estimates) for the parameters of the trend model can be obtained as those minimizing the sum of squared errors (SSE), i.e.

$$\sum_{t=1}^{n} \hat{y}_t^2 = \sum_{t=1}^{n} (x_t - m_t)^2 ,$$

where x_t is the observed path of X_t and $\hat{y}_t$ is the estimate of the observed path of Y_t.

3. Trend – curve fitting

Dealing with trend via curve fitting

As an example, the linear trend

$$m_t = \beta_0 + \beta_1 t, \ t = 1,2, \dots, n$$

can be fitted to the $\{x_1, x_2, \dots, x_n\}$ data by estimating β_0 and β_1 as to minimize

$$\sum_{t=1}^{n} \hat{y}_t{}^2 = \sum_{t=1}^{n} (x_t - m_t)^2 = \sum_{t=1}^{n} (x_t - \beta_0 - \beta_1 t)^2$$

with respect to β_0 and β_1. This method is the least squares regression and can be used to estimate higher-order polynomials and other curves. The detrended series is obtained by

$$\hat{y}_t = x_t - (b_0 + b_1 t)$$

where b_0 and b_1 are the estimates for β_0 and β_1, respectively.

stats::lm (here) → linear models estimated by ordinary least-squares,
stats::glm (here) → generalized linear models i.e. lm with correlated or heteroscedastic errors,
stats::nls (here) → non linear models by least-squares.

3. Trend – Smoothing (MA, exponential, kernel)

Dealing with trend via smoothing

Smoothing

- is a data-driven operation (non-parametric),
- smooths out the irregular roughness of the original data,
- can be used to estimate trend or seasonality.

Smoothing does not provide a statistical model, but it can be a good approach to estimate various components of the time series.

The term filter is also used to describe a smoothing procedure. E.g. if the smoothed process S_t is calculated as a linear combination of X for surrounding times t as

$$S_t = \frac{1}{2q+1} \sum_{j=-q}^{q} X_{t-j}$$

this consists of a linear filter applied to the original data (different from curve fitting…).

3. Trend – Smoothing (MA, exponential, kernel)

Dealing with trend via smoothing (finite order moving average filter)

Consider $X_t = m_t + Y_t$, $t = 1,2, \dots, n$ and the two-sided moving average

$$W_t = \frac{1}{2q+1} \sum_{j=-q}^{q} X_{t-j}$$

with $q \in \mathbb{N}$. Then, for $q + 1 \leq t \leq n - q$

$$W_t = \frac{1}{2q+1} \sum_{j=-q}^{q} (m_{t-j} + Y_{t-j}) = \frac{1}{2q+1} \sum_{j=-q}^{q} m_{t-j} + \frac{1}{2q+1} \sum_{j=-q}^{q} Y_{t-j} \approx m_t$$

assuming that, over the $[t-q, t+q]$ interval, m_t is approximately linear and the average of the error terms is close to zero. Thus,

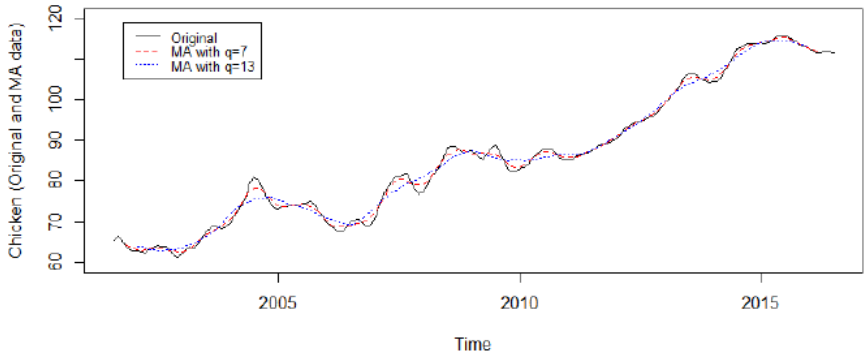
$$\widehat{m}_t = \frac{1}{2q+1} \sum_{j=-q}^{q} X_{t-j}$$

As X_t is not observed for $t \leq 0$ or $t > n$, then $\widehat{m}_t$ is not defined for $t \leq q$ or $t > n - q$.

3. Trend – Smoothing (MA, exponential, kernel)

Dealing with trend via smoothing (finite order moving average filter)

Remark: This approach is defined for odd-length windows $(2q + 1)$ but can be adjusted for even-length by adding only $1/2$ of the 2 most extreme lags so that W_t and X_t line up. This keeps the window symmetric.




```
library(zoo)
library(astsa)
MA7 = zoo::rollmean(chicken, k=7); # equivalent to MA7 = filter(chicken, sides=2, filter=rep(1/7,7))
MA13 = zoo::rollmean(chicken, k=13)
# Plot Multiple Time Series in the same graph
ts.plot(chicken, MA7, MA13,
      gpars=list(ylab="Chicken (Original and MA data)", ylim = c(60,120), col=c("black", "red", "blue"), lty=c(1:3)))
legend("topleft", inset = 0.05, legend=c("Original", "MA with q=7", "MA with q=13"),
     col=c("black", "red", "blue"), lty=1:3, cex=0.8)
```

3. Trend – Smoothing (MA, exponential, kernel)

Dealing with trend via smoothing (exponential smoothing)

When estimating a trend, it seems reasonable to weight recent observations more than observations that are less recent.

For $\alpha \in [0,1]$, the one side moving averages $\hat{m}_t$, $t = 1,2,\ldots,n$ defined by the recursion

$$\hat{m}_t = \alpha X_t + (1-\alpha)\hat{m}_{t-1} \quad \text{and} \quad \hat{m}_1 = X_1$$

can be used for trend estimate. This approach is often referred as *exponential smoothing*, since the above recursion implies that

$$\hat{m}_t = \sum_{i=0}^{t-2} \alpha(1-\alpha)^i X_{t-i} + (1-\alpha)^{t-1} X_1$$

which constitutes a weighted moving average of X_t, X_{t-1}, … with geometric weights $w_i = \alpha(1-\alpha)^i$ (except for X_1). These weights decrease exponentially with i and thus, recent observations have a higher weight than less recent observations in the computation of $\hat{m}_t$, $t = 1,2,\ldots,n$.

3. Trend – Smoothing (MA, exponential, kernel)

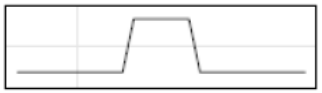
Dealing with trend via smoothing (kernel smoothing)

Other weighting schemes can also be used. Kernel smoothing is a moving average smoother that uses a weight function (or kernel), to average the observations as

$$\hat{m}_t = \sum_{i=1}^{n} w_i(t) X_i \quad \text{with} \quad w_i(t) = \frac{K\left(\frac{t-i}{b}\right)}{\sum_{j=1}^{n} K\left(\frac{t-j}{b}\right)}$$

where K is the kernel function and b is the bandwidth (increasing b leads to smoother the result). Several kernels can be used with this framework, namely

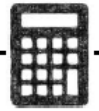
Boxcar



$K(z) = (c - d) f(z|c, d)$
where $f(z|c, d)$ is the pdf of the uniform in the $[c, d]$ interval

Normal

$$K(z) = \frac{1}{\sqrt{2\,\pi}} e^{-z^2/2}$$



The stats::ksmooth implements this procedure in R (see ksmooth help).
The kernels are scaled so that their quartiles are at $\pm 0.25 \times b$. For the standard normal distribution, the quartiles are ± 0.674 and, for a monthly time series, $b = 1$ approximately smooths a little over one year.

3. Trend – Differencing

Dealing with trend via differencing

Instead of estimating the trend (by curve fitting or smoothing) and removing it from the original series, one can use differencing to eliminate the trend component.

If the appropriate model is $X_t = m_t + Y_t$ then differencing X_t leads to

$$X_t - X_{t-1} = (m_t + Y_t) - (m_{t-1} + Y_{t-1}) = m_t - m_{t-1} + Y_t - Y_{t-1}.$$

Considering $m_t = \delta + m_{t-1} + e_t$ (random walk) then $m_t - m_{t-1} = \delta + e_t$ and

$$X_t - X_{t-1} = \delta + e_t + Y_t - Y_{t-1},$$

where e_t is white noise and is independent of Y_t.

Given that Y_t is a stationary process then the process $Y_t - Y_{t-1}$ is also stationary as

$$\text{Cov}(Y_t - Y_{t-1}, Y_{t+h} - Y_{t-1+h}) = 2\gamma_Y(h) - \gamma_Y(h+1) - \gamma_Y(h-1)$$

where $\gamma_Y(h)$ is the ACF function of Y_t. Thus, $X_t - X_{t-1}$ is also stationary.

Therefore, differencing X_t yields a stationary process.

3. Trend – Differencing

Dealing with trend via differencing

Differencing is also suitable if the trend is considered fixed i.e. $m_t = \beta_0 + \beta_1 t$. In this case, it follows that

$$X_t - X_{t-1} = \beta_1 + Y_t - Y_{t-1}$$

which produces the stationary process $X_t - X_{t-1}$.

Because differencing plays a central role in time series analysis, it receives its own notation. The first difference operator is

$$\nabla X_t = X_t - X_{t-1} = (1 - B)X_t$$

where B is the backward shift operator defined as $BX_t = X_{t-1}$. Furthermore, the differences of higher order can be defined from the difference operator of order d

$$\nabla^d = (1 - B)^d$$

being expanded algebraically to evaluate for higher integer values of d, similarly as polynomial functions of real variables.

3. Trend – Differencing

Dealing with trend via differencing

For example, the second order difference becomes

$$\nabla^2(X_t) = (1-B)^2 X_t = (1-2B+B^2)X_t = X_t - 2X_{t-1} + X_{t-2}$$

by the linearity of the operator B. Also,

$$\nabla^2(X_t) = (X_t - X_{t-1}) - (X_{t-1} - X_{t-2}) = \nabla(X_t - X_{t-1}) = \nabla\big(\nabla(X_t)\big)$$

i.e. the second order difference is equivalent to the difference of the first difference.

Remark: If ∇ is applied to a linear function, the result is a constant. In the same way, any polynomial of order k can be reduced to a constant by applying ∇^k.

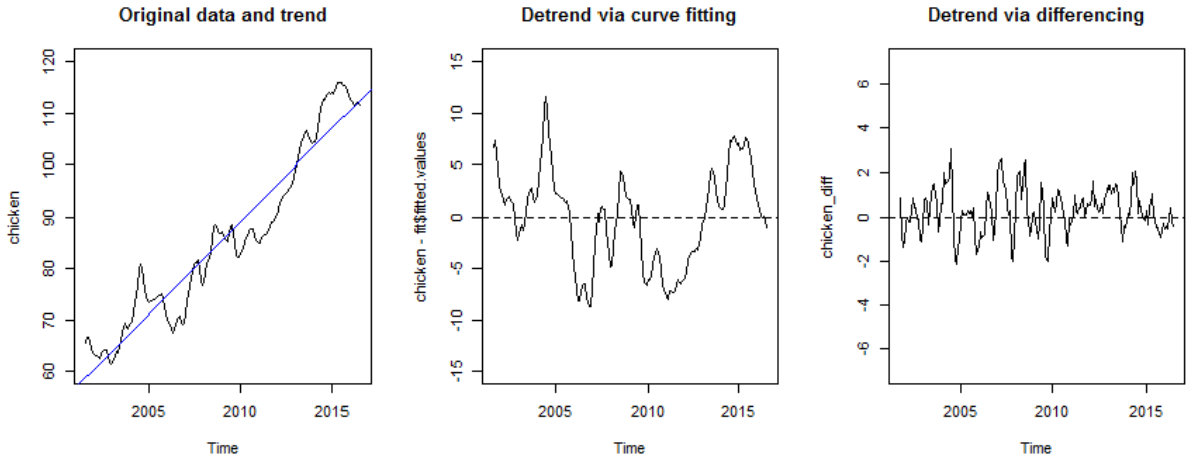
Notes:

- This suggests the possibility, given any sequence $\{x_t\}$ of data, of applying the ∇ operator repeatedly until the sequence $\{\nabla^k(x_t)\}$ can be plausibly modeled as a realization of a stationary process.
- The order of differencing required in practice is usually quite small ($k \leq 2$). This relies on the fact that many functions can be well approximated, on an interval of finite length, by a low degree polynomial.

3. Trend – Differencing

Dealing with trend via differencing

Differencing over curve fitting has the advantage of (usually) requiring the estimation of fewer parameters and not assuming a fixed trend throughout the observation period.

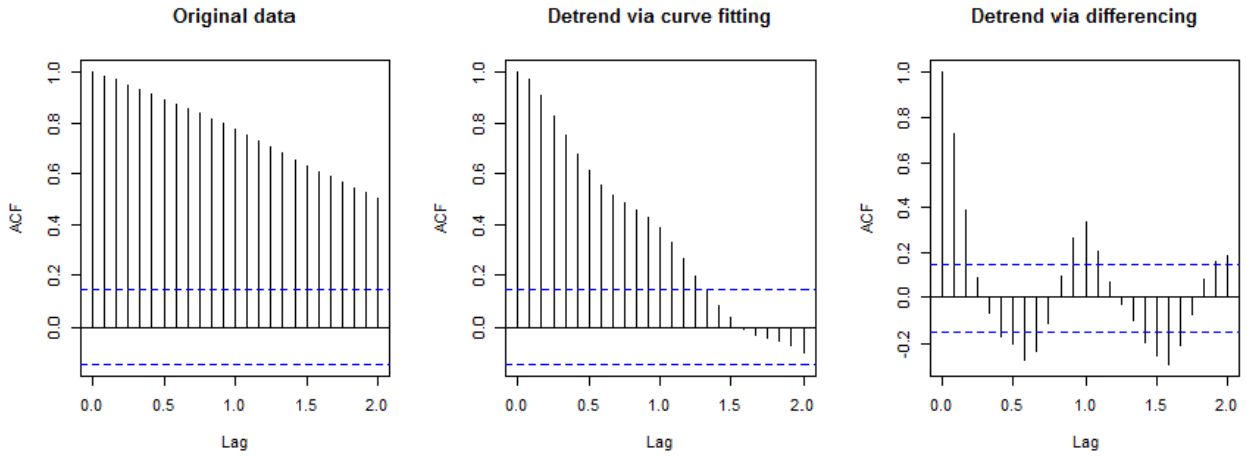



```
library(astsa)
fit <- stats::lm(chicken~time(chicken), na.action=NULL); summary(fit); anova(fit)
par(mfrow=c(1,3))
plot.ts(chicken, ylim = c(60,120), main="Original data and trend"); abline(fit, col = "blue")
plot.ts(chicken-fit$fitted.values, ylim = c(-15,15), main="Detrend via curve fitting"); abline(h=0, lty=2, col = "black")
chicken_diff=diff(chicken, differences = 1)
plot.ts(chicken_diff, ylim = c(-7,7), main="Detrend via differencing"); abline(h=0, lty=2, col = "black")
```

3. Trend – Differencing

Dealing with trend via differencing

Differencing over curve fitting has the advantage of (usually) requiring the estimation of fewer parameters and not assuming a fixed trend throughout the observation period.

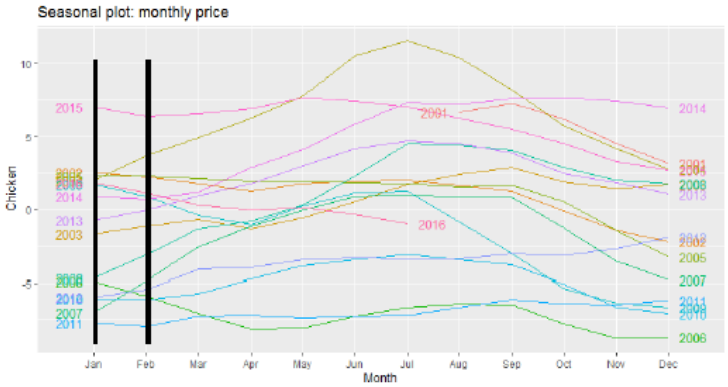


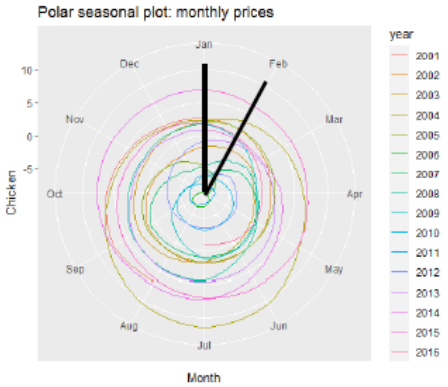

```
par(mfrow=c(1,3))
acf(chicken, lag.max = 24, main="Original data")
acf(chicken-fit$fitted.values, lag.max = 24, main="Detrend via curve fitting")
acf(diff(chicken), lag.max = 24, main="Detrend via differencing")
```

4. Seasonality – Averaging for the period

Dealing with seasonality via averaging

Idea: estimate the seasonal component by averaging the detrended series for each time point of the period over the entire observation interval.





Averaged for each time point of the period = months

```
library(astsa)
library(ggplot2)
library(forecast)
fit <- lm(chicken~time(chicken), na.action=NULL); x = chicken -fit$fitted.values
forecast::ggseasonplot(x, year.labels=TRUE, year.labels.left=TRUE) + ylab("Chicken") + ggtitle("Seasonal plot: monthly price")
ggseasonplot(x, polar=TRUE) + ylab("Chicken") + ggtitle("Polar seasonal plot: monthly prices")
```

4. Seasonality – Averaging for the period

Dealing with seasonality via averaging

After detrending the original data, an estimate of the seasonal effect (plus noise) at time t can be obtained. For the additive model, this estimate is

$$X_t - \hat{m}_t = \widehat{s_t + Y_t}$$

The overall seasonal effect $\hat{s}_i$, $i = 1,2,\dots,k$ (where k is the length of the period) can be computed by averaging $w_t = \widehat{s_t + Y_t}$ for each time point of the period over the entire observation interval of the time series.

This corresponds to the following operation

$$\hat{s}_i = \frac{1}{[n/k]} \sum_{i=1}^{k} \sum_{j=0}^{[n/k]} w_{i+j*k} \quad , i = 1,2,\dots,k$$

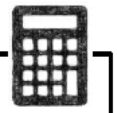
where [ ] returns the integer part of a number.

The sequence $\hat{s}_i$, $i = 1,2,\dots,k$ is then repeated over the observation interval to compose a time series with the seasonal component over the entire observation interval.

4. Seasonality – Averaging for the period

Dealing with seasonality via averaging

This approach is not directly available in R and can be implemented as follows…



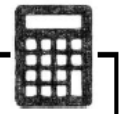
```
# Define the function
get_seasonality <- function(x) {
 ll = length(x)
 ff = frequency(x)
 periods <- ll%/%ff ## number of observed periods; %/% is integer division
 index <- seq(1, ll, by = ff) - 1
 # Get mean by each point of one period
 mm <- numeric(ff)
 for (i in 1:ff) {
  mm[i] <- mean(x[index + i], na.rm = TRUE)
  }
# mm <- mm - mean(mm) ## subtract mean to make overall mean = 0
## create ts object for seasonality
st <- ts(rep(mm, periods + 1)[seq(ll)], start = start(x),
            frequency = ff)
return(st)
}

st = get_seasonality(x) # Call the function
```

4. Seasonality – Averaging for the period

Dealing with seasonality via averaging

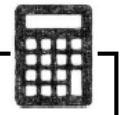
The following implementation returns an estime of the seasonality after removing the linear trend.



```
library(astsa)
fit = lm(chicken~time(chicken), na.action=NULL); x = chicken -fit$fitted.values
st = get_seasonality(x)
plot(st)
```

The estimation of the seasonality after removing the trend (computed from a finite order moving average filter) is implemented in the "Decompose" function in R, allowing for

- additive decomposition,
- multiplicative decomposition.



```
library(astsa)
x = chicken
add_model = stats::decompose(x); plot(add_model)
mult_model = stats::decompose(x, type= "multiplicative"); plot(mult_model)
```

4. Seasonality – Differencing

Dealing with seasonality via differencing

The d-difference operator applied to nonseasonal data $\nabla^d(X_t) = (1-B)^d X_t$ can be adapted to deal with seasonality of period s by the lag-s differencing operator

$$\nabla_s(X_t) = (1-B^s)X_t = X_t - X_{t-s}.$$

When applying ∇_s to the additive model

$$X_t = m_t + s_t + Y_t$$

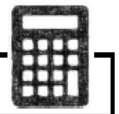
where s_t has period s (i.e. $s_t = s_{t+ks}$ with $k \in \mathbb{Z}$), then

$$\nabla_s(X_t) = X_t - X_{t-s} = m_t - m_{t-s} + Y_t - Y_{t-s}$$

which gives a decomposition into a trend $(m_t - m_{t-s})$ and a noise component $(Y_t - Y_{t-s})$.

In many cases, the use of ∇_s is sufficient to eliminate both seasonality and trend. Otherwise, the trend can be eliminated using the previous methods, e.g. by applying ∇^d to the deseasonalized series.

The base::diff(x, lag = s, differences = d, ...) implements this procedure in R (see diff help).



4. Seasonality – Harmonic regression

Dealing with seasonality via harmonic regression

When dealing with long seasonal periods, a dynamic regression with Fourier terms is often preferred to the use of ∇_s. This is suitable for models with noise and no trend as

$$X_t = s_t + Y_t$$

where s_t is a periodic function with period s (i.e. $s_t = s_{t+ks}$ with $k \in \mathbb{Z}$).

The harmonic regression makes use of sine/cosine functions to model the seasonality

$$s_t = a_0 + \sum_{j=1}^{K} \left(a_j \cos(\lambda_j t) + b_j \sin(\lambda_j t)\right)$$

where

- $a_0, a_1, \ldots, a_K, b_1, \ldots, b_K$ are unknown parameters,
- λ_j are fixed (and known) frequencies being an integer multiple of $2\pi/s$ and
- K is the number of frequencies (the higher the K, more "wiggly" is the seasonal pattern).

4. Seasonality – Harmonic regression

Dealing with seasonality via harmonic regression

Some advantages of this approach:

- it allows any length seasonality;
- allows to remove long term seasonality, leaving the short-term dynamics to be handle in the error term;
- Fourier terms of different frequencies can be included (more than one seasonal period);
- the smoothness of the seasonal pattern can be controlled by K (the pattern is smoother for smaller K);

One potential drawback is that the seasonality is assumed to be fixed i.e. the seasonal pattern is not allowed to change over time. In practice, seasonality is usually remarkably constant, so this may not be a big disadvantage except for long time series.

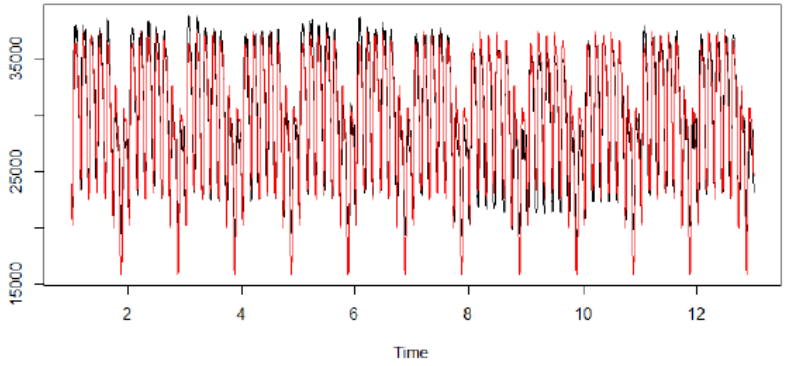
4. Seasonality – Harmonic regression

Dealing with seasonality via harmonic regression

The frequencies are fixed but K has to be selected. This can be done e.g. by fitting the harmonic terms with different K and choose that

- maximizing the fit (i.e. minimizing the sum of squared errors between the data and the fitted harmonic series);
- optimizing a predefined criterion (e.g. AIC, AICc, BIC) when coupling harmonic regression for s_t with an ARIMA model for Y_t.

```
> head(taylor)
Multi-Seasonal Time Series:
Start: 1 1
Seasonal Periods: 48 336
Data:
[1] 22262 21756 22247 22759 22549 22313
```




```
library(forecast)
plot(taylor)
head(taylor)
taylor.lm <- tslm(taylor ~ fourier(taylor, K = c(3, 3)))
ts.plot(taylor, taylor.lm$fitted.values, gpars = list(col = c("black", "red")))
```

5. STL approach

Seasonal and Trend decomposition using Loess (STL)

STL is an algorithm for decomposing a time series into trend (m_t), seasonal (s_t) and remainder (Y_t) component, following the additive model

$$X_t = m_t + s_t + Y_t.$$

STL makes use of LOESS (LOcally wEighted regreSion Smoother) to iteratively obtain $\hat{m}_t$ and $\hat{s}_t$, estimates of m_t and s_t, respectively. After, the estimate of Y_t is obtained by

$$\hat{y}_t = X_t - \hat{m}_t - \hat{s}_t.$$

Idea of the STL algorithm (Cleveland et al, 1990)

The algorithm runs through two loops:

- In the outer loop, robustness weights are assigned to each data point depending on the size of the remainder. This allows to reduce/eliminate the effects of outliers.
- The inner loop iteratively updates $\hat{m}_t$ and $\hat{s}_t$.

After finishing the algorithm, what is left is the remainder $\hat{y}_t$.

5. STL approach

Seasonal and Trend decomposition using Loess (STL)

The inner loop has the following general steps

(1) Detrend the series: $X_t - \hat{m}_t^{\,k}$ where k is the loop number;

(2) Cycle-subseries smooth: The detrended series is broken into cycle-subseries (i.e. each subseries is one cycle) and each is loess smoothed with smoothing parameter $n(s)$. The smoothed values yield a temporary seasonal time series $C_t^{\,k+1}$;

(3) Low pass filter of $C_t^{\,k+1}$ to obtain $L_t^{\,k+1}$, by applying two moving averages of lag equal to three followed by loess filtering with a span (in lags) of $n(l)$;

(4) Detrend smoothed cycle-subseries: $\hat{s}_t^{\,k+1} = C_t^{\,k+1} - L_t^{\,k+1}$, the current s_t estimate;

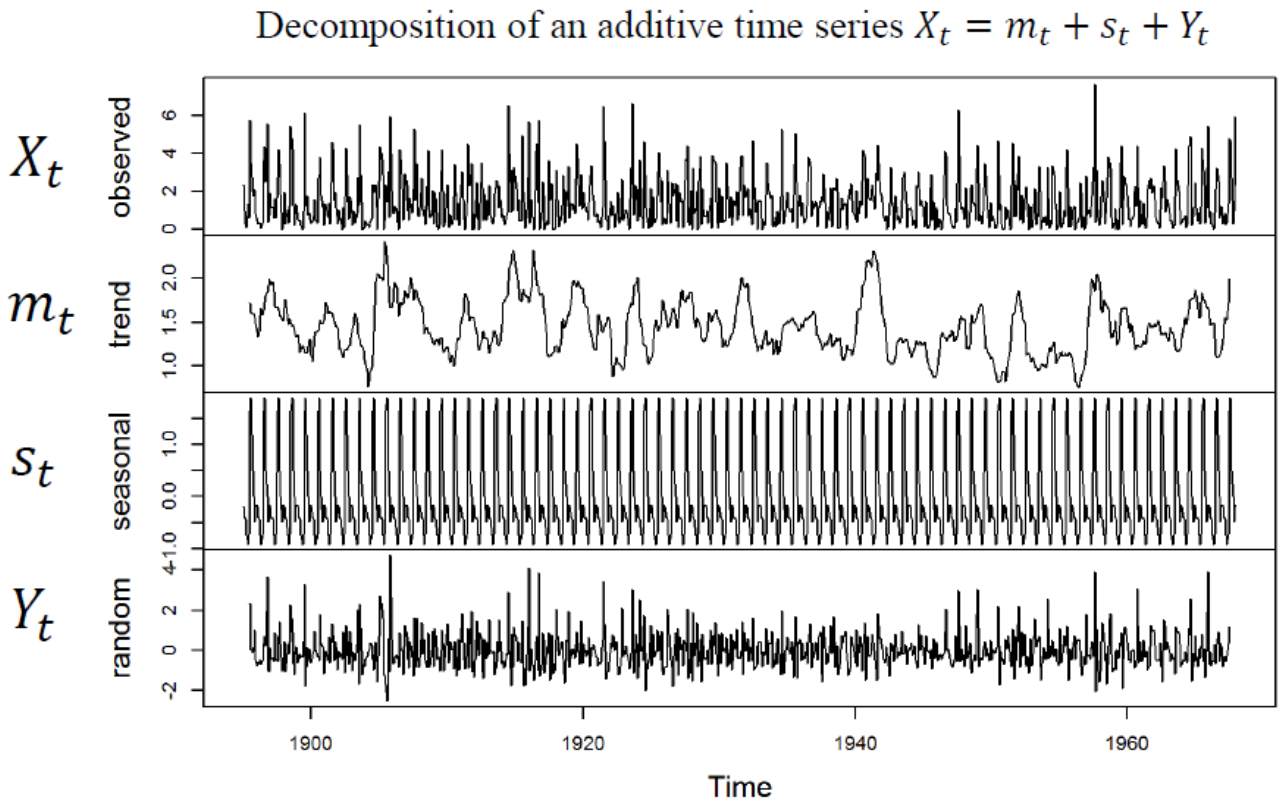
(5) Deseasonalize the series: $X_t - \hat{s}_t^{\,k+1}$;

(6) Trend smooth: Loess smooth of the deseasonalized series with parameter $n(s)$, resulting in $\hat{m}_t^{\,k+1}$, the current m_t estimate.

This process is repeated for several iterations k to improve the accuracy in the estimation of the components.

5. STL approach

Seasonal and Trend decomposition using Loess (STL)



5. STL approach

Seasonal and Trend decomposition using Loess (STL)

The implementation in R software provides default values for all parameters used by the STL algorithm, except for $n(l)$ which represents the span (in lags) of the loess window for seasonal extraction.

There are other stl versions e.g. allowing for several features including to deal with NA values, local quadratic smoothing, post-trend smoothing and endpoint blending (stlplus, stlplus package) and multiple seasonal decomposition (mstl, forecast package).

```
install.packages("forecast")
library(forecast)

# STL decomposition
head(nottem)
plot(stats::stl(nottem, s.window = "periodic"))
plot(stats:: stl(nottem, s.window = 7, t.window = 50, t.jump = 1))

# multiple seasonal STL decomposition
head(taylor)
stl2 = forecast::mstl(taylor, s.window=c(48,336))
autoplot(stl2)
```

6. Testing residuals

Testing the Estimated Noise Sequence

The goal of the described data transformations is to produce a series with no apparent deviations from stationarity and, in particular, with no apparent trend or seasonality.

After processing, the next step is to model the estimated noise sequence Y_t (i.e., the residuals obtained after detrend and deseasonalize the original data).

If there is no dependence among between these residuals, then they can be regarded as observations of independent random variables, and there is no further modeling to be done except to estimate their mean and variance.

However, if there is significant dependence among the residuals, then a stationary time series model (accounting for the dependence) should be considered for the noise. In this context, dependence means (in particular) that past observations of Y_t can help in predicting future values.

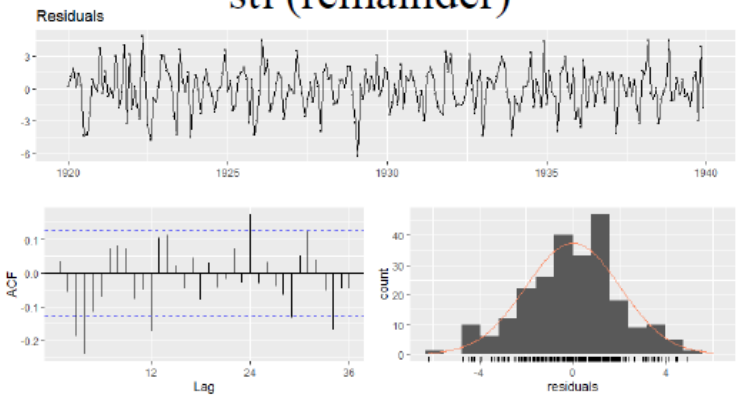
This section presents simple tools for checking the hypothesis that the residuals from are observed values of independent and identically distributed random variables.

6. Testing residuals

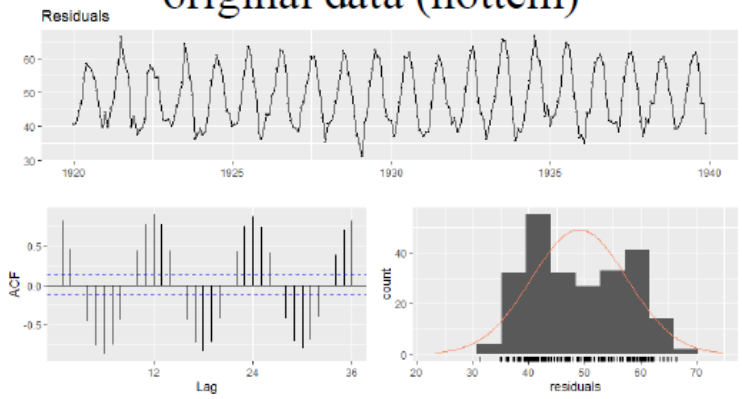
Testing the Estimated Noise Sequence

The checkresiduals implements visualization and statistical tests to check if residuals satisfy some assumptions (uncorrelated, normally distributed): produces a time plot, an ACF, a histogram with super-imposed normal curve, does a Ljung-Box or Breusch-Godfrey test on the residuals with appropriate number of lags and degrees of freedom.

stl (remainder)



original data (nottem)




```
a = stats::stl(nottem, s.window = "periodic")
head(a$time.series)
res = a$time.series[, "remainder"]
forecast::checkresiduals(res)
forecast::checkresiduals(nottem)
```

7. Final notes

Final notes

There is no unique (right or wrong) approach to address the deterministic components (trend and seasonality) of a time series. However, one should consider

(1) stabilize the variance,

(2) estimate/remove the trend,

(3) estimate/remove the seasonal component.

In practice, the practioner may design and compare different approaches to the problem solving and select that(those) with optimal fit according to a predefined criterion. E.g.

Approach 1: stabilize the variance + trend removal (by curve fitting – linear, …)

Approach 2: stabilize the variance + trend removal (by first order differencing)

Approach 3: stabilize the variance + first order and seasonal differencing

Approach 4: seasonal differencing (may also deal with trend and variance-stabilization)

7. Final notes

References

- Shumway R.H. and Stoffer D.S. (2011), Time Series Analysis and its Applications with R examples (4th ed), Springer texts in Statistics. (https://www.stat.pitt.edu/stoffer/tsa4/tsa4.pdf)
- Brockwell P.J. and Davies R.A. (2016), Introduction to Time Series and Forecasting (3rd ed), Springer Texts in Statistics. (https://link.springer.com/content/pdf/10.1007%2F978-3-319-29854-2.pdf)
- Hyndman R.J. and Athanasopoulos G. (2021), Forecasting: Principles and Practice (3rd ed), OTexts: Melbourne, Australia. (https://otexts.com/fpp3/)
- Bickel P. and Doksum K. (1981), An analysis of transformations revisited. Journal of the American Statistical Association, 76(374), 296–311.
- Box G. and Cox D. (1964), An analysis of transformations. Journal of the Royal Statistical Society. Series B, Statistical Methodology, 26(2), 211–252.
- Guerrero V. (1993), Time-series analysis supported by power transformations. Journal of Forecasting, 12, 37–48.
- Cleveland R.B., Cleveland W.S., McRae J.E. and Terpenning I. (1990), STL: A Seasonal-Trend Decomposition Procedure Based on Loess. Journal of Official Statistics, 6, 3–73.