Time series forecasting

Introduction and exponential smoothing

Julien JACQUES

Universite Lumière Lyon 2

Time series with R

Descriptive statistics for time series

Some statistical tests for time series

Exponential Smoothing

Times series

A **time series** is:

- a series of data points indexed in time order
- a sequence taken at successive equally spaced points in time.
- it is a sequence of discrete-time data

$$(x_t)_{1 \leq t \leq n} = (x_1, \ldots, x_n)$$

where t is time (seconde, day, year...).

Times series

A **time series** is:

- a series of data points indexed in time order
- a sequence taken at successive equally spaced points in time.
- it is a sequence of discrete-time data

$$(x_t)_{1 \leq t \leq n} = (x_1, \ldots, x_n)$$

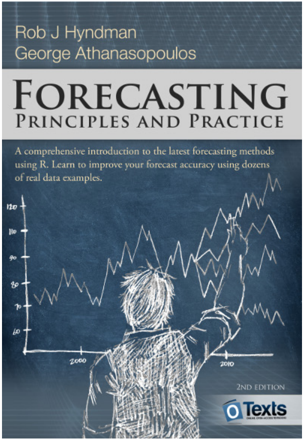
where t is time (seconde, day, year...).

Our goal is to **forecast the future** of the time series

$$x_{n+1}, x_{n+2}, \ldots$$

Reference

Hyndman R.J. and Athanasopoulos G. *Forecasting: Principles and Practice*, OTexts, 2013.
https://robjhyndman.com/uwafiles/fpp-notes.pdf



Exemples of times series

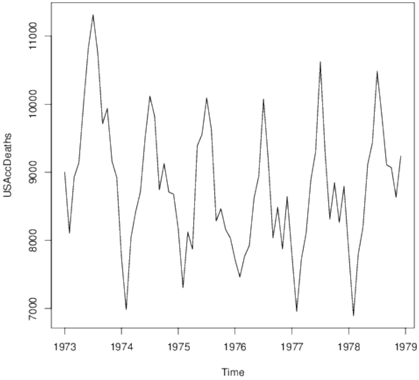


Figure 1: Number of accidental deaths in USA from 1973 to 1978

Exemples of times series

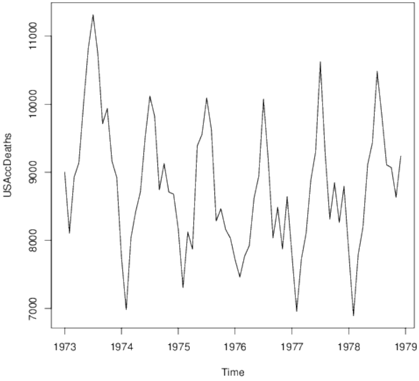


Figure 1: Number of accidental deaths in USA from 1973 to 1978

it seems to be a *periodicity*: we talk about **seasonal pattern**, which occurs when time series are affected by seasonal factor (day of the week, month of the year...). The frequency is fixed and knwon.

Exemples of times series

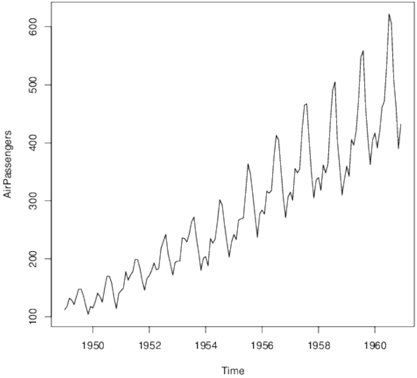


Figure 2: Monthly Airline Passenger Numbers 1949-1960

Exemples of times series

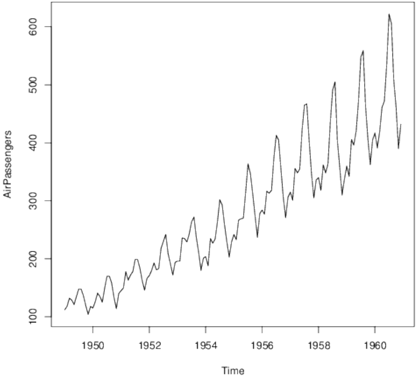


Figure 2: Monthly Airline Passenger Numbers 1949-1960

it seems to be a *seasonal* pattern but also a **trend pattern** (long-time increase or decrease, not necessarily linear)

Exemples of times series

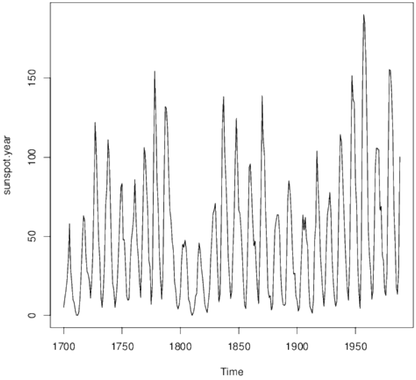


Figure 3: Annual number of sunspots observed on the surface of the sun from 1700 to 1980

Exemples of times series

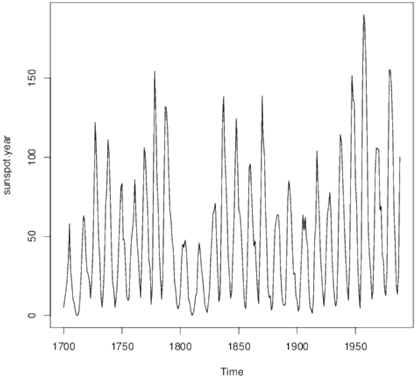


Figure 3: Annual number of sunspots observed on the surface of the sun from 1700 to 1980

it seems to be a *seasonal* pattern or maybe **cyclic pattern** (rises and falls that are not of fixed frequency)

Exemples of times series

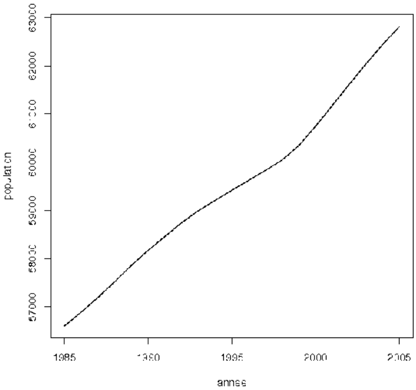


Figure 4: French population from 1985 to 2005

Exemples of times series

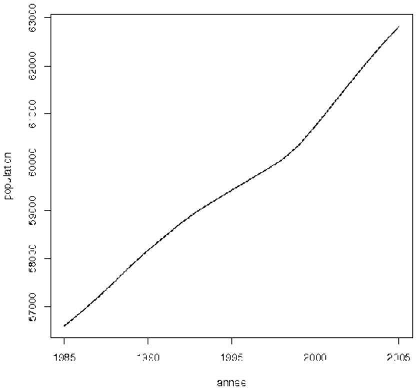


Figure 4: French population from 1985 to 2005

it seems to be a linear *trend*

Exemples of times series

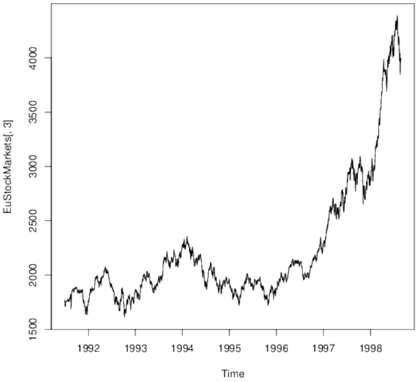


Figure 5: Daily closing values of the CAC40 from 1991 to 1998

Exemples of times series

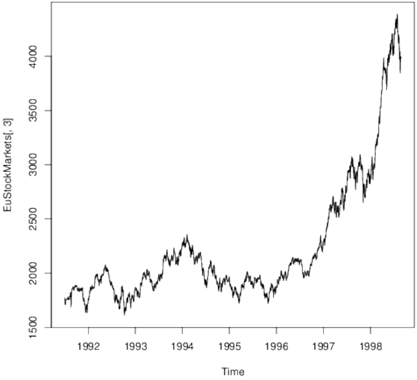


Figure 5: Daily closing values of the CAC40 from 1991 to 1998

it seems to be nothing regular. . .

Exemples of times series

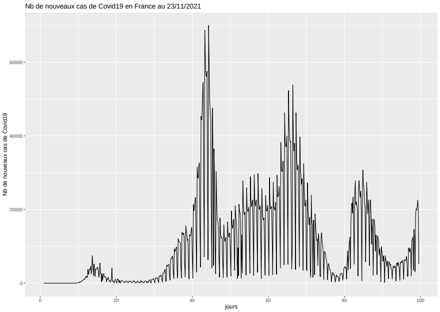


Figure 6: Covid19 number of new cases

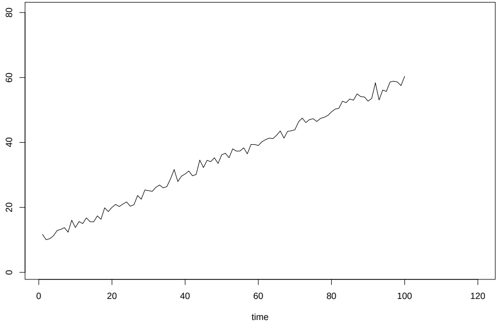
An already known forecasting method

Now, we will start with some simple forecasting method, that you already know!

An already known forecasting method

Load and plot the following series

```
data=read.table(file="http://eric.univ-lyon2.fr/jjacques/Download/DataSet/serie1.txt")
plot(data$V1,type='l',xlim=c(1,120),ylim=c(1,80),xlab='time',ylab='')
```

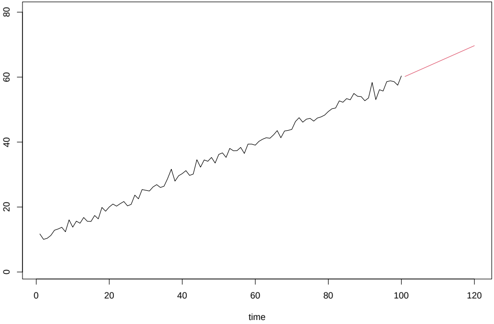


To do: Forecast this series for the next 20 times!

An already known forecasting method

We can use linear regression

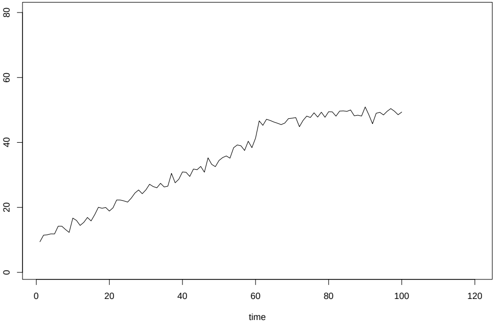
```
t=1:100;x=data$V1
model=lm(x~t)
newt=data.frame(t=101:120)
p=predict(model,newt)
plot(t,x,type='l',xlim=c(1,120),ylim=c(1,80),xlab='time',
     ylab='')
lines(newt$t,p,col=2)
```



An already known forecasting method

Load and plot the following series

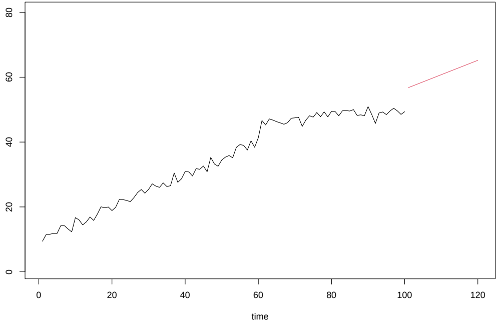
```
data=read.table(file="http://eric.univ-lyon2.fr/jjacques/Download/DataSet/serie2.txt")
plot(data$V1,type='l',xlim=c(1,120),ylim=c(1,80),xlab='time',ylab='')
```



To do: Forecast this series for the next 20 times!

An already known forecasting method

```
t=1:100;x=data$V1
model=lm(x~t)
newt=data.frame(t=101:120)
p=predict(model,newt)
plot(t,x,type='l',xlim=c(1,120),ylim=c(1,80),xlab='time',
     ylab='')
lines(newt$t,p,col=2)
```



Linear regression is not efficient since each observations have the same weight: we should be able to weight the data according to their age...

Time series with R

Time series R

In R, the `ts` object is dedicated to time series:

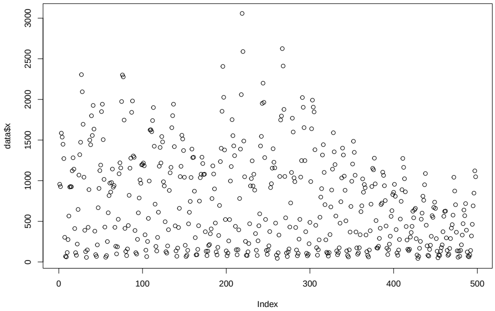
```
data("AirPassengers")
str(AirPassengers)
```

```
##  Time-Series [1:144] from 1949 to 1961: 112 118 132 129
```

Creation of ts object

We load the data from any format (here csv for instance)

```
data=read.csv(file="http://eric.univ-lyon2.fr/jjacques/Download/DataSet/varicelle.csv")
plot(data$x)
```

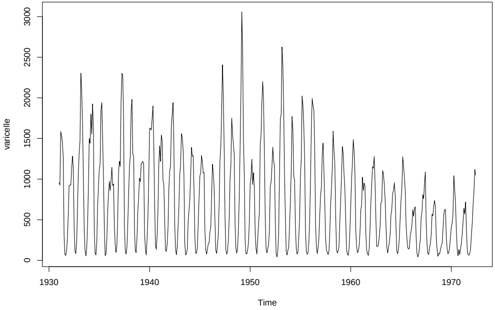


Creation of ts object

We indicate to R the specificity of the ts object:

- monthly data with annual seasonality: `freq=12`
- start in January 1931: `start=c(1931,1)`
- end in June 1972: `end=c(1972,6)`

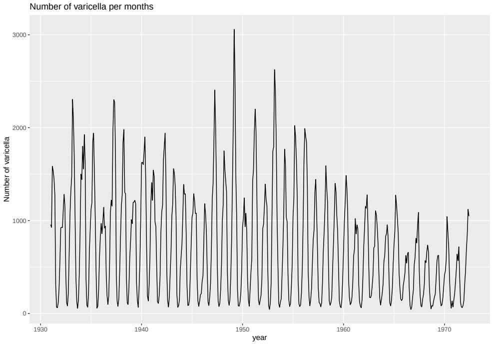
```
varicelle<-ts(data$x,start=c(1931,1),end=c(1972,6),freq=12)
plot(varicelle)
```



Plot with forecast

The forecast library proposes nice plots

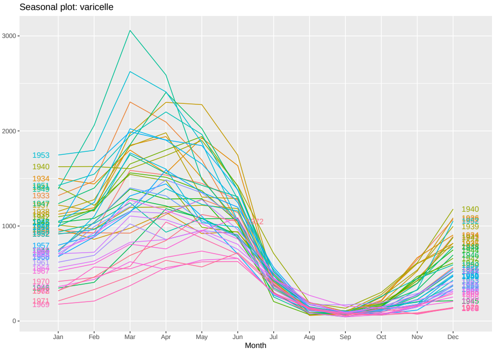
```
library(forecast)
library(ggplot2)
autoplot(varicelle) +
  ggtitle('Number of varicella per months')+
  xlab('year')+
  ylab('Number of varicella')
```



Plot with `forecast`

It could be convenient to use seasonal plot

```
ggseasonplot(varicelle,year.labels= TRUE,year.labels.left=TRUE)
```

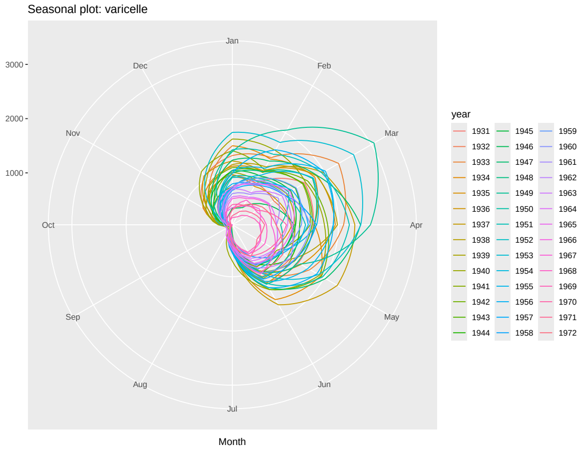


in particular for checking the size of the seasonality

Plot with forecast

or also with the polar option

```
ggseasonplot(varicelle,polar=TRUE)
```

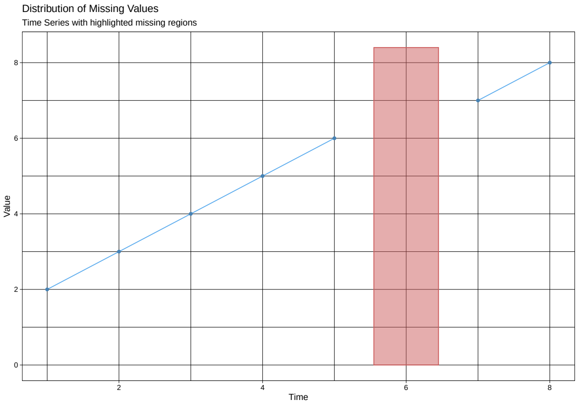


Missing data imputation

Some time series could have missing data.

The following package proposes imputation method.

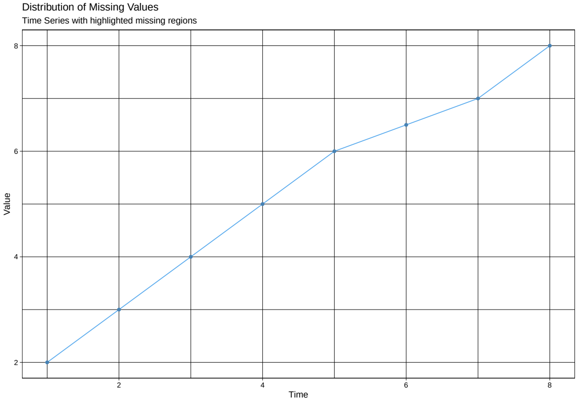
```
library(imputeTS)
x <- ts(c(2, 3, 4, 5, 6, NA, 7, 8))
ggplot_na_distribution(x)
```



Missing data imputation

Most simple imputation method is interpolation (linear, spline. . .):

```
x=na_interpolation(x)
ggplot_na_distribution(x)
```



Descriptive statistics for time series

Descriptive statistics for time series

Empirical **mean**:

$$\bar{x}_n = \frac{1}{n} \sum_{t=1}^{n} x_t$$

```r
mean(varicelle)
```

```
## [1] 732.4076
```

Empirical **variance**:

$$\hat{\sigma}_n(0) = \frac{1}{n} \sum_{t=1}^{n} (x_t - \bar{x}_n)^2$$

```r
var(varicelle)
```

```
## [1] 347785.4
```

Descriptive statistics for time series

Empirical **auto-covariance** of order h (*covariance between lagged values*):

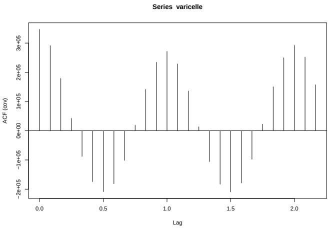
$$\hat{\sigma}_n(h) = \frac{1}{n-h}\sum_{t=1}^{n-h}(x_t - \bar{x}_n)(x_{t+h} - \bar{x}_n),$$

It measures the linear covariance between x_t and x_{t-h}

```
tmp=acf(varicelle,type="cov",plot = FALSE)
tmp$acf[1:3,1,1]
```

```
## [1] 347087.0 291348.5 179126.1
```

```
plot(tmp)
```



Descriptive statistics for time series

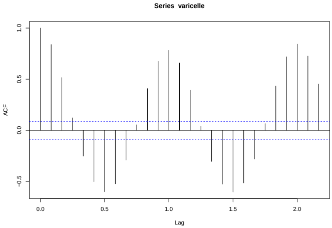
Empirical **auto-correlation** of order h:

$$\hat{\rho}_n(h) = \frac{\hat{\sigma}_n(h)}{\hat{\sigma}_n(0)} \quad \in [-1, 1]$$

```
tmp=acf(varicelle,type="cor",plot = FALSE)
tmp$acf[1:3,1,1]
```

```
## [1] 1.0000000 0.8394105 0.5160841
```

```
plot(tmp)
```



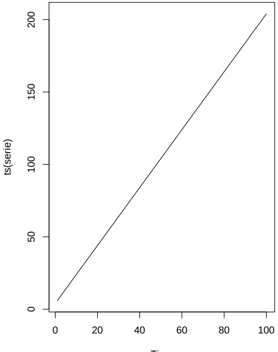
The plot is known as **correlogram**. Values into the blue lines ($\pm 2/\sqrt{n}$) are not significantly different from zero.

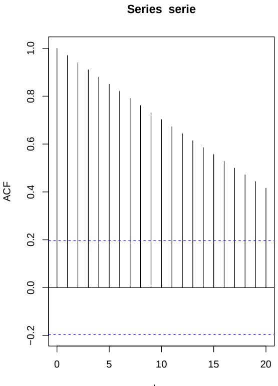
Auto-correlation properties

- if the time serie $(x_t)_{1\leq t\leq n}$ is a pure linear trend $x_t = at + b$, then for all h:

$$\hat{\rho}_n(h) \xrightarrow[n\to\infty]{} 1$$

```r
serie=2*(1:100)+4
par(mfrow=c(1,2))
plot(ts(serie))
acf(serie)
```



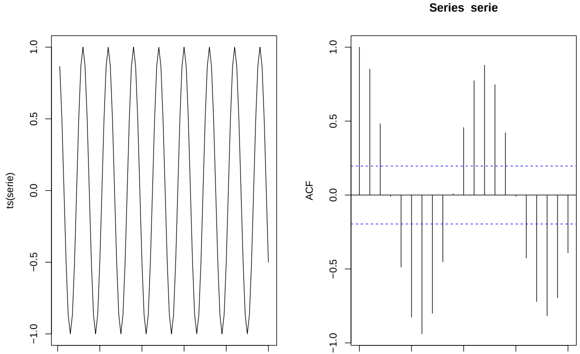


Auto-correlation properties

- if the time serie $(x_t)_{1\leq t\leq n}$ is a pure seasonal pattern, for instance $x_t = a\cos\frac{2t\pi}{T}$, then for all h:

$$\hat{\rho}_n(h) \xrightarrow[n\to\infty]{} \cos\frac{2h\pi}{T}$$

```
serie=cos(2*pi/12*(1:100))
par(mfrow=c(1,2))
plot(ts(serie))
acf(serie)
```



Auto-correlation properties

Thus, the presence of *trend* and *season pattern* are observable in the auto-correlation plots.

We can also use this plot to check the value of the periodicity...

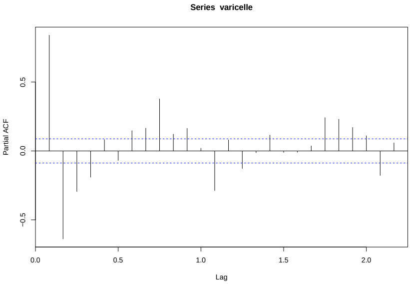
Descriptive statistics for time series

Empirical **partial auto-correlation** of order h measures the linear correlation between x_t and x_{t-h}, **but removing the effect of** $x_{t-1}, \dots, x_{t-h+1}$

```
tmp=pacf(varicelle,type="cor",plot = FALSE)
tmp$acf[1:3,1,1]
```

```
## [1]  0.8394105 -0.6382268 -0.2944475
```

```
plot(tmp)
```



Exercice

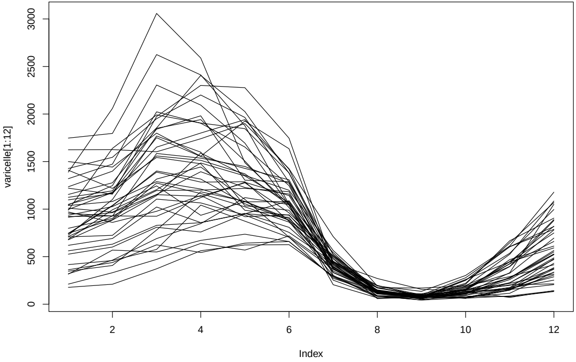
File “http://eric.univ-lyon2.fr/jjacques/Download/DataSet/varicelle.csv” contains the mensual number of varicella from January 1931 to June 1972.

- Load this data set and build a `ts` object.
- Plot the time series.
- Is there some trend, seasonal pattern or cyclic pattern?
- What is the mean mensual number of varicella?
- Plot the correlogram and interpret it.
- Plot the seasonal plot.
- Compute the annual numbers of varicella and plot them from 1931 to 1972.
- What can you say from this two latter graphs?

Correction

- Manual seasonal plot

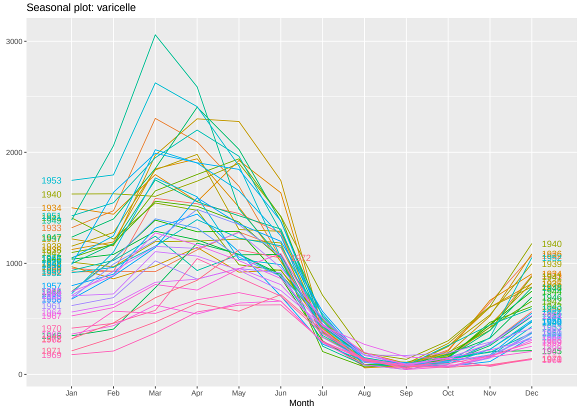
```
plot(varicelle[1:12],type="l",ylim=c(min(varicelle),max(var
for (i in 1:41) lines(varicelle[(1+12*i):(12*(i+1))])
```



Correction

- Automatic seasonal plot

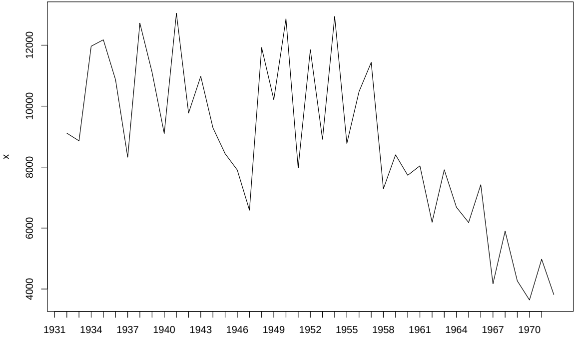
```r
ggseasonplot(varicelle,year.labels= TRUE,year.labels.left=TRUE)
```



Correction

- Annual evolution (manually)

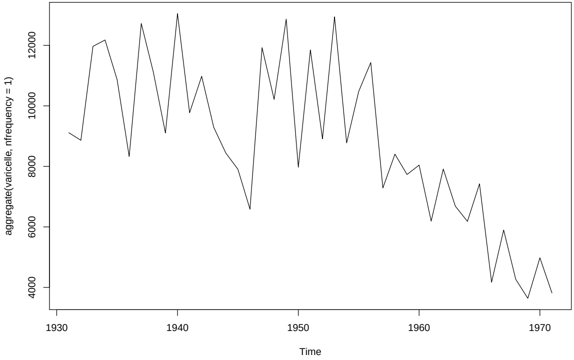
```
x=rep(0,41)
for (i in 0:40) x[i+1]<-sum(varicelle[(1+12*i):(12*(i+1))])
plot(x,type='l',xaxt='n',xlab='')
axis(1,at = 0:40,labels = 1931:1971)
```



Correction

- Annual evolution (automatically)

```
plot(aggregate(varicelle,nfrequency=1))
```



Some statistical tests for time series

Auto-correlation significativity

The Box test tests if there exists at least one among the first `lag` autocorrelations which is significant

```
Box.test(varicelle,lag=10,type="Box-Pierce")
```

```
##
##  Box-Pierce test
##
## data:  varicelle
## X-squared = 1091.7, df = 10, p-value < 2.2e-16
```

Here the p-value is very low, below than 0.05, that means that there is some significant autocorrelations among the 10 first order autocorrelations.

Tests for the presence of a trend

It is possible to test the presence of some parametric trends in the time series.

- Testing for **linear trend**: an usual t-test can not be used if the time series is auto-correlated. Noguchi, Gel, and Duguay (2011) propose an adaptation of this t-test:

```r
library(funtimes)
notrend_test(varicelle)$p.value
```

```
## [1] 0.427
```

```r
wavk_test(varicelle ~ t)$p.value
```

```
## [1] 0.208
```

Tests for the presence of a trend

- Testing for **monotonic trend** with Mann–Kendall test:

```
notrend_test(varicelle, test = "MK")$p.value
```

```
## [1] 0.471
```

- Testing for **any type of trend** (Wang, Akritas, and Van Keilegom 2008):

```
notrend_test(varicelle, test = "WAVK")$p.value
```

```
## [1] 0.743
```

- Testing for a specific parametric trend (for instance polynomial here):

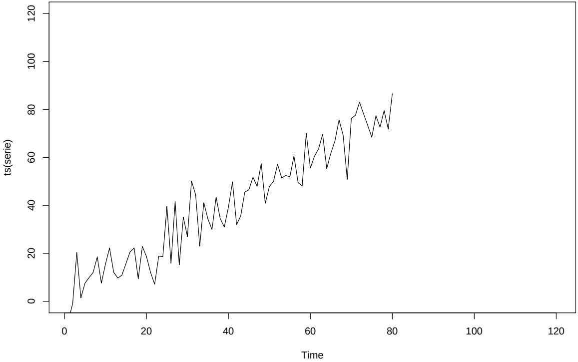
```
wavk_test(varicelle ~ poly(t, 2))$p.value
```

```
## [1] 0.064
```

Exponential Smoothing

Introduction to exponential smoothing

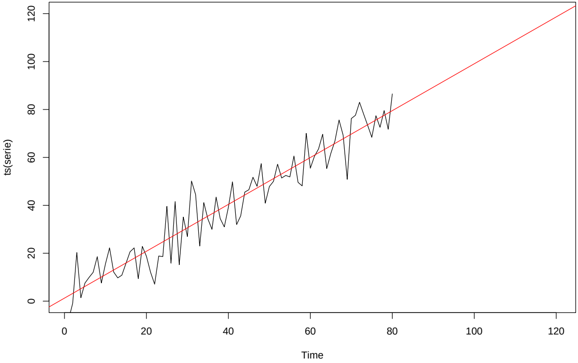
Have you an idea of forecasting model?



Introduction to exponential smoothing

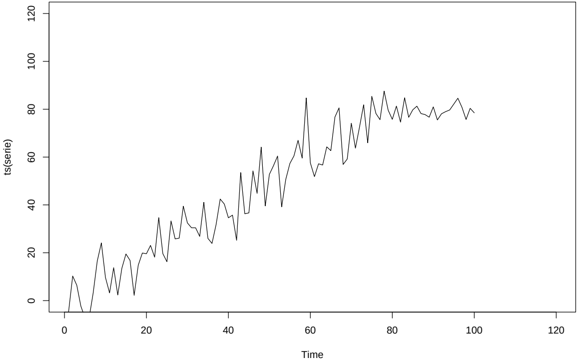
Maybe a linear regression?

```
mod=lm(serie~temps)
plot(ts(serie),xlim=c(1,120),ylim=c(0,120))
abline(mod$coefficients,col="red")
```



Introduction to exponential smoothing

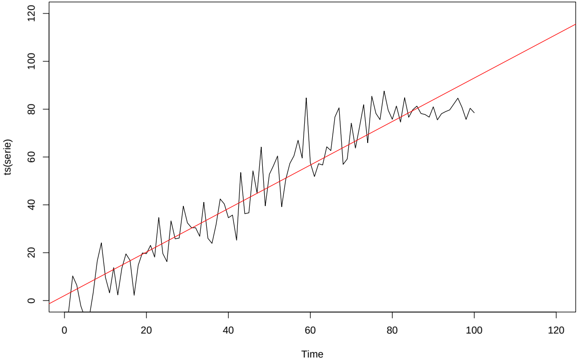
And now?



Introduction to exponential smoothing

Linear regression again?

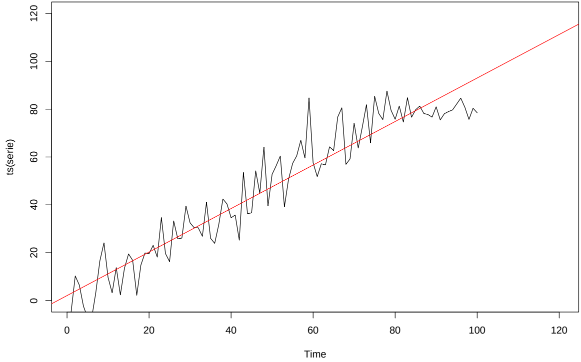
```
mod=lm(serie~temps)
plot(ts(serie),xlim=c(1,120),ylim=c(0,120))
abline(mod$coefficients,col="red")
```



Introduction to exponential smoothing

Linear regression again?

```
mod=lm(serie~temps)
plot(ts(serie),xlim=c(1,120),ylim=c(0,120))
abline(mod$coefficients,col="red")
```



Linear regression is not efficient since each observations have the same weight: we should be able to weight the data according to their age...

Exponential Smoothing

Exponential Smoothing is a collection of models (constant, linear, seasonal. . .) in which **the importance of the observed data decreases with their age**

Simple Exponential Smoothing (SES)

Given a smoothing constant $0 < \alpha < 1$, forecast with **Simple Exponential Smoothing** is:

$$\hat{x}_{n,h} = \alpha \sum_{j=0}^{n-1} (1-\alpha)^j x_{n-j}.$$

With this model, forecast is:

- **constant** (does not depend of horizon h),
- a **ponderated mean** of past observations,
- the closer is α to 1, the faster the weight of past observations decreases.

Constant α should be tuned on the data.

Evaluating forecast accuracy

How α can be chosen? More generaly, how to compare different forecasting models?

The data set $x_1, \ldots, x_n$ is separated into **train and test subsets**:

The size of the test part can depend on :

- the forecasting horizon h we want to predict
- the size of the season pattern (select 1 or 2 season in the test dataset)

Warning: once the forecasting model is selected, it should be estimated again on the whole dataset $x_1, \ldots, x_n$ before to forecast the future.

Cross validation for time series

In order to be not dependent on the test set, **time series cross-validation** can be implemented:

- the test set is moved progressively in the past
- using at each time the observations before the test set as training set

Evaluating forecast accuracy

On the test data, several indicators can be computed:

- **Root Mean Square Error**:

$$\text{RMSE} = \sqrt{\frac{1}{n-m}\sum_{h=1}^{n-m}(\hat{x}_{m,h} - x_{m+h})^2}$$

  where m is the size of training set.

- **Mean Absolute Percentage Error**:

$$\text{MAPE} = \frac{100}{n-m}\sum_{h=1}^{n-m}\frac{|\hat{x}_{m,h} - x_{m+h}|}{x_{m+h}}$$

Tools to subset a time series

Extract all data from 1950

```
window(varicelle,start=1950)
```

Extract the first or last observations

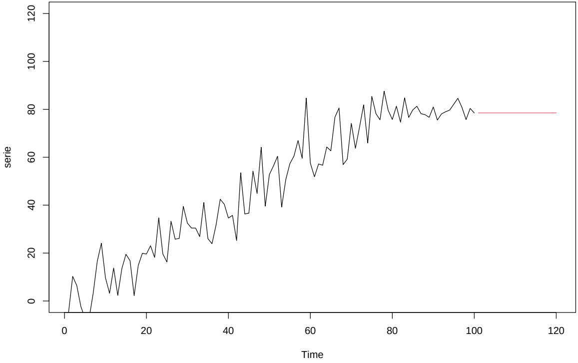
```
head(varicelle,12)
tail(varicelle,6)
```

The `subset` function allows more type of subsetting.

Simple Exponential Smoothing (SES)

SES forecast with $\alpha = 0.1$

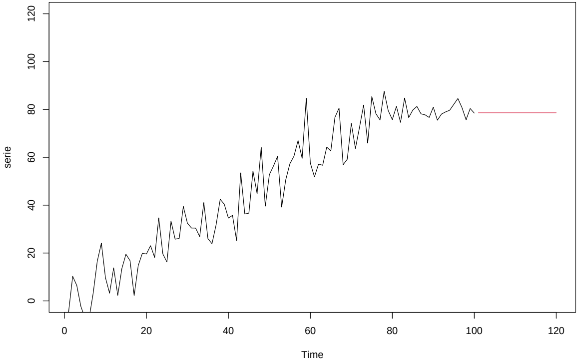
```
serie=ts(serie)
LES=HoltWinters(serie,alpha=0.1,beta=FALSE,gamma=FALSE)
plot(serie,xlim=c(1,120),ylim=c(0,120))
p<-predict(LES,n.ahead=20)
lines(p,col=2)
```



Simple Exponential Smoothing (SES)

SES forecast with $\alpha = 0.9$

```
LES=HoltWinters(serie,alpha=0.9,beta=FALSE,gamma=FALSE)
plot(serie,xlim=c(1,120),ylim=c(0,120))
p<-predict(LES,n.ahead=20)
lines(p,col=2)
```



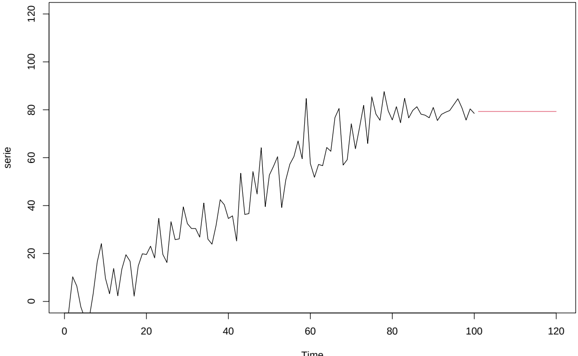
Simple Exponential Smoothing (SES)

For an automatic and optimal choice of α, let choose option alpha=NULL:

```
LES=HoltWinters(serie,alpha=NULL,beta=FALSE,gamma=FALSE)
print(LES$alpha)
```

```
## [1] 0.3350895
```

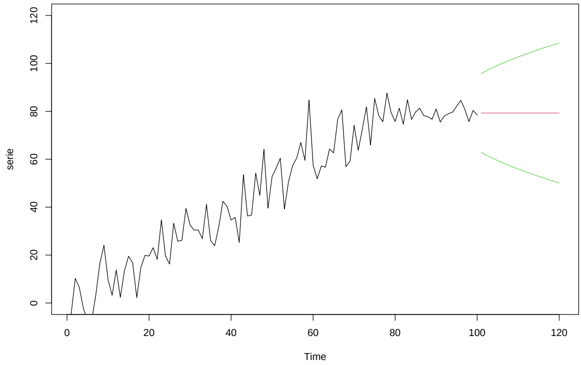
```
plot(serie,xlim=c(1,120),ylim=c(0,120))
p<-predict(LES,n.ahead=20)
lines(p,col=2)
```



Simple Exponential Smoothing (SES)

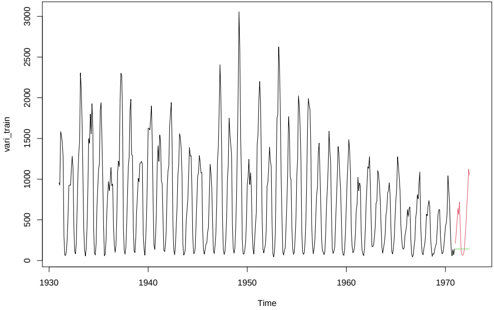
We can also add a forecasting interval.

```
LES=HoltWinters(serie,alpha=NULL,beta=FALSE,gamma=FALSE)
plot(serie,xlim=c(1,120),ylim=c(0,120))
p<-predict(LES,n.ahead=20,prediction.interval = TRUE)
lines(p[,1],col=2)
lines(p[,2],col=3);lines(p[,3],col=3);
```



Varicella forecasting with SES

```
data=read.csv(file="http://eric.univ-lyon2.fr/jjacques/Download/DataSet/varicelle.csv")
vari_train<-ts(data$x[1:480],start=c(1931,1),end=c(1970,12),freq=12)
vari_test<-ts(data$x[481:498],start=c(1971,1),end=c(1972,6),freq=12)
plot(vari_train,xlim=c(1931,1973))
lines(vari_test,col=2)
SES=HoltWinters(vari_train,alpha=NULL,beta=FALSE,gamma=FALSE)
p1<-predict(SES,n.ahead=18)
lines(p1,col=3)
```



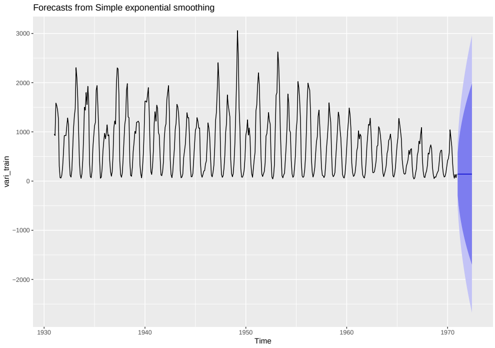
Varicella forecasting with SES

SES is also available through the `forecast` package

```
SES=ses(vari_train,h=18)
round(accuracy(SES),2)
```

```
##                 ME   RMSE    MAE    MPE MAPE MASE ACF1
## Training set -1.6 338.15 251.29 -24.53   61 1.04 0.51
```

```
autoplot(SES)
```



Different models of exponential smoothing

- Simple Exponential Smoothing: forecasting with a constant
- Non seasonal Holt-Winters smoothing: forecasting with a linear trend
- Additive seasonal Holt-Winters: forecasting with a linear trend plus a seasonal pattern
- Multiplicative seasonal Holt-Winters: forecasting with a linear trend time a seasonal pattern

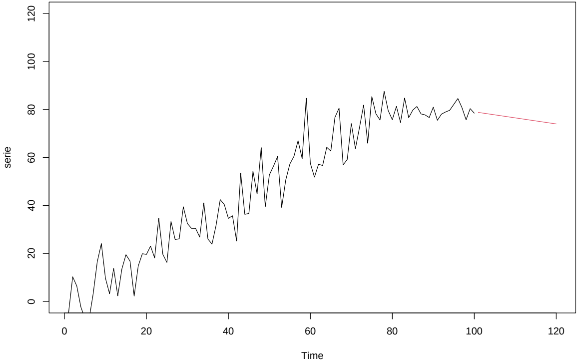
Non seasonal Holt-Winters smoothing

Forecasting is done with the linear trend

$$\hat{x}_{n,h} = \hat{a}_1 + \hat{a}_2 h.$$

This model has two smoothing constants (α, β) acting on a_1 and a_2.

```
LES=HoltWinters(serie,alpha=NULL,beta=NULL,gamma=FALSE)
plot(serie,xlim=c(1,120),ylim=c(0,120))
p<-predict(LES,n.ahead=20)
lines(p,col=2)
```



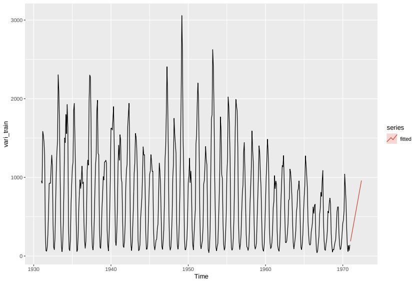
Varicella forecasting with non seasonal HW smoothing

We can also use the `forecast` package

```
HOLT=holt(vari_train,h=18)
round(accuracy(HOLT),2)
```

```
##                ME   RMSE    MAE   MPE  MAPE MASE ACF1
## Training set 0.22 330.42 259.65 54.15 89.34 1.08 0.01
```

```
autoplot(vari_train) + autolayer(HOLT,series='fitted',PI=FALSE)
```



the option `PI=FALSE` remove the prediction interval

Damped non seasonal Holt-Winters smoothing

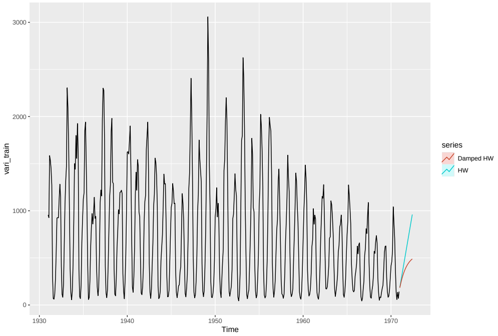
It is possible to add a damping parameter $0 < \phi < 1$ in order to dampen the trend

$$\hat{x}_{n,h} = \hat{a}_1 + \hat{a}_2(\phi + \phi^2 + \ldots + \phi^h).$$

- $\phi = 1$ lead to the usual non seasonal HW
- using $0 < \phi < 1$ dampens the trend so that it approaches a constant in the future

Varicella forecasting with non seasonal HW smoothing

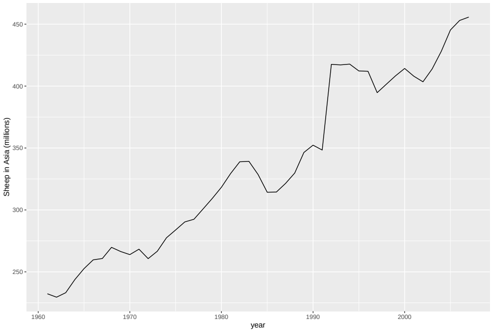
```
HOLT1=holt(vari_train,h=18)
HOLT2=holt(vari_train,damped=TRUE,phi=0.9,h=18)
autoplot(vari_train) +
  autolayer(HOLT1,series='HW',PI=FALSE) +
  autolayer(HOLT2,series='Damped HW',PI=FALSE)
```



Example: Sheep livestock in Asia

We will compare SES, HW and damped HW for forecasting the sheep livestock population in Asia.

```
library(fpp)
data(livestock)
autoplot(livestock) +
  xlab("year") +
  ylab("Sheep in Asia (millions)")
```



Example: Sheep livestock in Asia

To compare the method we can divide the time series into train / test subset, but we can also use time series cross validation implemented in `tsCV`:

```
e1 <- tsCV(livestock, ses, h=1)
e2 <- tsCV(livestock, holt, h=1)
e3 <- tsCV(livestock, holt, damped=TRUE, h=1)
```

To compare MSE:

```
mean(e1^2, na.rm=TRUE)
```

```
## [1] 178.2531
```

```
mean(e2^2, na.rm=TRUE)
```

```
## [1] 173.365
```

```
mean(e3^2, na.rm=TRUE)
```

```
## [1] 162.6274
```

The best model seems to be the Damped HW

Example: Sheep livestock in Asia

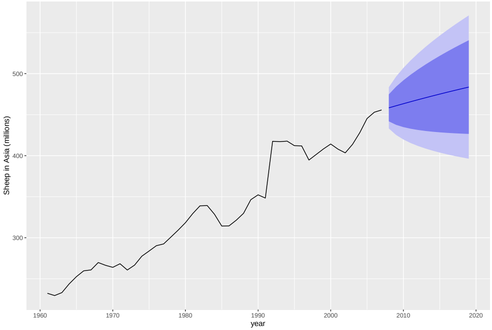
```r
HWd=holt(livestock,damped=TRUE,h=12)
HWd[["model"]]
```

```
## Damped Holt's method
##
## Call:
##  holt(y = livestock, h = 12, damped = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.9999
##     beta  = 3e-04
##     phi   = 0.9798
##
##   Initial states:
##     l = 223.35
##     b = 6.9046
##
##   sigma:  12.8435
##
##      AIC     AICc      BIC
## 427.6370 429.7370 438.7379
```

Example: Sheep livestock in Asia

Forecasting with the Damped HW

```
autoplot(livestock) +
  autolayer(HWd) +
  xlab("year") +
  ylab("Sheep in Asia (millions)")
```



Additive seasonal Holt-Winters

Now, we will add a seasonal pattern to the HW linear trend:

$$y_t = a_1 + a_2(t - n) + s_t,$$

where s_t is a seasonal pattern of period T.

Forecasting are:

$$\begin{aligned}
\hat{x}_{n,h} &= \hat{a}_1 + \hat{a}_2 h + \hat{s}_{n+h-T} \quad 1 \leq h \leq T, \\
\hat{x}_{n,h} &= \hat{a}_1 + \hat{a}_2 h + \hat{s}_{n+h-2T} \quad T+1 \leq h \leq 2T,
\end{aligned}$$

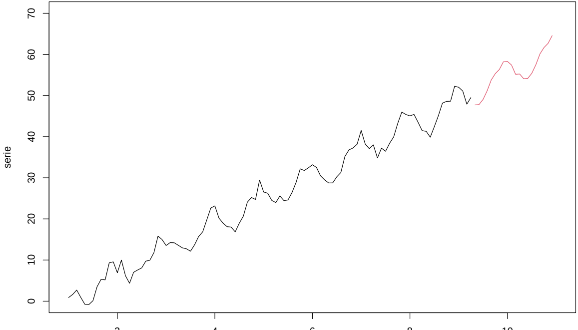
and so on for $h \geq 2T$.

This model has 3 smoothing constant α, β et γ: greater they are, lower are the importance of oldest observations. They act repectively on a_1, a_2 and s_t.

Additive seasonal Holt-Winters

In order to estimate the seasonal pattern, we should precise the corresponding period

```
serie=0.5*(1:100)+rnorm(100,0,1)+3*cos(pi/6*(1:100))
serie=ts(serie,start=c(1,1),end=c(9,4),frequency = 12)
LES=HoltWinters(serie,alpha=NULL,beta=NULL,gamma=NULL)
plot(serie,xlim=c(1,11),ylim=c(0,70))
p<-predict(LES,n.ahead=20)
lines(p,col=2)
```



Multiplicative seasonal Holt-Winters

The multiplicative seasonal Holt-Winters model is

$$y_t = [a_1 + a_2(t-n)] \times s_t,$$

where s_t is a seasonal pattern of period T.

Forecasting are:

$$\begin{aligned}
\hat{x}_{n,h} &= [\hat{a}_1 + \hat{a}_2 h]\hat{s}_{n+h-T} \quad 1 \leq h \leq T, \\
\hat{x}_{n,h} &= [\hat{a}_1 + \hat{a}_2 h]\hat{s}_{n+h-2T} \quad T+1 \leq h \leq 2T,
\end{aligned}$$

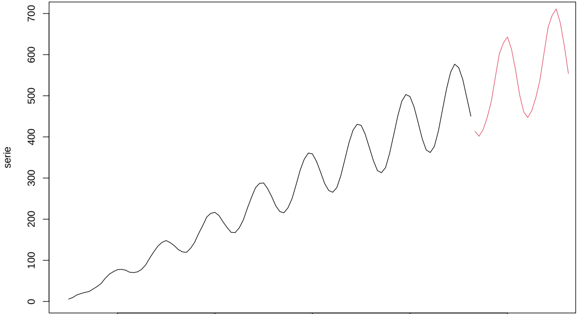
and so on for $h \geq 2T$.

Multiplicative seasonal Holt-Winters

```r
serie=5*(1:100)+rnorm(100,0,1)+cos(pi/6*(1:100))*(1:100)
serie=ts(serie,start=c(1,1),end=c(9,4),frequency = 12)
LES=HoltWinters(serie,alpha=NULL,beta=NULL,gamma=NULL,
                seasonal = "multi")
```

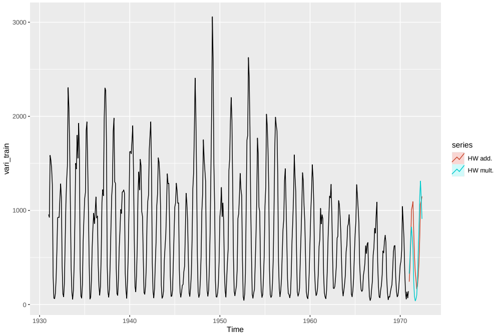
```
## Warning in HoltWinters(serie, alpha = NULL, beta = NULL, gamm
## = "multi"): optimization difficulties: ERROR: ABNORMAL_TERMIN
```

```r
plot(serie,xlim=c(1,11),ylim=c(0,700))
p<-predict(LES,n.ahead=24)
lines(p,col=2)
```



Varicella forecasting with seasonal HW smoothing

```
fit1=hw(vari_train,seasonal='additive',h=18)
fit2=hw(vari_train,seasonal='multiplicative',h=18)
autoplot(vari_train) +
  autolayer(fit1,series='HW add.',PI=FALSE) +
  autolayer(fit2,series='HW mult.',PI=FALSE)
```



Varicella forecasting with seasonal HW smoothing

We can compute the RMSE of both model

```
print(sqrt(mean((fit1$mean-vari_test)^2)))
```

```
## [1] 238.2674
```

```
print(sqrt(mean((fit2$mean-vari_test)^2)))
```

```
## [1] 214.7901
```

The multiplicative seasonal Holt-Winters seems to be the best.

Varicella forecasting with seasonal HW smoothing

We can also compare with damped version of the seasonal HW, but the results are not better:

```
fit3=hw(vari_train,seasonal='additive',damped=TRUE,h=18)
fit4=hw(vari_train,seasonal='multiplicative',damped=TRUE,h=18)
print(sqrt(mean((fit1$mean-vari_test)^2)))
```

```
## [1] 238.2674
```

```
print(sqrt(mean((fit2$mean-vari_test)^2)))
```

```
## [1] 214.7901
```

```
print(sqrt(mean((fit3$mean-vari_test)^2)))
```

```
## [1] 279.7424
```

```
print(sqrt(mean((fit4$mean-vari_test)^2)))
```

```
## [1] 375.6358
```

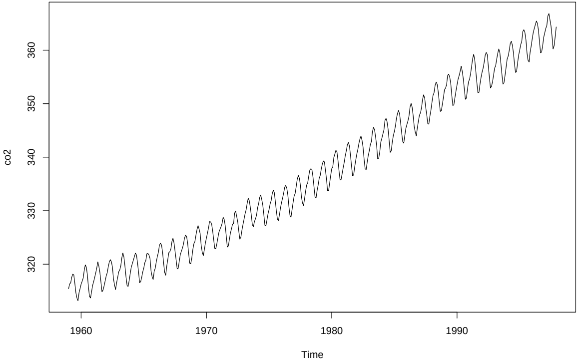
Exercice: forecasting of co2 concentration

File co2 contains CO_2 concentrations near Mauna Loa volcano (Hawaï) from 1959 to 1997.

1. plot the data
2. which exponential smoothing model could be appropriate?
3. In order to validate your choice, evaluate your forecast on data between 1990 and 1997 using as training set data from 1959 to 1989.
4. If your forecast seems correct, let use this model to forecast co2 concentration from 1997 to 2007. If not, try other exponential smoothing models.

Exercice: forecasting of co2 concentration

```r
plot.ts(co2)
```



Exercice: cross-validated MAPE

Our goal is now to forecast only one year of $CO2$ concentrations.

1. Keep year 1997 as final test set.
2. Implement cross validation on data until 1996, for comparing two models. That means:

- to forecast 1996 using data until 1995,
- to forecast 1995 using data until 1994,
- ... and at each time store your MAPE errors.

4. Which is the best one? Is this confirm on the test set (1997)?

Exponential smoothing: conclusion

Exponential smoothing is an effective forecasting method, which takes into account:

- a linear trend in the series
- a seasonal pattern

These components of the series are **deterministic** (i.e. not stochastic).

In the sequel, we will see models for the **stochastic** part of the time series.

San Francisco precipitation forecast

San Fransisco precipitation from 1932 to 1966 are available here:
http://eric.univ-lyon2.fr/jjacques/Download/DataSet/sanfran.dat

- Data until 1963 will be used as training set, in order to forecast precipitations for 1964, 1965 and 1966.
- Test several exponential smoothing models, and plot on the same graph the forecast and actual values.
- Which model seems to be graphically the best? And for RMSE?
- Interpret the value of the smoothing (and eventually damping) parameters.

Time series forecasting

ARIMA models

Julien JACQUES

Université Lumière Lyon 2

Trend and seasonal pattern estimation

ARMA models

Non-seasonal ARIMA models

Seasonal ARIMA models

Heteroscedastic series

Trend and seasonal pattern estimation

Removing trend + seasonal pattern

In order to modelize the stochastic part of the times series, we have to **remove the deterministic part** (trend + seasonal pattern)

We will see two methods:

- Estimation by moving average
- Removing by differencing

Time series components

We assume that the time series can be decomposed into:

$$x_t = T_t + S_t + \epsilon_t$$

where :

- T_t is the trend,
- S_t is the seasonal pattern (of period T)
- ϵ_t is the residual part

Rk: if x_t admits a multiplicative decomposition, $\log x_t$ admits an additive decomposition.

Moving average

A moving average estimation of the trend T_t of order m (m-MA) is:

$$\hat{T}_t = \frac{1}{m} \sum_{j=-k}^{k} x_{t+j}$$

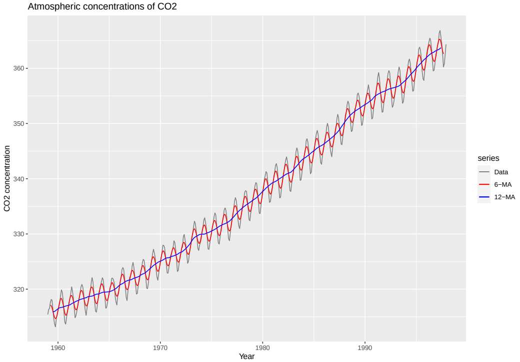
where $m = 2k + 1$.

$\hat{T}_t$ is the average of the m values nearby time t.

- greater is m, greater is the smoothing
- for series with seasonnal pattern of period T, we generally choose $m \geq T$.

Moving average

```
autoplot(co2, series="Data") +
  autolayer(ma(co2,6), series="6-MA") +
  autolayer(ma(co2,12), series="12-MA") +
  xlab("Year") + ylab("CO2 concentration") +
  ggtitle("Atmospheric concentrations of CO2 ") +
  scale_colour_manual(
    values=c("Data"="grey50","6-MA"="red","12-MA"="blue"),
    breaks=c("Data","6-MA","12-MA"))
```



Moving average

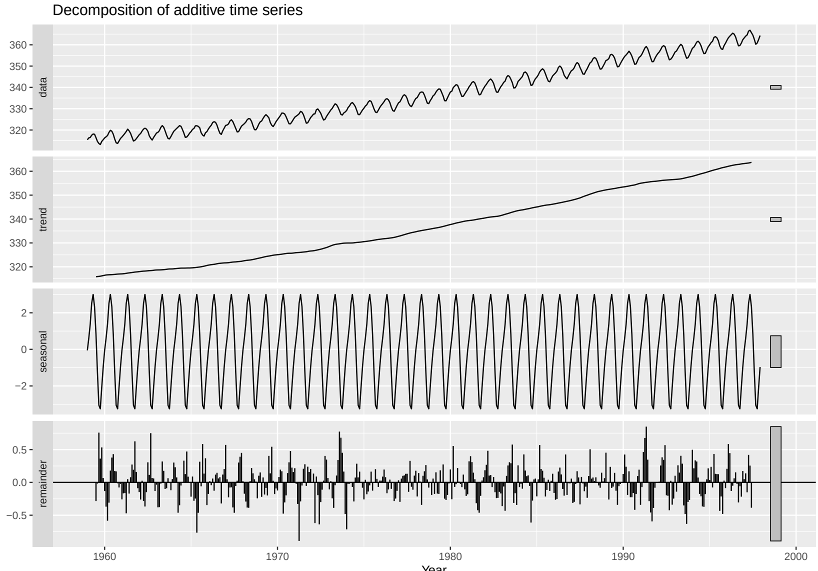
Once the trend T_t has been estimated, we remove it from the series:

$$\tilde{x}_t = x_t - \hat{T}_t$$

Estimation of the **seasonal pattern** is obtained by simply **averaging the values of $\tilde{x}_t$ on each season**.

Moving average

```
autoplot(decompose(co2,type="additive"))+
  xlab('Year')
```



Moving average

Advantage:

- quickly gives an overview of the components of the series

Disadvantage:

- no forecast is possible with such non parametric estimation

Differencing

Let Δ_T be the operator of *lag* T which maps x_t to $x_t - x_{t-T}$:

$$\Delta_T x_t = x_t - x_{t-T}.$$

Differencing

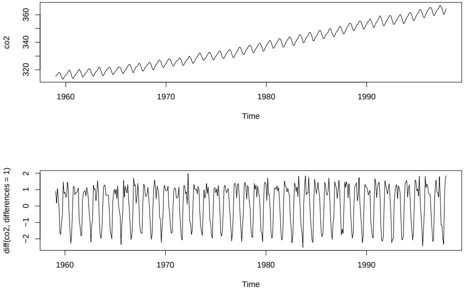
Let x_t be a time series with a polynomial trend of order k :

$$x_t = \sum_{j=0}^{k} a_j t^j + \epsilon_t.$$

Then $\Delta_T x_t$ **admits a polynomial trend of order** $k-1$.

Applying Δ_T reduces by 1 the degree of the polynomial trend.

```
par(mfrow=c(2,1))
plot(co2)
plot(diff(co2,differences=1))
```

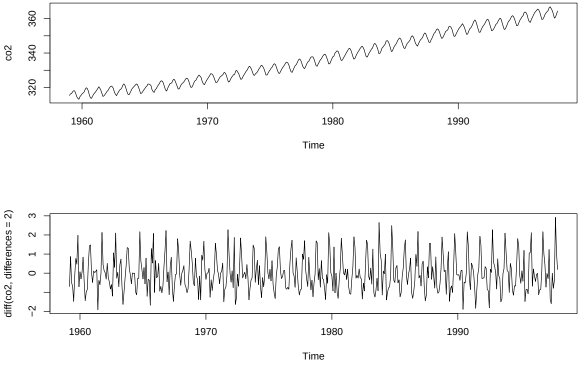


Differencing

Applying Δ_T k times reduces by k the degree of the polynomial trend.

$$\Delta_T^k = \underbrace{\Delta_T \circ \ldots \circ \Delta_T}_{k \text{ times}}$$

```r
par(mfrow=c(2,1))
plot(co2)
plot(diff(co2,differences=2))
```



Differencing

Let x_t be a time series with a ternd T_t and a season pattern S_t of period T:

$$x_t = T_t + S_t + \epsilon_t.$$

Then,

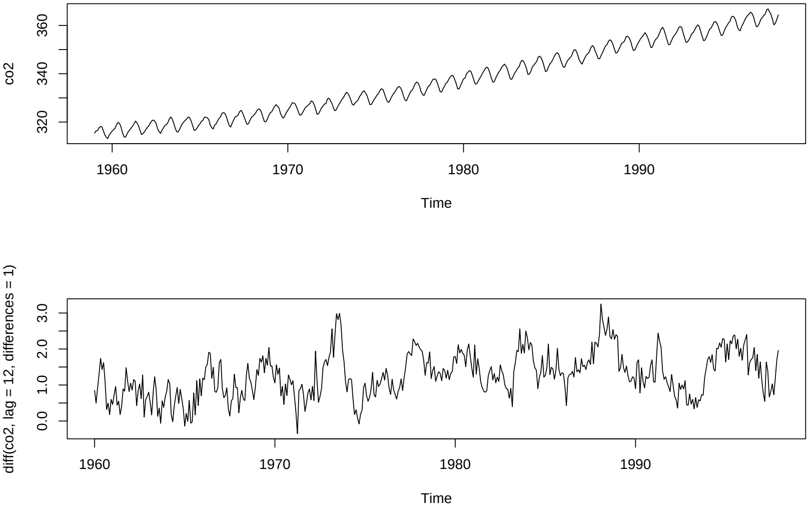
$$\Delta_T x_t = (T_t - T_{t-T}) + (\epsilon_t - \epsilon_{t-T})$$

does not admit any more seasonal pattern.

Applying Δ_T^k remove a seasonal pattern of period T and a polynomial trend of order k

Differencing

```
par(mfrow=c(2,1))
plot(co2)
plot(diff(co2,lag=12,differences=1))
```



Differencing

Advantage:

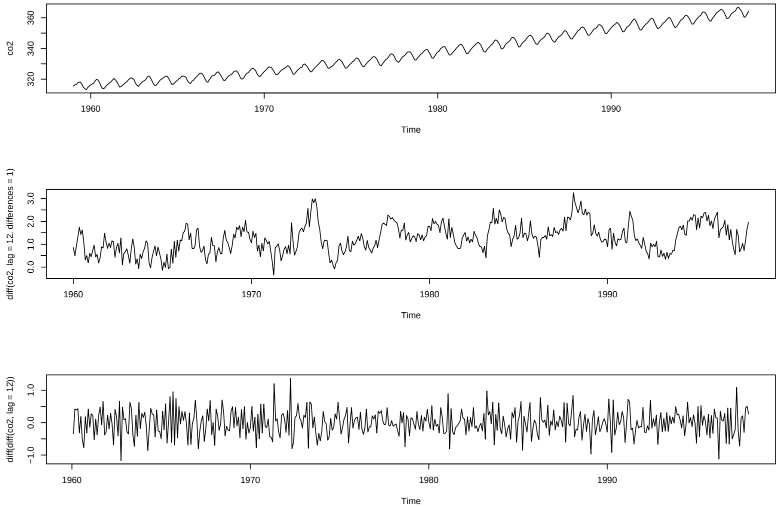
- easy to understand
- allows forecast since we can forecast $\Delta_T x_t$ and then go back to x_t

In practice :

- we start by removing the season by applying Δ_T
- then, if it visually does not seem stationary, we apply again Δ_1
- eventually we apply again Δ_1, but we will try to keep small value for the number k of differencing.

Differencing

```
par(mfrow=c(3,1))
plot(co2)
plot(diff(co2,lag=12,differences=1))
plot(diff(diff(co2,lag=12)))
```



Stationary series

x_t is a **stationary time series** if, for all s, the distribution of $(x_t, \ldots, x_{t+s})$ does not depend on t.

Consequently, a stationary time serie is one whose properties do not depend on the time at which the series is observed.

In particular, a stationary time serie has:

- no trend
- no season pattern

(A stationary time serie can have a cyclic pattern since its period is not constant.)

ARMA models, one of the main objects of this course, are models for stationary time serie.

White noise

A **white noise** is an independent and identically distributed series with zero mean.

A Gaussian white noise ϵ_t are i.i.d. observations from $\mathcal{N}(0, \sigma^2)$

In such series, there is nothing to forecast. Or more precisely, the best forecast for such series is its means: 0.

White noise

After having differecing our time series for removing trend + seasonal pattern, we have to **check that the residual series is not a white noise**.

In the countrary case, our work is finished: there is nothing else to forecast than trend and seasonal pattern, thus let use exponential smoothing.

```
Box.test(diff(co2,lag=12,differences=1),lag=10,type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  diff(co2, lag = 12, differences = 1)
## X-squared = 1415.4, df = 10, p-value < 2.2e-16
```

Here the p-value is very low, we reject that `diff(co2,lag=12,differences=1)` can be assimilted to a white noise

Exercice

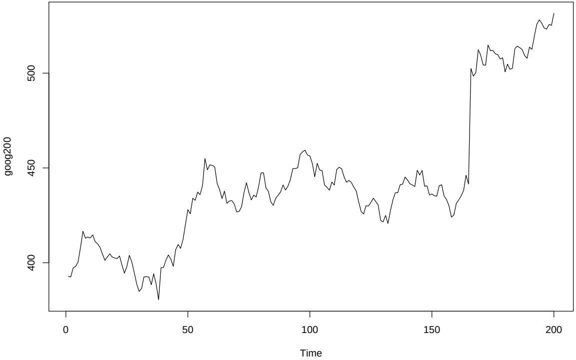
We study the number of passengers per month (in thousands) in air transport, from 1949 to 1960. This time series is available on R (`AirPassengers`).

- Plot this time series graphically. Do you think this process is stationary? Does it show trends and seasonality?
- Apply the differencing method to remove trend and seasonal pattern. Specify the period of the seasonal pattern, the degree of the polynomial trend.
- Does the differenciated series seems stationary?
- Is it a white noise?

Exercice

Same exercice with the Google stock price:

```r
library(fpp2)
plot(goog200)
```



ARMA models

Autoregressive models AR_p

An autoregressive model (x_t) of order p (AR_p) can be written:

$$x_t = c + \epsilon_t + \sum_{j=1}^{p} a_j x_{t-j}, \tag{1}$$

where ϵ_t is a white noise of variance σ^2.

An AR_p model is the sum of:

- a random chock ϵ_t, independent from previous observation
- a linear regression of the previous obseration $\sum_{j=1}^{p} a_j X_{t-j}$

Rk: we restrict AR_p models to stationary models, which implies some restrictions on the value of the coefficients a_j.

AR_p properties

- autocorrelation $\rho(h)$ exponentialy decreases to 0 when $h \to \infty$
- partial autocorrelation $r(h)$ is null for all $h > p$, and is equal to a_p at order p :

$$\begin{aligned} r(h) &= 0 \qquad \forall h > p, \\ r(p) &= a_p. \end{aligned}$$

Example of AR_1

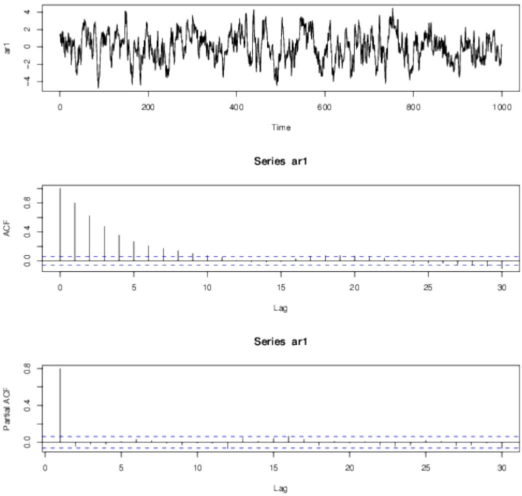


Figure 1: $AR1$ ($x_t = 0.8x_{t-1} + \epsilon_t$), autocorrelation et partial autocorrelation

Example of AR_1

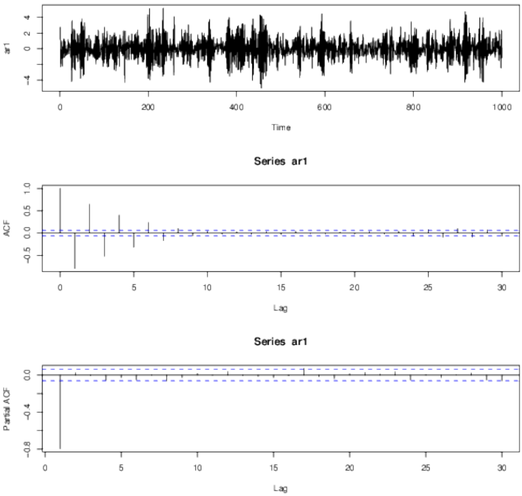


Figure 2: $AR1$ ($x_t = -0.8x_{t-1} + \epsilon_t$), autocorrelation et partial autocorrelation

Example of AR_2

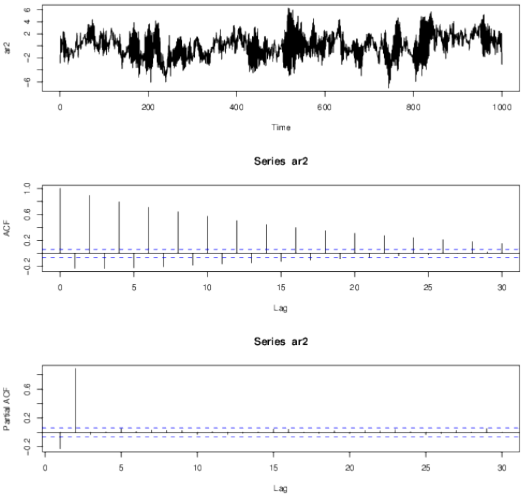


Figure 3: AR_2 ($x_t = 0.9x_{t-2} + \epsilon_t$), autocorrelation et partial autocorrelation

Example of AR_2

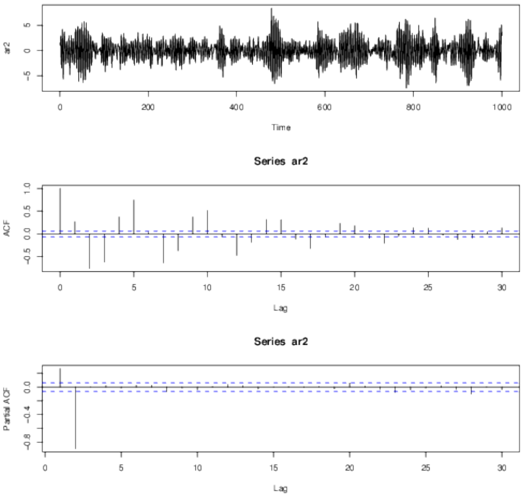


Figure 4: AR_2 ($x_t = -0.5x_{t-1} - 0.9x_{t-2} + \epsilon_t$), autocorrelation et partial autocorrelation

It's your turn!

Function `arima.sim` allows to simulate an AR_p.

Do it several times and observe the auto-correlations (partial or not)

```r
par(mfrow=c(3,1))
modele<-list(ar=c(0.8))
ar1<-arima.sim(modele,1000)
plot.ts(ar1)
acf(ar1)
pacf(ar1)
```

Moving average models MA_q

A moving average model (x_t) of order q (MA_q) can be written:

$$X_t = c + \epsilon_t + b_1\epsilon_{t-1} + \ldots + b_q\epsilon_{t-q},$$

where ϵ_j for $t - q \leq j \leq t$ are white noises of variance σ^2.

Warning: Moving average models should not be confused with moving average smoothing...

MA_q properties

- autocorrelation $\rho(h)$ is null for all $h > q$:

$$\sigma(h) = \begin{cases} \sigma^2 \sum_{k=0}^{q-h} b_k b_{k+h} & \forall h \leq q \\ 0 & \forall h > q \end{cases} \quad \text{où } b_0 = 1$$

- partial autocorrelation exponentialy decreases to 0 when $h \to \infty$
- any AR_p can be seen as an MA_∞
- under some conditions on the b_j, an MA_q can be seen as an AR_∞

Example of MA_1

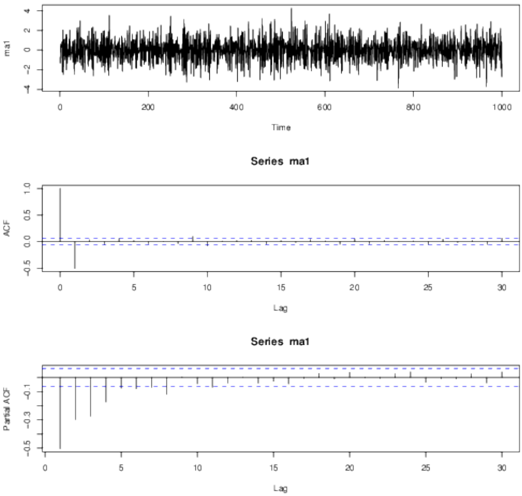


Figure 5: MA_1 ($x_t = \epsilon_t - 0.8\epsilon_{t-1}$), autocorrelation et partial autocorrelation

Example of MA_1

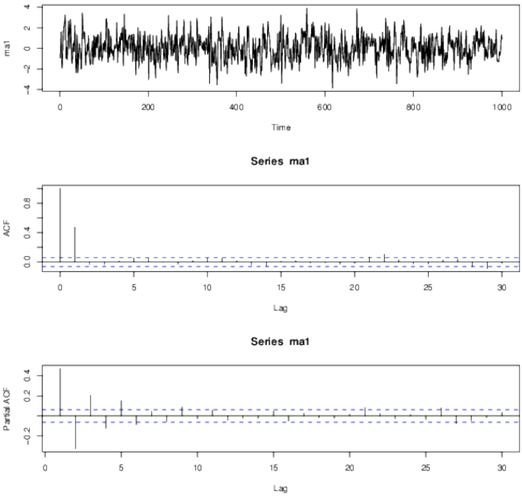


Figure 6: MA_1 ($x_t = \epsilon_t + 0.8\epsilon_{t-1}$), autocorrelation et partial autocorrelation

Example of MA_3

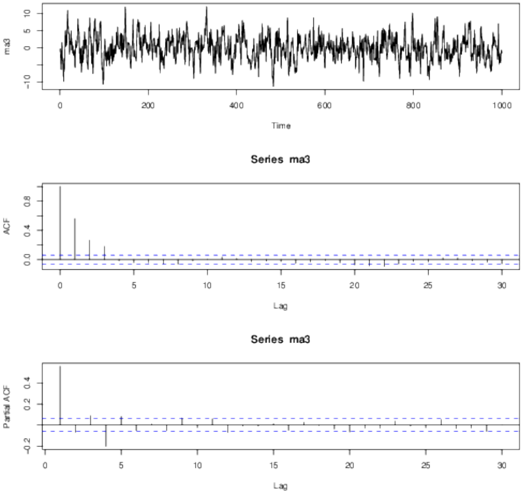


Figure 7: MA_3, autocorrelation et partial autocorrelation

It's your turn!

Function `arima.sim` allows to simulate an MA_q.

Do it several times and observe the auto-correlations (partial or not)

```
modele<-list(ma=c(0.8))
ma1<-arima.sim(modele,1000)
plot.ts(ma1)
acf(ma1)
pacf(ma1)
```

Autoregressive moving average model $ARMA_{pq}$

An autoregressive moving average model $ARMA_{pq}$ can be written:

$$x_t = c + \sum_{k=1}^{p} a_k x_{t-k} + \sum_{j=0}^{q} b_j \epsilon_{t-j}.$$

where ϵ_j for $t - q \leq j \leq t$ are white noise of variance σ^2.

Properties

- autocorrelation of an $ARMA_{p,q}$ exponentially descreases to 0 when $h \to \infty$, *from order* $q + 1$.

Example of $ARMA_{2,2}$

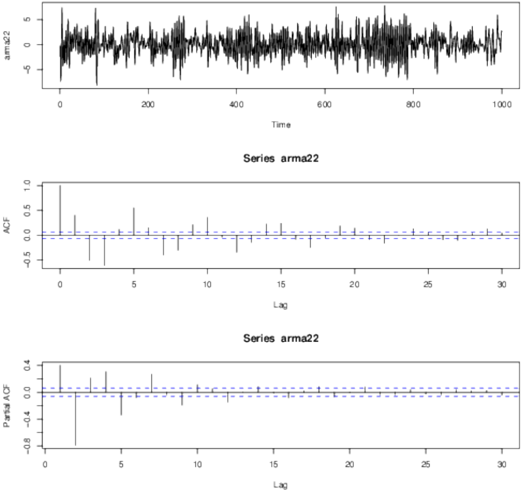


Figure 8: $ARMA_{2,2}$, autocorrelation et partial autocorrelation

Properties of MA_q, AR_p and $ARMA_{p,q}$

	MA_q	AR_p	$ARMA_{p,q}$
ACF	$\rho(h) = 0 \; \forall h > q$	$\lim_{h\to\infty} \rho(h) = 0$	$\forall h > q, \; \lim_{h\to\infty} \rho(h) = 0$
PACF	$\lim_{h\to\infty} r(h) = 0$	$r(h) = 0 \; \forall h > p$	
		et $r(p) = a_p$	

These properties *may* help to identify the order of a MA_q or an AR_p...

Non-seasonal ARIMA models

Non-seasonal ARIMA models

x_t is an $ARIMA_{p,d,q}$ model if $\Delta^d x_t$ is an $ARMA_{p,q}$ model
($\Delta^d x_t$ is x_t differenced d times)

ARIMA means *Auto Regressive Integrated Moving Average*

Selecting the orders p, d and q can be difficult.

Understanding ARIMA models

The intercept c of the model and the differencing order d have an important **effect on the long-term forecasts**:

- $c = 0$ and $d = 0 \Rightarrow$ long-term forcasts go to 0
- $c = 0$ and $d = 1 \Rightarrow$ long-term forcasts go to constant $\neq 0$
- $c = 0$ and $d = 2 \Rightarrow$ long-term forcasts will follow a straight line
- $c \neq 0$ and $d = 0 \Rightarrow$ long-term forcasts go to the mean of the data
- $c \neq 0$ and $d = 1 \Rightarrow$ long-term forcasts will follow a straight line
- $c \neq 0$ and $d = 2 \Rightarrow$ long-term forcasts will follow a quadratic trend

Some particular ARIMA models

- $ARIMA_{(0,1,0)}$ = random walk
- $ARIMA_{(0,1,1)}$ without constant = simple exponential smoothing
- $ARIMA_{(0,2,1)}$ without constant = linear exponential smoothing
- $ARIMA_{(1,1,2)}$ with constant = damped-trend linear exponential smoothing

Estimation

Once orders (p, d, q) are selected, **maximum likelihood estimation** (MLE) through optimization algorithms is used to estimate model parameters $\theta = (c, a_1, \ldots, a_p, b_1, \ldots, b_q)$

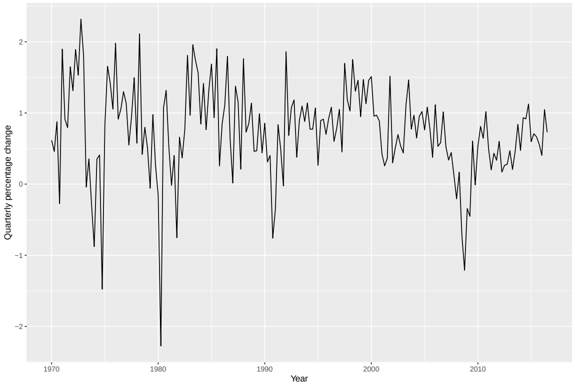
Model selection

- MLE can not be used to choose orders (p, d, q):
  higher are $(p, d, q) \Rightarrow$ higher is the number of parameters $\Rightarrow$ higher is the flexibility of the model $\Rightarrow$ higher is the likelihood
- MLE should be penalized by the complexity of the model ($\simeq$ number of parameters $\nu = p + q + 2$):
  - $AIC = -2 \log L(\hat{\theta}) + 2\nu$
  - $BIC = -2 \log L(\hat{\theta}) + \ln(n)\nu$
  - or for small sample size $AICc = AIC + \frac{2\nu(\nu+1)}{n-\nu-1}$
- or directly compute RMSE on test data

Example: US consumption expenditure

The following data contains quarterly percentage changes in US consumption expenditure

```r
library(fpp2)
autoplot(uschange[,"Consumption"]) +
  xlab("Year") + ylab("Quarterly percentage change")
```



Example: US consumption expenditure

```r
Arima(uschange[,"Consumption"],order=c(2,0,2))
```

```
## Series: uschange[, "Consumption"]
## ARIMA(2,0,2) with non-zero mean
##
## Coefficients:
##          ar1      ar2      ma1     ma2    mean
##       1.3908  -0.5813  -1.1800  0.5584  0.7463
## s.e.  0.2553   0.2078   0.2381  0.1403  0.0845
##
## sigma^2 = 0.3511:  log likelihood = -165.14
## AIC=342.28   AICc=342.75   BIC=361.67
```

Warning: *the `ar1` parameter 1.3908 is the effect of* $(x_{t-1} - c)$ *on* x_t, *where* c *is the intercept of the model (`mean`).*

How to choose order (p, d, q) in practice

In practice, you have two choices, depending on your goal:

- to obtain quickly a good forecast, convenient if you have a lot of series to predict
  - let's use automatic function

```
auto.arima(uschange[,"Consumption"])
```

```
## Series: uschange[, "Consumption"]
## ARIMA(1,0,3)(1,0,1)[4] with non-zero mean
##
## Coefficients:
##           ar1     ma1     ma2     ma3     sar1    sma1
##       -0.3548  0.5958  0.3437  0.4111  -0.1376  0.3834
## s.e.   0.1592  0.1496  0.0960  0.0825   0.2117  0.1780
##
## sigma^2 = 0.3481:  log likelihood = -163.34
## AIC=342.67   AICc=343.48   BIC=368.52
```

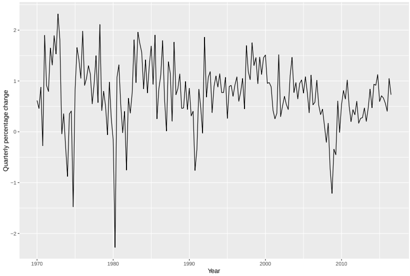
How to choose order (p, d, q) in practice

In practice, you have two choices, depending on your goal:

- to obtain a good forecast and an understanding of the model
  - let's start by differencing the series if needed, in order to obtain something visually stationary
  - look at the ACF and PACF plot ot identify possible models
  - take eventually into account knowledge on the series (knwon autocorrelation...)
  - estimate models and select the best one by AICc / AIC / BIC

Example: US consumption expenditure

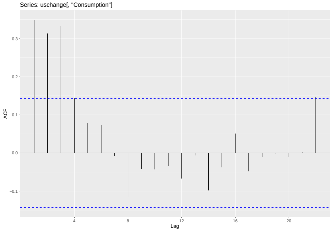
```
autoplot(uschange[,"Consumption"]) +
  xlab("Year") + ylab("Quarterly percentage change")
```



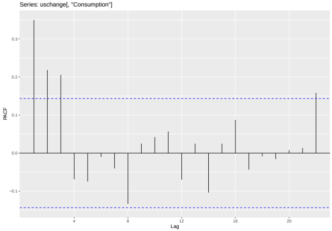
The series seems approximatively stationary. . .

Example: US consumption expenditure

```
ggAcf(uschange[,"Consumption"])
```




```
ggPacf(uschange[,"Consumption"])
```



May be an AR_3 or an MA_3

Example: US consumption expenditure

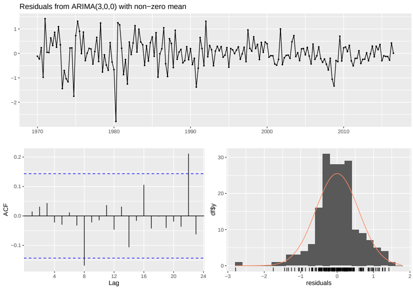
```r
Arima(uschange[,"Consumption"],order=c(3,0,0))
```

```
## Series: uschange[, "Consumption"]
## ARIMA(3,0,0) with non-zero mean
##
## Coefficients:
##          ar1     ar2     ar3    mean
##       0.2274  0.1604  0.2027  0.7449
## s.e.  0.0713  0.0723  0.0712  0.1029
##
## sigma^2 = 0.3494:  log likelihood = -165.17
## AIC=340.34   AICc=340.67   BIC=356.5
```

Example: US consumption expenditure

We check that residuals are un-correlated (LB test pvalue >0.05)

```
model=Arima(uschange[,"Consumption"],order=c(3,0,0))
checkresiduals(model)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(3,0,0) with non-zero mean
## Q* = 6.7407, df = 5, p-value = 0.2407
##
```

Example: US consumption expenditure

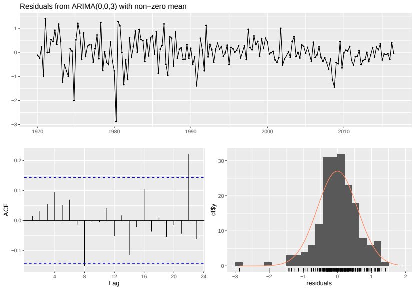
```r
Arima(uschange[,"Consumption"],order=c(0,0,3))
```

```
## Series: uschange[, "Consumption"]
## ARIMA(0,0,3) with non-zero mean
##
## Coefficients:
##          ma1     ma2     ma3    mean
##       0.2403  0.2187  0.2665  0.7473
## s.e.  0.0717  0.0719  0.0635  0.0739
##
## sigma^2 = 0.354:  log likelihood = -166.38
## AIC=342.76   AICc=343.09   BIC=358.91
```

Example: US consumption expenditure

The residuals are also uncorrelated

```
model=Arima(uschange[,"Consumption"],order=c(0,0,3))
checkresiduals(model)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,3) with non-zero mean
## Q* = 8.5791, df = 5, p-value = 0.1271
##
```

Example: US consumption expenditure

- AIC criterion slightly better for AR_3 (340.34) than for MA_3 (342.76)
- Note that AICc for AR_3 is better than for the model chosen by `auto.arima`! That is because all the possible models are not tested, but a stepwise search is used (see Hyndman, p245)

Forecasting

Once the model is selected, it will be use to forecast the future of the series.

For an AR_p:

- forecasting at horizon $h = 1$:

  $$\hat{x}_{n+1} = \hat{c} + \hat{a}_1 x_n + \ldots + \hat{a}_p x_{n+1-p}$$

  95% prediction interval can be obtained by: $\pm 1.96 \hat{x}_{n+1}$

- forceasting at horizon $h = 2$:

  $$\hat{x}_{n+2} = \hat{c} + \hat{a}_1 \hat{x}_{n+1} + \hat{a}_2 x_n + \ldots + \hat{a}_p x_{n+2-p}$$

- and so on...

Forecasting

Once the model is selected, it will be use to forecast the future of the series.

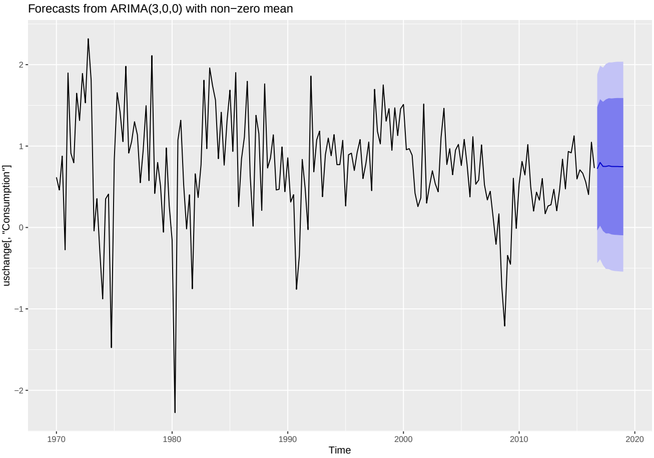
For an MA_q:

$$\hat{x}_{n+1} = \hat{c} + \hat{b}_1 \hat{\epsilon}_n + \ldots + \hat{b}_q \hat{\epsilon}_{n+1-q}$$

where $\hat{\epsilon}_n = x_n - \hat{x}_n$ and $\hat{\epsilon}_{n+1-q} = x_{n+1-q} - \hat{x}_{n+1-q}$

Example: US consumption expenditure

```
fit=Arima(uschange[,"Consumption"],order=c(3,0,0))
autoplot(forecast(fit,h=10))
```



Exercice: uschange

The following time series contain percentage changes in personal disposable income and unemployment rate for the US, from 1960 to 2016.

```r
autoplot(uschange[,c("Income","Unemployment")])
```

Choose an ARIMA model and forecast the income and unemployment rate for 2017 to 2020.

Seasonal ARIMA models

Backshift notation

A convenient notation for ARIMA models is **backshift notation**:

$$\begin{aligned} Bx_t &= x_{t-1} \\ B(Bx_t) &= B^2 x_t = x_{t-2} \end{aligned}$$

With this notation:

$$\begin{aligned} \Delta x_t &= (1-B)x_t = x_t - x_{t-1} \\ \Delta_T x_t &= (1-B^T)x_t = x_t - x_{t-T} \\ \Delta^d x_t &= (1-B)^d x_t \\ \Delta_T^d x_t &= (1-B^T)^d x_t \end{aligned}$$

Backshift notation

The backshift notation of an $ARIMA_{p,d,q}$ model is:

$$\underbrace{(1 - a_1 B - \ldots - a_p B^p)}_{AR_p} \underbrace{(1 - B)^d x_t}_{d \text{ differences}} = c + \underbrace{(1 + b_1 B - \ldots + b_q B^q)}_{MA_q} \epsilon_t$$

For instance, an $ARIMA_{1,1,1}$ without constant model is:

$$(1 - a_1 B)(1 - B) x_t = (1 + b_1 B)\epsilon_t$$

Rk: R uses a slightly different parametrization (see Hyndman p237)

Seasonal ARIMA models

A seasonnal ARIMA (SARIMA) model is formed by including additional seasonal terms in an ARIMA:

$$ARIMA \quad \underbrace{(p, d, q)}_{\text{non-seasonnal part}} \quad \underbrace{(P, D, Q)_T}_{\text{seasonnal part}}$$

where T is the period of the seasonal part.

Corresponding backshift notations is, for an $SARIMA_{(1,1,1)(1,1,1)_{12}}$ without constant model is:

$$(1 - a_1 B)(1 - a_2 B^{12})(1 - B)(1 - B^{12})x_t = (1 + b_1 B)(1 + b_2 B^{12})\epsilon_t$$

SARIMA properties

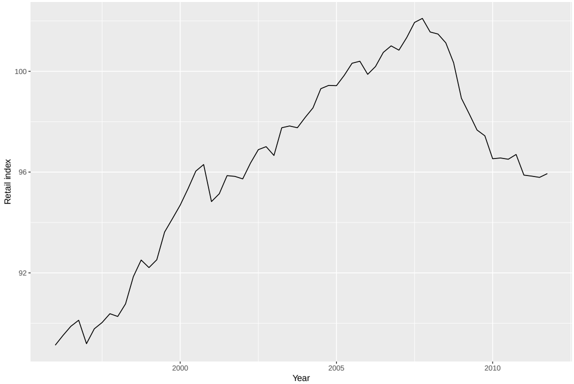
The seasonal part of an AR or MA model can be seen in the seasonal lags of the PACF and ACF.

For instance:

- an $SARIMA_{(0,0,0)(0,0,1)_{12}}$ will show:
  - a spike at lag 12 in the ACF, and no other significant spikes
  - exponential decay in the seasonal lags of the PACF (i.e. at lag 12, 24, 36...)
- an $SARIMA_{(0,0,0)(1,0,0)_{12}}$ will show:
  - a spike at lag 12 in the PACF, and no other significant spikes
  - expoenntial decay in the seasonal lags of the ACF

Example: European quaterly retail trade

```
autoplot(euretail) + ylab("Retail index") + xlab("Year")
```



This time series is clearly non stationary: trend an probably seasonal pattern of period 4 (*quaterly retrail trade...*)

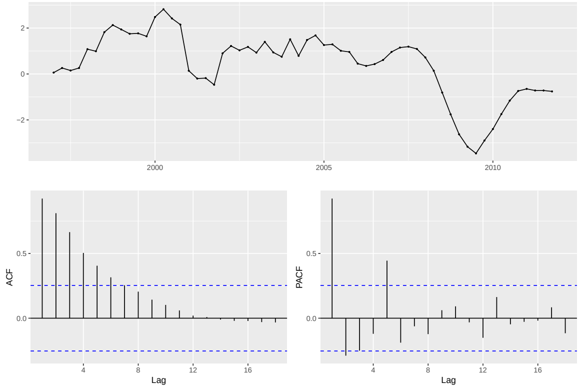
Example: European quaterly retail trade

Let's differenciate

```r
ggtsdisplay(diff(euretail,lag=4))
```

or equivalently

```r
euretail %>% diff(lag=4) %>% ggtsdisplay()
```

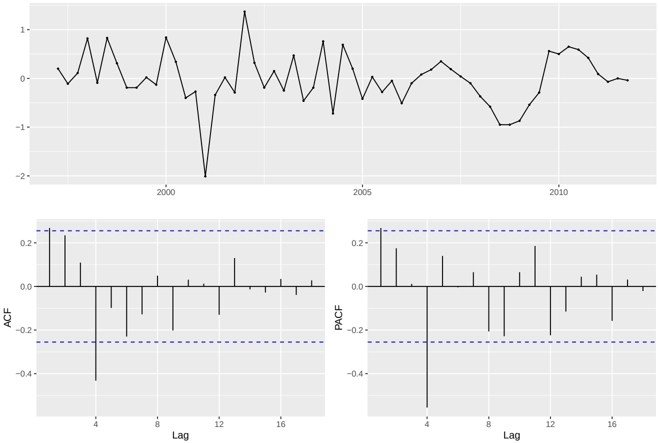


The linear decay of the ACF suggests that there is still a trend

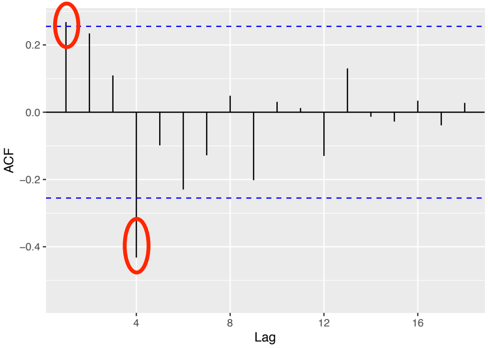
Example: European quaterly retail trade

Let's differenciate again

```
euretail %>% diff(lag=4) %>% diff()  %>% ggtsdisplay()
```



Example: European quaterly retail trade



- the slightly significant ACF at lag 1 suggests a non-seasonnal MA_1
- the significant ACF at lag 4 (the size of the period) suggests a seasonnal MA_1

Consequently we can try an $SARIMA_{(0,1,1)(0,1,1)_4}$.

Rk: similar reasoning with PACF suggests $SARIMA_{(1,1,0)(1,1,0)_4}$

Example: European quaterly retail trade

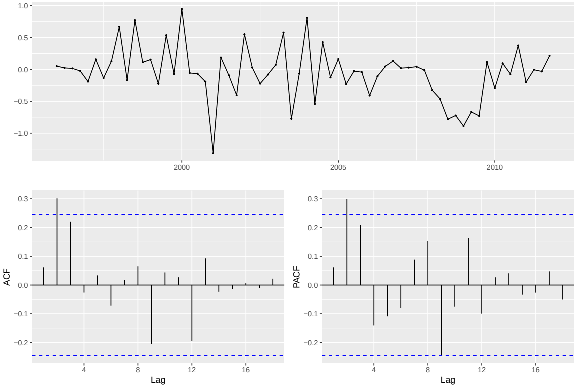
Let's estimate an $SARIMA_{(0,1,1)(0,1,1)_4}$

```
fit=Arima(euretail, order=c(0,1,1), seasonal=c(0,1,1))
```

Example: European quaterly retail trade

Let's have a look to the residual

```
fit %>% residuals() %>% ggtsdisplay()
```

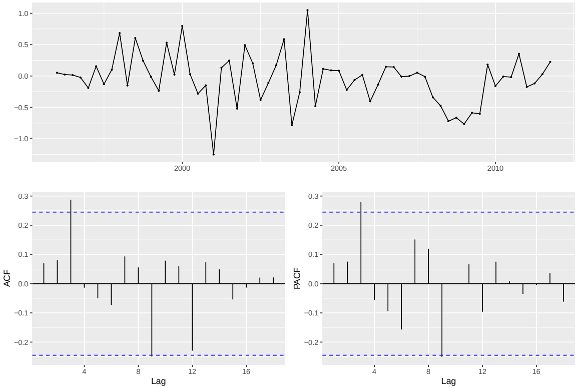


There is still significant ACF and PACF at lag 2. We can add some additional non-seasonal terms (for instance with $SARIMA_{(0,1,2)(0,1,1)_4}$)

Example: European quaterly retail trade

Let's estimate an $SARIMA_{(0,1,2)(0,1,1)_4}$

```
euretail %>%
  Arima(order=c(0,1,2), seasonal=c(0,1,1)) %>%
  residuals() %>% ggtsdisplay()
```

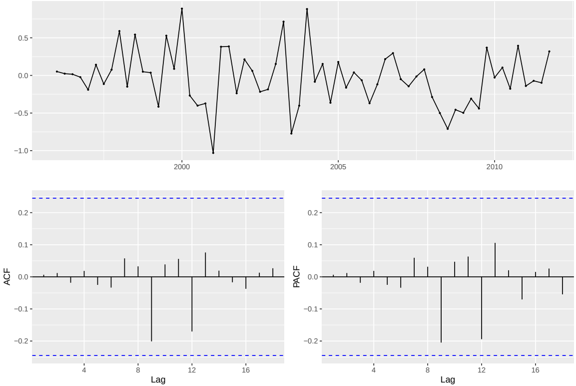


There is still significant ACF and PACF at lag 3.

Example: European quaterly retail trade

Let's estimate an $SARIMA_{(0,1,3)(0,1,1)_4}$

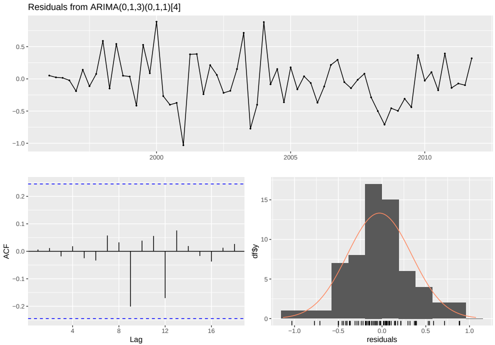
```
fit=Arima(euretail, order=c(0,1,3), seasonal=c(0,1,1))
fit %>% residuals() %>% ggtsdisplay()
```



Now the model seems to have capture all auto-correlations.

Example: European quaterly retail trade

```r
checkresiduals(fit)
```

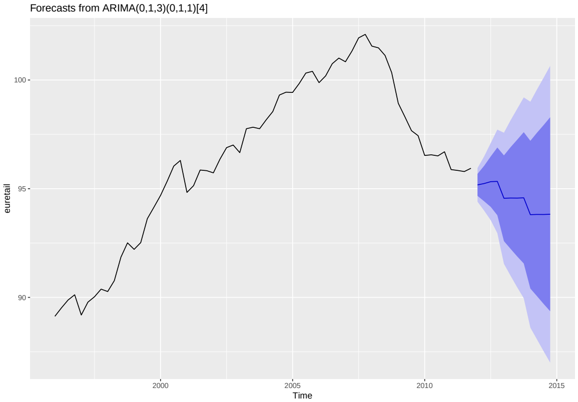



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,3)(0,1,1)[4]
## Q* = 0.51128, df = 4, p-value = 0.9724
##
## Model df: 4.   Total lags used: 8
```

Example: European quaterly retail trade

The model passes all checks: it is ready for forecasting

```
fit %>% forecast(h=12) %>% autoplot()
```



Exercice: San Francisco precipitation

San Fransisco precipitation from 1932, January to 1966, December are available here:
http://eric.univ-lyon2.fr/jjacques/Download/DataSet/sanfran.csv

- Try to improve your forecast obtained with exponential smoothing

Exercice: Varicella dataset

- Try to improve your forecast obtained with exponential smoothing

Heteroscedastic series

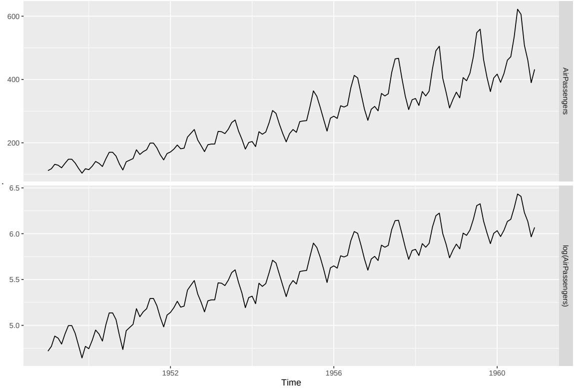
Stabilizing the variance

Previous models assume that the variance is stable in time.

For some series variance can decrease or increase.

Taking the log can help to stabilize it.

```
cbind(AirPassengers,log(AirPassengers)) %>%
autoplot(facets=TRUE)
```



Stabilizing the variance

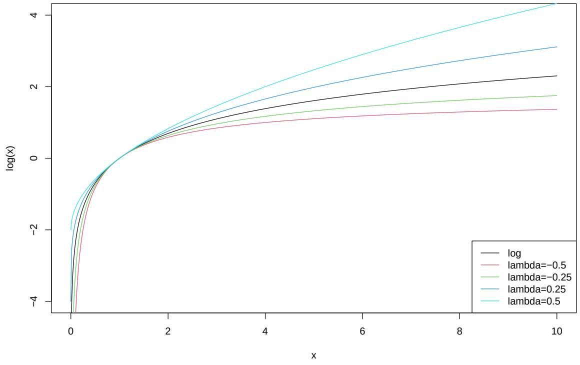
Rather than log transformation we can also use power transformation (square roots...).

A more general method for stabilizing the variance is to use Box-Cox transformation:

$$y_t = \begin{cases} \log(x_t) & \text{if } \lambda = 0 \\ (x_t^\lambda - 1)/\lambda & \text{if } \lambda \neq 0 \end{cases}$$

Box-Cox transformation

```
x=seq(0,10,0.01)
plot(x,log(x),type='l',ylim=c(-4,4))
lambda=-0.5;lines(x,(x^lambda-1)/lambda,col=2)
lambda=-0.25;lines(x,(x^lambda-1)/lambda,col=3)
lambda=0.25;lines(x,(x^lambda-1)/lambda,col=4)
lambda=0.5;lines(x,(x^lambda-1)/lambda,col=5)
legend('bottomright',col=1:5,lty=1,legend=c('log','lambda=-0.5',
```



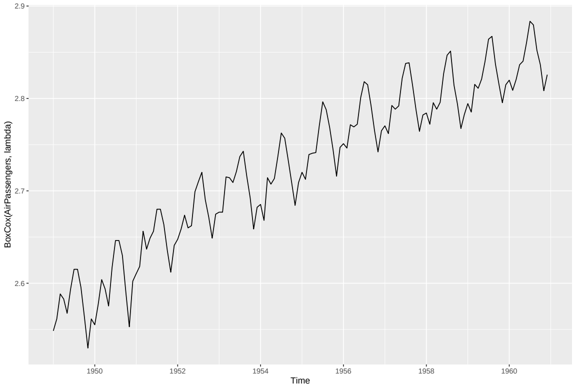
Stabilizing the variance

The `BoxCox.lambda()` function will choose a value of λ for you

```
(lambda=BoxCox.lambda(AirPassengers))
```

```
## [1] -0.2947156
```

```
autoplot(BoxCox(AirPassengers,lambda))
```



Stabilizing the variance

The BoxCox transformation is available as an option in the `hw` or `auto.arima` functions.

Automatic choice of λ is obtained by selecting: `lambda="auto"`.

ARCH and GARCH models

Such techniques allows to stabilize a variance which monotically increases or decreases.

For more complexe variations of the variance, as it can be in financial series, specific models for non constant variance exist:

- **ARCH: autoregressive conditional heteroscedasticity**
- and their generalization **GARCH**

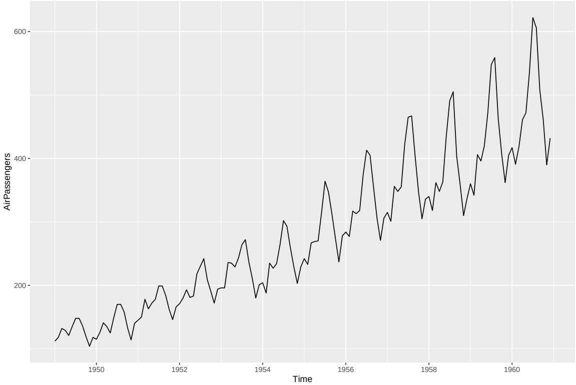
For more details refer to:

Brockwell P.J. et Davis R.A. Introduction to Time Series and Forecasting, Springer, 2001.

AirPassengers

Try to obtain the best model (exponential smoothing, SARIMA) for the AirPassengers data.

```
autoplot(AirPassengers)
```



The models will be evaluated on a test set made up of the last two years.

Time series forecasting

Machine learning methods

Julien JACQUES

Université Lumière Lyon 2

Neural network models

Other Machine Learning models

Neural network models

Neuron

A neuron is a *model*, with p features, which map the p *inputs* $x^1, \ldots, x^p$ to an *output* y :

$$y = g\left(\alpha_0 + \sum_{j=1}^{p} \alpha_j x^j\right)$$

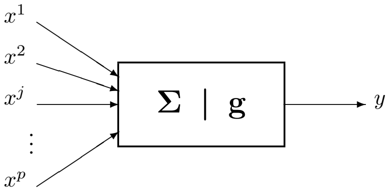


Figure 1: Neuron representation

- Σ: linear combination of inputs
- g: activation function

A specific neuron: linear model

One neuron with linear activation function $g(x) = x$ is the usual *linear model*:

$$y = \alpha_0 + \sum_{j=1}^{p} \alpha_j x^j$$

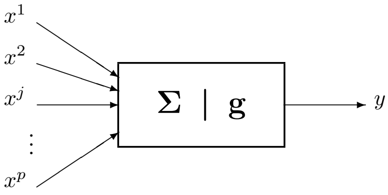


Figure 2: Neuron representation

Neural networks

A neural network is the association of several neurons, in a more or less complex graph, characterized by:

- its architecture (layer . . .)
- its complexity (number of neurons, presence of loops)
- activation functions
- the objective: supervised or unsupervised learning . . .

Multilayer perceptron

- A multilayer perceptron is made up of **layers**
- Layer: set of neurons without connection between them
- It has an input layer, an output layer, and **one or more hidden layers**
- The neurons are all connected at the input to each of the neurons of the previous layer and at the output to each of the neurons of the next layer

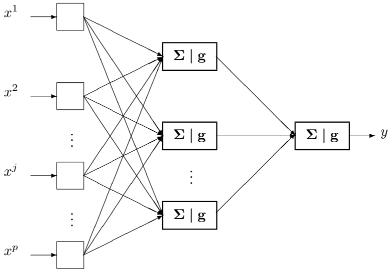


Figure 3: Multilayer perceptron with 1 hidden layer

Neural Network Auto-Regression (NNAR)

For **non seasonal** data:

- $NNAR_{p,k}$ model:
  - Inputs: lagged values of the time series $x_{t-1}, \ldots, x_{t-p}$
  - 1 hidden layer with k neurons
  - sigmoïd activation function
  - Rk: $NNAR_{p,0} = AR_p$

Neural Network Auto-Regression (NNAR)

For **non seasonal** data:

- $NNAR_{p,k}$ model:
  - Inputs: lagged values of the time series $x_{t-1}, \dots, x_{t-p}$
  - 1 hidden layer with k neurons
  - sigmoïd activation function
  - Rk: $NNAR_{p,0} = AR_p$

For **seasonal** data (of period T), we add lagged values from the same season as last observed values:

- $NNAR_{(p,P,k)_T}$ model:
  - Inputs: lagged values of the time series $x_{t-1}, x_{t-2}, \dots, x_{t-p}, x_{t-T}, x_{t-2T}, \dots, x_{t-PT}$
  - 1 hidden layer with k neurons
  - sigmoïd activation function
  - Rk: $NNAR_{(p,P,0)_T} = SARIMA_{(p,0,0)(P,0,0)_T}$

`nnetar` function

Estimation of an $NNAR_{(p,P,k)_T}$ with the `forecast` package:

```
nnetar(x, p, P, size=k)
```

- if `p` not specified, it is chosen automatically by minimizing AIC of a linear AR_p model
- if `P` not specified, $P = 1$ is chosen
- if `k` not specified, $k = (p + P + 1)/2$ is chosen

Other options:

- `xreg` allows to add external regressors
- `lambda` allows to use Box-Cox transformation

Neural Network Auto-Regression (NNAR)

- Advantage over a linear model (AR_p):
  - more flexible, modeling non-linear relation
- Dis-advantage over a linear model (AR_p):
  - none well-defined sochastic model -> prediction interval not direct (need boostrap simulations, option `PI=TRUE`)
  - not possible to *integrate* differecing

More Neural Network

More sophisticated neural networks for time series as Recurrent Neural Network (but also LSTM, GRU...).

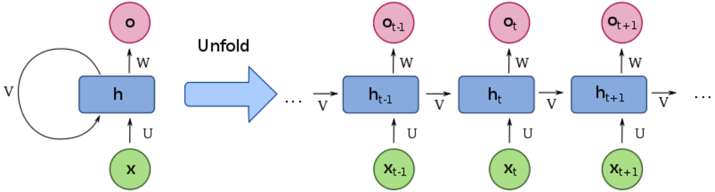


Figure 4: Neuron representation

RNN in R

RNN are implemented in the Keras lib. for Python.

We can used it through the `keras` R package

```
library(keras)
```

The idea is to split the whole time series into sub-series.

For instance for weekly data:

- $x_7 = f(x_6, \ldots, x_1)$
- $x_{14} = f(x_{13}, \ldots, x_8)$
- and so on
- $x_{n-7} = f(x_{n-8}, \ldots, x_{n-13})$

A neural network is learned to estimate f, and then used to forecast x_{n+1}

- $\hat{x}_{n+1} = \hat{f}(x_n, \ldots, x_{n-6})$

More Neural Network

To my experience, such models:

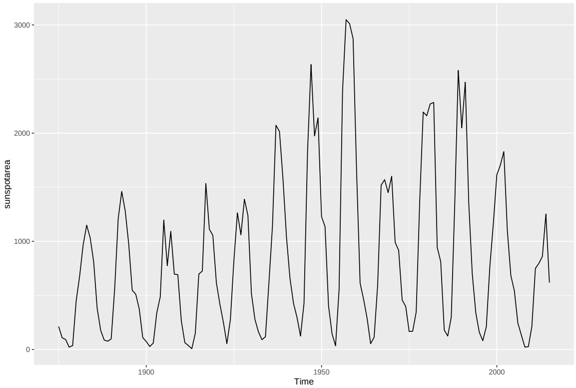
- are efficient when the series is hard to forecast, with no evident model behind and when usual model are not efficient
- need time series of large sizes

Note that you can use RNN directly from R thanks to the `keras` package.

If you want to make your own opinion, let have a look for instance to: https://www.r-bloggers.com/2020/05/time-series-with-arima-and-rnn-models/

Example: sunspots

```
autoplot(sunspotarea)
```



No seasonal but **cyclic pattern** $\Rightarrow$ can not be modelized by usual linear *SARIMA* models

Example: sunspots

$NNAR_{p,k}$ model estimation, with automatic choice of p and k:

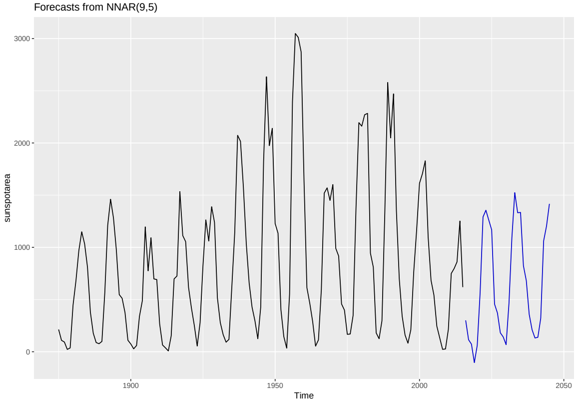
```
fit=nnetar(sunspotarea)
print(fit)
```

```
## Series: sunspotarea
## Model:  NNAR(9,5)
## Call:   nnetar(y = sunspotarea)
##
## Average of 20 networks, each of which is
## a 9-5-1 network with 56 weights
## options were - linear output units
##
## sigma^2 estimated as 11308
```

Example: sunspots

Forecasting for next 30 years:

```
autoplot(forecast(fit,h=30))
```



- asymetric cyclicity as been modelled well

Exercice: San Francisco precipitation

San Fransisco precipitation from 1932 to 1966 are available here:
http://eric.univ-lyon2.fr/jjacques/Download/DataSet/sanfran.dat

- Try to improve your forecasts obtained with exponential smoothing and SARIMA models with neural network models

Exercices:

Try to improve all your previous forecast with NN ! . . . good luck ;-)

Other Machine Learning models

Data prepration

- any machine learning model can be tune once we correctly split the time series $x_1, \ldots, x_n$ into input|output data set:

$$
\begin{array}{ccc|c}
 & \textit{inputs} & & \textit{output} \\
x_1 & \cdots & x_T & x_{T+1} \\
 & & & \\
x_{n-T-1} & \cdots & x_{n-2} & x_{n-1} \\
x_{n-T} & \cdots & x_{n-1} & x_n
\end{array}
$$

- then, we just have to learn any function f:

$$x_n = f(x_{n-T}, \ldots, x_{n-1})$$

- f can be Random Forest, Gradient boosting, SVM, LM...

Data prepration for San Francisco data set

```
data=scan(file="data/sanfran.csv",skip=1)
sanfran<-ts(data,start=c(1932,1),end=c(1966,12),freq=12)
library(forecast)
sanfran_train=window(sanfran,start=c(1932,1),end=c(1963,12))
sanfran_test=window(sanfran,start=c(1964,1),end=c(1966,12))
data=as.vector(sanfran_train)[1:13]
for (i in 1:(length(as.vector(sanfran_train))-13)){
data=rbind(data,as.vector(sanfran_train)[(i+1):(i+13)])
}
print(head(sanfran))
```

```
##         Jan   Feb   Mar   Apr   May   Jun
## 1932 16.26 29.46 18.03 24.13 22.35 22.10
```

```
print(data[1:2,1:6])
```

```
##        [,1]  [,2]  [,3]  [,4]  [,5]  [,6]
## data 16.26 29.46 18.03 24.13 22.35 22.10
##      29.46 18.03 24.13 22.35 22.10 12.95
```

Random Forest

We fit the model

```
library(randomForest)
fitRF=randomForest(x=data[,-13], y=data[,13])
```

And then sequentially forecast the next 36 values

```
pred=rep(NULL,36)
newdata=tail(sanfran_train,12)
for (t in 1:36){
  pred[t]=predict(fitRF,newdata=newdata)
  newdata=c(newdata[-1],pred[t])
}
prevRF=ts(pred,start=c(1964,1),end=c(1966,12),frequency = 1
```

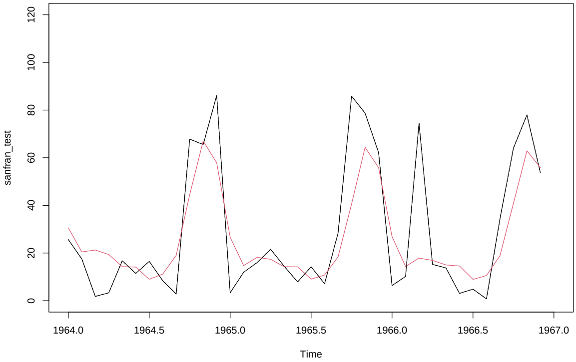
Random Forest

Forecasting results

```
print(sqrt(mean((prevRF-sanfran_test)^2)))
```

```
## [1] 16.91693
```

```
plot(sanfran_test,xlim=c(1964,1967),ylim=c(0,120))
lines(sanfran_test,lty=2)
lines(prevRF,col=2)
```



Gradient Boosting

Random forest is an aggregation of independent decision/regression trees built on bootstrap samples (*bagging*).

Boosting method consists of aggregating non independent models (typ. trees):

- a first model is trained on the whole data set
- a second model is then trained on the same data set, but by given a larger weight to the bad predicted observations
- and so on...

Several boosting model exists: AdaBoost, LPBoost, XGBoost, GradientBoost, BrownBoost...

Gradient Boosting

Here we use the probably most known boosting method: XGBoost

```
library(xgboost)
```

Many parameters have to be tune, among which:

- max_depth: maximum depth of a tree
- 0<eta<1: the learning rate (scale the contribution of each tree)
- nrounds: maximum number of boosting iterations.
- objective: learning loss (*reg:squarederror* or *reg:squaredlogerror* for regression).

```
model<- xgboost(data = data[,1:12], label = data[,13],
                max_depth = 10, eta = .5, nrounds = 100,
                nthread = 2, objective = "reg:squarederror")
```

```
## [1]  train-rmse:21.074956
## [2]  train-rmse:13.236779
## [3]  train-rmse:8.915509
## [4]  train-rmse:6.398479
## [5]  train-rmse:4.753323
## [6]  train-rmse:3.674946
```

Gradient Boosting

Now we will do forecasting sequentially

```
pred=rep(NULL,36)
newdata=tail(sanfran_train,12)
for (t in 1:36){
  pred[t]=predict(model,matrix(newdata,1,12))
  newdata=c(newdata[-1],pred[t])
}
prevXGBOOST=ts(pred,start=c(1964,1),end=c(1966,12),frequency=12)
```

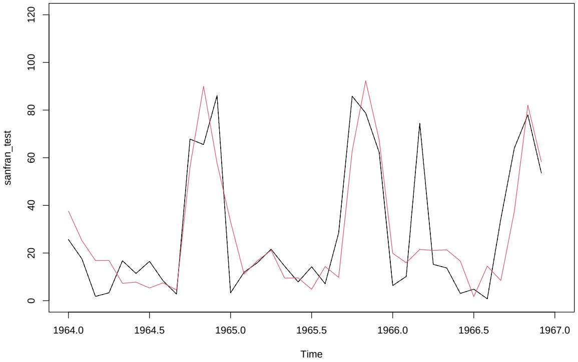
Gradient Boosting

Forecasting results

```
print(sqrt(mean((prevXGBOOST-sanfran_test)^2)))
```

```
## [1] 16.19329
```

```
plot(sanfran_test,xlim=c(1964,1967),ylim=c(0,120))
lines(sanfran_test,lty=2)
lines(prevXGBOOST,col=2)
```



SVM

Model fitting and sequential forecasting

```
library(e1071)
fitSVM=svm(x=data[,-13], y=data[,13])
pred=rep(NULL,36)
newdata=tail(sanfran_train,12)
for (t in 1:36){
  pred[t]=predict(fitSVM,newdata=matrix(newdata,1,12))
  newdata=c(newdata[-1],pred[t])
}
prevSVM=ts(pred,start=c(1964,1),end=c(1966,12),frequency =
```

SVM

Forecasting results

```
print(sqrt(mean((prevSVM-sanfran_test)^2)))
```

```
## [1] 17.53055
```

```
plot(sanfran_test,xlim=c(1964,1967),ylim=c(0,120))
lines(sanfran_test,lty=2)
lines(prevSVM,col=6)
```

LM

Model fitting and sequential forecasting

```
data=data.frame(data)
colnames(data)=c('x1','x2','x3','x4','x5','x6','x7','x8','x
fitLM=lm(y~.,data=data)
pred=rep(NULL,36)
newdata=data.frame(matrix(tail(sanfran_train,12),1,12))
colnames(newdata)=c('x1','x2','x3','x4','x5','x6','x7','x8
for (t in 1:36){
  pred[t]=predict(fitLM,newdata=newdata)
  newdata=data.frame(c(newdata[-1],pred[t]))
  colnames(newdata)=c('x1','x2','x3','x4','x5','x6','x7','x
}
prevLM=ts(pred,start=c(1964,1),end=c(1966,12),frequency = 1
```

Rk: be carreful, `lm` assume i.i.d. observation, what is wrong here

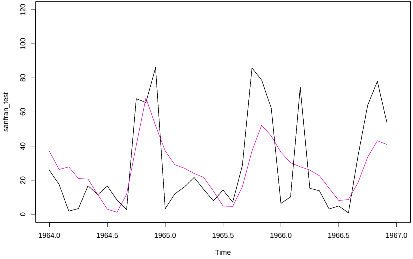
LM

Forecasting results

```r
print(sqrt(mean((prevLM-sanfran_test)^2)))
```

```
## [1] 20.49199
```

```r
plot(sanfran_test,xlim=c(1964,1967),ylim=c(0,120))
lines(sanfran_test,lty=2)
lines(prevLM,col=6)
```



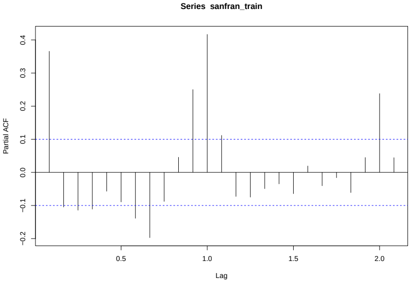
Which inputs ?

In the previous application, we used the 12 previous observations.

$$x_n = f(x_{n-1}, \ldots, x_{n-12})$$

Is-it a good idea?

```r
pacf(sanfran_train)
```



Looking at the PACF can advice us to use:

$$x_n = f(x_{n-1}, x_{n-8}, x_{n-11}, x_{n-12}, x_{n-24})$$

New data preparation

```r
data=as.vector(sanfran_train)[c(1,13,14,17,24,25)]
for (i in 1:(length(as.vector(sanfran_train))-25)){
data=rbind(data,as.vector(sanfran_train)[i+c(1,13,14,17,24,25)])
}
print(str(data))
```

```
##  num [1:360, 1:6] 16.3 29.5 18 24.1 22.4 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:360] "data" "" "" "" ...
##   ..$ : NULL
## NULL
```

RF with new inputs

```r
library(randomForest)
fitRF=randomForest(x=data[,-6], y=data[,6])
```

And then sequentially forecast the next 36 values

```r
pred=rep(NULL,36)
newdata=sanfran_train[length(sanfran_train)-c(1,8,11,12,24)]
newsf=sanfran_train
for (t in 1:36){
  pred[t]=predict(fitRF,newdata=newdata)
  newsf=c(newsf,pred[t])
  newdata=newsf[length(newsf)-c(1,8,11,12,24)]
}
prevRF=ts(pred,start=c(1964,1),end=c(1966,12),frequency = 12)
```

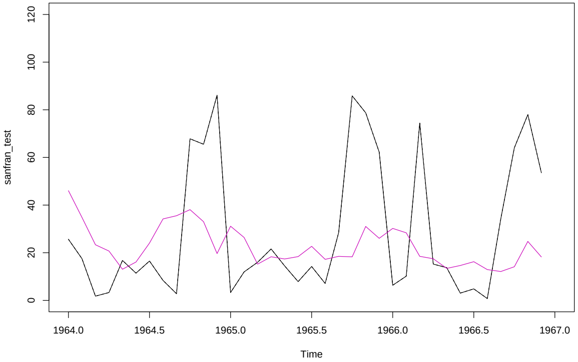
RF with new inputs

Forecasting results

```
print(sqrt(mean((prevRF-sanfran_test)^2)))
```

```
## [1] 29.21335
```

```
plot(sanfran_test,xlim=c(1964,1967),ylim=c(0,120))
lines(sanfran_test,lty=2)
lines(prevRF,col=6)
```



To go further

- forecasting is done sequentially : $\hat{x}_{n+2}$ uses $\hat{x}_{n+1}$.
- improvement would be to:
  - forecast directly $\hat{x}_{n+2}$
  - simultaneously forecast $(\hat{x}_{n+1}, \hat{x}_{n+2})$
- for this machine learning model with multivariate output have to be used
  - but not so much models are developed for this task

Multivariate LM

We implement a LM model with multivariate output.

We consider 1 year (12 obs) as output, and 2 years as inputs (24 obs).

```
data=as.vector(sanfran_train)[1:36]
for (i in 1:(length(as.vector(sanfran_train))-36)){
  data=rbind(data,as.vector(sanfran_train)[(i+1):(i+36)])
}
data=data.frame(data)
for (i in 1:36) colnames(data)[i]=paste('x',i,sep='')
fitLM=lm(cbind(x36,x35,x34,x33,x32,x31,x30,x29,x28,x27,x26,x25)~.,data=data)
```

Multivariate LM

Sequential forecasting by year

```
pred=rep(NULL,36)
newdata=data.frame(matrix(tail(sanfran_train,24),1,24))
for (i in 1:24) colnames(newdata)[i]=paste('x',i,sep='')
pred[1:12]=predict(fitLM,newdata=newdata)
for (j in 1:2){
  newdata=data.frame(c(newdata[-(1:12)],pred[(12*(j-1)+1):(12*j)]))
  for (i in 1:24) colnames(newdata)[i]=paste('x',i,sep='')
  pred[(12*(j)+1):(12*(j+1))]=predict(fitLM,newdata=newdata)
}
prevMLM=ts(pred,start=c(1964,1),end=c(1966,12),frequency = 12)
```

Multivariate LM

Forecasting results

```
print(sqrt(mean((prevMLM-sanfran_test)^2)))
```

```
## [1] 26.66568
```

```
plot(sanfran_test,xlim=c(1964,1967),ylim=c(0,120))
lines(sanfran_test,lty=2)
lines(prevMLM,col=6)
```

Multivariate LM

We can change by sequentially forecasting months by months and not year by year

```
pred=rep(NULL,36)
newdata=data.frame(matrix(tail(sanfran_train,24),1,24))
for (i in 1:24) colnames(newdata)[i]=paste('x',i,sep='')
pred[1:12]=predict(fitLM,newdata=newdata)
for (j in 1:35){
  newdata=data.frame(c(newdata[-1],pred[j]))
  for (i in 1:24) colnames(newdata)[i]=paste('x',i,sep='')
  pred[j+1]=predict(fitLM,newdata=newdata)[1]
}
prevMLM=ts(pred,start=c(1964,1),end=c(1966,12),frequency = 12)
```

Multivariate LM

Forecasting results

```
print(sqrt(mean((prevMLM-sanfran_test)^2)))
```

```
## [1] 33.72657
```

```
plot(sanfran_test,xlim=c(1964,1967),ylim=c(0,120))
lines(sanfran_test,lty=2)
lines(prevMLM,col=6)
```

Time series forecasting

Multivariate time series

Julien JACQUES

Université Lumière Lyon 2

Time series regression models

Dynamic regression model

Grouped time series models: VAR models

Time series regression models

Time series regression models

Let assume that you want to explain your serie x_t according to k features $z_{1t}, \ldots, z_{kt}$:

$$x_t = c + \beta_1 z_{1t} + \ldots + \beta_k z_{kt} + \epsilon_t.$$

Usual linear regression model assume that the error ϵ_t are independent and identically distributed according: $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$.

Such model can be estimated with the usual `lm` function or with

```
?tslm
```

Time series regression models

In addition to the effect of external features, times series often contain:

- a **trend**. A linear model including a linear trend can be written:

$$x_t = \underbrace{c + \beta_0 t}_{\text{trend}} + \underbrace{\beta_1 z_{1t} + \ldots + \beta_k z_{kt}}_{\text{covariates}} + \epsilon_t.$$

- a **seasonal pattern** of period T. Corresponding regression model is:

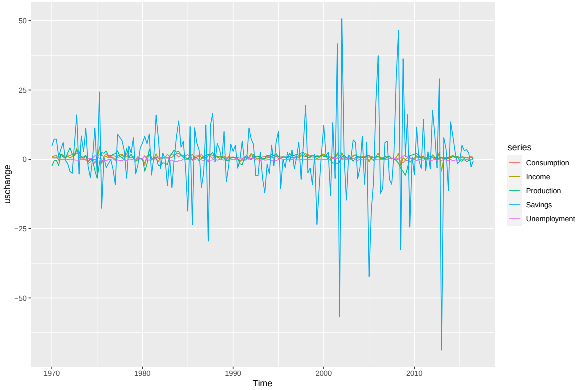
$$x_t = \underbrace{c + \beta_0 t}_{\text{trend}} + \underbrace{\delta_2 d_{2t} + \ldots + \delta_T d_{Tt}}_{\text{seasonal effect}} + \underbrace{\beta_1 z_{1t} + \ldots + \beta_k z_{kt}}_{\text{covariates}} + \epsilon_t.$$

  where $d_{2t}, \ldots, d_{Tt}$ are the dummy notations for the $T-1$ *days* of the period: $d_{jt} = 1$ if $j = t$ and 0 otherwise. Note that the effect of the first day d_{1t} is included in the intercept, so d_{jt} is the additional effet of day j in comparison with day 1.

Time series regression models

Let's go back to the uschange time series

```r
library(fpp2)
autoplot(uschange)
```



Time series regression models

We want to predict Consumption using other times series

```
fit=tslm(Consumption~Income+Production+Unemployment+Savings,data=uschange)
summary(fit)
```

```
##
## Call:
## tslm(formula = Consumption ~ Income + Production + Unemployment +
##     Savings, data = uschange)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.88296 -0.17638 -0.03679  0.15251  1.20553
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.26729    0.03721   7.184 1.68e-11 ***
## Income        0.71449    0.04219  16.934  < 2e-16 ***
## Production    0.04589    0.02588   1.773   0.0778 .
## Unemployment -0.20477    0.10550  -1.941   0.0538 .
## Savings      -0.04527    0.00278 -16.287  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3286 on 182 degrees of freedom
## Multiple R-squared:  0.754,  Adjusted R-squared:  0.7486
## F-statistic: 139.5 on 4 and 182 DF,  p-value: < 2.2e-16
```

Time series regression models

We can add a trend and a seasonnal pattern

```
fit=tslm(Consumption~Income+Production+Unemployment+Savings+trend+season,data=uschange)
summary(fit)
```

```
##
## Call:
## tslm(formula = Consumption ~ Income + Production + Unemployment +
##     Savings + trend + season, data = uschange)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.88653 -0.15100 -0.00713  0.14232  1.10178
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.4535889  0.0717294   6.324    2e-09 ***
## Income        0.7093775  0.0419836  16.897   <2e-16 ***
## Production    0.0389018  0.0264104   1.473   0.1425
## Unemployment -0.2396921  0.1096766  -2.185   0.0302 *
## Savings      -0.0450622  0.0027690 -16.274   <2e-16 ***
## trend        -0.0010066  0.0004616  -2.181   0.0305 *
## season2      -0.1294052  0.0669461  -1.933   0.0548 .
## season3      -0.0602444  0.0671966  -0.897   0.3712
## season4      -0.1495544  0.0675787  -2.213   0.0282 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.322 on 178 degrees of freedom
## Multiple R-squared:  0.769,  Adjusted R-squared:  0.7586
## F-statistic: 74.06 on 8 and 178 DF,  p-value: < 2.2e-16
```

Feature selection

As for any multivariate regression model, we have to select which are the best features to include in the model.

Comparison between models can be done with usual criteria (AIC, AICc, BIC, adjusted R^2, ...)

Those criterion can be obtained as follows:

```
CV(fit)
```

```
##          CV          AIC         AICc          BIC        AdjR2
##   0.1141794 -413.0495738 -411.7995738 -380.7384876    0.7585933
```

Feature selection

In the previous model we have seen that `Production` is not significant in the model.

We can remove it and compare the model to the previous one

```
CV(fit)
```

```
##           CV          AIC         AICc          BIC        AdjR2
##    0.1141794 -413.0495738 -411.7995738 -380.7384876    0.7585933
```

```
fit2=tslm(Consumption~Income+Unemployment+Savings+trend+season,data=usc
CV(fit2)
```

```
##           CV          AIC         AICc          BIC        AdjR2
##    0.1136653 -412.7840112 -411.7670620 -383.7040336    0.7570159
```

There is no evident difference between these models (better CV, AIC, AICc and BIC, but worse AdjR^2).

Feature selection

Stepwise selection procedure should be used to correctly select the best set of features.

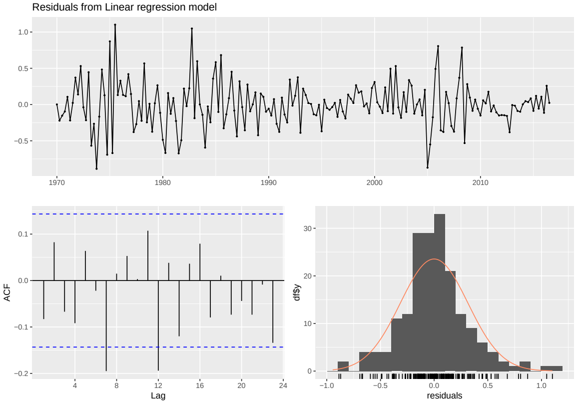
To the best of my knowledge, such procedures are not available for the `tslm` function.

But you can use the `lm` function (with time and season as covariates), and use usual stepwise function for `lm` as `step` or `stepAIC`.

Checking the residuals

Usual checking of linear model can/should be done:

```
checkresiduals(fit,test=FALSE,plot=TRUE)
```



Checking the residuals

including test of non correlation of the residuals

```
checkresiduals(fit,test='LB',plot=FALSE)
```

```
##
##  Ljung-Box test
##
## data:  Residuals from Linear regression model
## Q* = 13.5, df = 8, p-value = 0.09577
##
## Model df: 0.   Total lags used: 8
```

Here the residual are correlated, which means that this regression model (which assumes independent residuals) is not appropriated.

Dynamic regression model

Dynamic regression model

We saw in the previous model:

$$x_t = c + \beta_1 z_{1t} + \ldots + \beta_k z_{kt} + \epsilon_t.$$

that the residuals ϵ_t are not independent.

Dynamic regression model modelizes the **residuals with an** $ARIMA_{p,d,q}$ **model**

Dynamic regression model

We saw in the previous model:

$$x_t = c + \beta_1 z_{1t} + \ldots + \beta_k z_{kt} + \epsilon_t.$$

that the residuals ϵ_t are not independent.

Dynamic regression model modelizes the **residuals with an** $ARIMA_{p,d,q}$ **model**

The choice of the orders p, d, q can be done by examining the residuals or automatically with the `auto.arima` function.

Dynamic regression model

Let's try for instance an $ARIMA_{1,0,2}$:

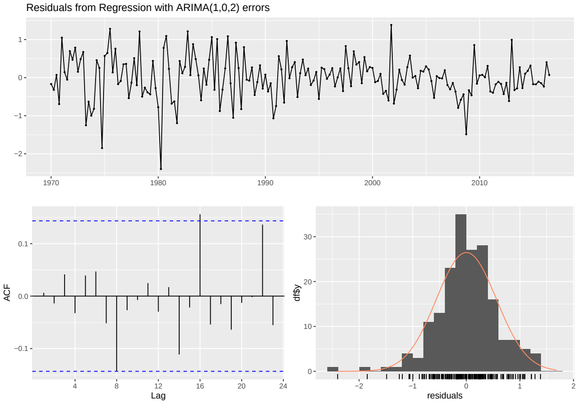
```
fit=Arima(uschange[,'Consumption'],xreg=uschange[,'Income'],order=c(1,0,2))
summary(fit)
```

```
## Series: uschange[, "Consumption"]
## Regression with ARIMA(1,0,2) errors
##
## Coefficients:
##          ar1      ma1     ma2  intercept    xreg
##       0.6922  -0.5758  0.1984     0.5990  0.2028
## s.e.  0.1159   0.1301  0.0756     0.0884  0.0461
##
## sigma^2 = 0.3219:  log likelihood = -156.95
## AIC=325.91   AICc=326.37   BIC=345.29
##
## Training set error measures:
##                       ME      RMSE       MAE     MPE     MAPE      MASE
## Training set 0.001714366 0.5597088 0.4209056 27.4477 161.8417 0.6594731
##                     ACF1
## Training set 0.006299231
```

Dynamic regression model

We can now check the residuals:

```r
checkresiduals(fit,test=FALSE)
```



Dynamic regression model

and test their autocorrelation:

```
checkresiduals(fit,plot=FALSE)
```

```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(1,0,2) errors
## Q* = 5.8916, df = 5, p-value = 0.3169
##
## Model df: 3.   Total lags used: 8
```

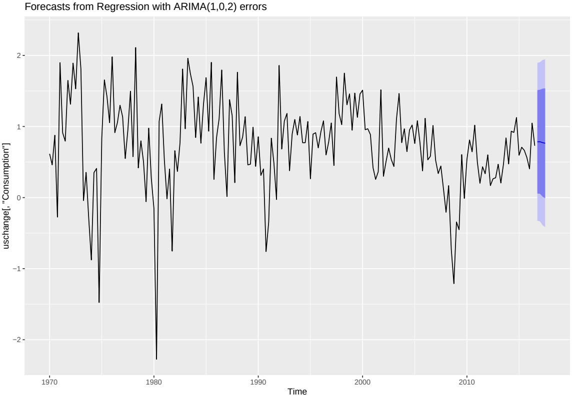
It seems that all the auto-correlations of the residuals have been modelled with this model.

Dynamic regression model

The model being validated, we can forecast the future !

Warning: since we use covariate, we should have the value of the covariate for the future !

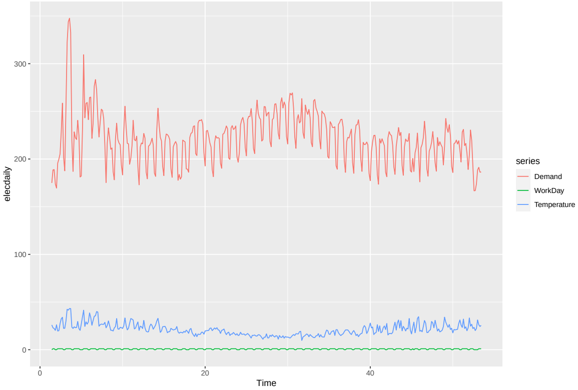
```
autoplot(forecast(fit,xreg=rep(mean(uschange[,2]),4)))
```



Exercice: Electricty demand

Try to find the best model for forecasting electricity (using or not covariates) demand

```
autoplot(elecdaily)
```



Forecasting efficiency will be evaluated on the last 7 days, and will assume that we dispose of a forecasting of the Temperature for the next 7 days (WorkDay are of course also known).

Grouped time series models: VAR models

VAR models

- Data : bivariate time series $(X_{1,t}, X_{2,t})$.
- We want to **forecast both time series**
- The idea is that each time series can help in forecasting the other one.
- **Vectoriel Auto-Regressive** model VAR_1 :

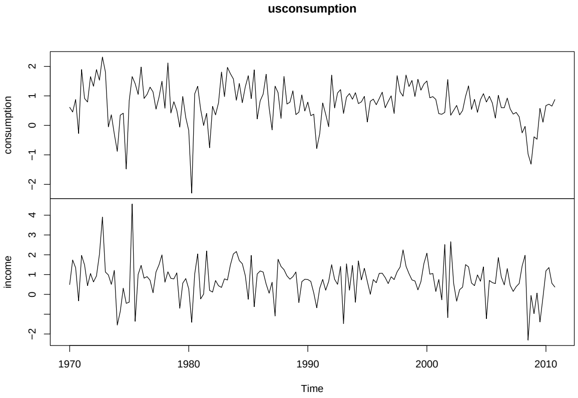
$$
\begin{aligned}
X_{1,t} &= c_1 + \epsilon_{1,t} + a_{1,1}X_{1,t-1} + a_{1,2}X_{2,t-1}, \\
X_{2,t} &= c_2 + \epsilon_{2,t} + a_{2,1}X_{1,t-1} + a_{2,2}X_{2,t-1},
\end{aligned}
$$

- High order model can be also considered VAR_p

Data usconsumption

We will work with data usconsumption

```
library(fpp)
data(usconsumption)
plot(usconsumption)
```

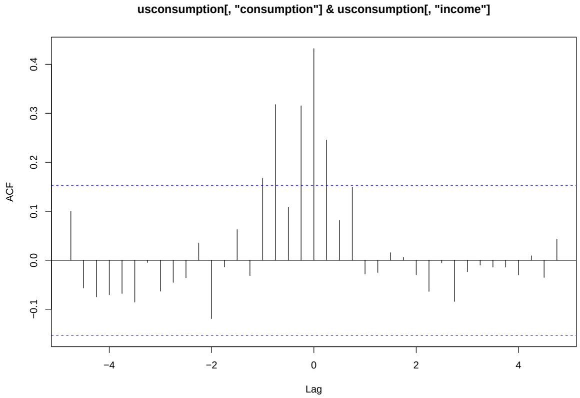


Cross correlation

We can compute the cross-covariance (or cross-correlation) between these two time series:

$$\widehat{ccov}_n(h) = \frac{1}{n-h} \sum_{t=1}^{n-h} (x_t - \bar{x}_n)(y_{t+h} - \bar{y}_n),$$

```
ccf(usconsumption[,"consumption"],usconsumption[,"income"])
```



Cross correlation test

It is possible to test the significancy of the cross correlation with the Dalla et al. 2020) procedure:

```r
library(testcorr)
cc.test(usconsumption[,"consumption"],usconsumption[,"income"],max.lag = 1,
        plot = FALSE,)
```

```
##
## Tests for zero cross-correlation of x and y
##
## | Lag|    CC|  Stand. CB(95%)|  Robust CB(95%)| Lag|     t| p-value| t-tilde
## |---:|-----:|---------------:|---------------:|---:|-----:|-------:|-------:
## |  -1| 0.315| (-0.153, 0.153)| (-0.175, 0.175)|  -1| 4.037|   0.000|   3.536
## |   0| 0.432| (-0.153, 0.153)| (-0.240, 0.240)|   0| 5.533|   0.000|   3.528
## |   1| 0.246| (-0.153, 0.153)| (-0.189, 0.189)|   1| 3.145|   0.002|   2.550
```

Data usconsumption

We choose two last years as test data

```
us_app=ts(usconsumption[1:156,],start=c(1970,1),end=c(2008,4),frequency = 4)
us_test=ts(usconsumption[157:164,],start=c(2009,1),end=c(2010,4),frequency = 4)
```

VAR_p model

Function VARselect allows to choose the best VAR_p model according to some criteria (among which AIC), for $1 \leq p \leq lag.max$

```
library(vars)
VARselect(us_app, lag.max=8, type="const",season=4)
```

The option `type` allows to introduce a trend (`"const"`, `"trend"`, `"both"`, `"none"`), the option `season` for a seasonal pattern and `exogen` for external covariates.

VAR_p model

```r
library(vars)
VARselect(us_app, lag.max=8, type="const")
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      5      1      1      5
##
## $criteria
##                 1         2          3          4          5
## AIC(n) -1.2314131 -1.217775 -1.2505936 -1.2505388 -1.2690326
## HQ(n)  -1.1820445 -1.135494 -1.1354000 -1.1024328 -1.0880142
## SC(n)  -1.1099045 -1.015261 -0.9670735 -0.8860130 -0.8235011
## FPE(n)  0.2918831  0.295903  0.2863752  0.2864365  0.2812579
##                 7          8
## AIC(n) -1.1932426 -1.1788984
## HQ(n)  -0.9463992 -0.8991427
## SC(n)  -0.5856995 -0.4903497
## FPE(n)  0.3036602  0.3082446
```

VAR_p model

Estimation of an VAR_5

```
var <- VAR(us_app, p=5,type = "const",season = 4,exogen=NULL)
summary(var)
```

```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: consumption, income
## Deterministic variables: const
## Sample size: 151
## Log Likelihood: -306.307
## Roots of the characteristic polynomial:
## 0.7553 0.7553 0.7247 0.7247 0.7229 0.6198 0.6198 0.5715 0.571
## Call:
## VAR(y = us_app, p = 5, type = "const", season = 4L, exogen =
##
##
## Estimation results for equation consumption:
## ==========================================
## consumption = consumption.l1 + income.l1 + consumption.l2 + i
```

VAR_p model

We check that the residual are a white noise

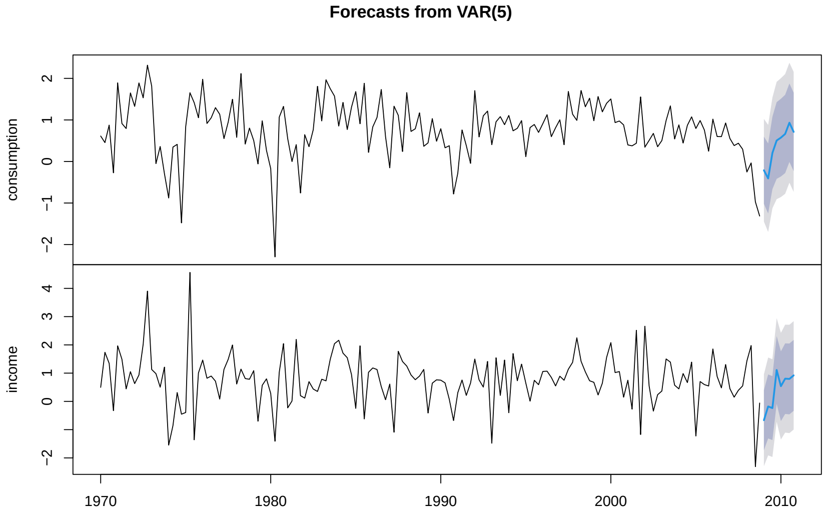
```r
serial.test(var, lags.pt=10, type="PT.asymptotic")
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object var
## Chi-squared = 14.26, df = 20, p-value = 0.8171
```

VAR_p model

Forecasting

```
fcst <- forecast(var,h=8)
plot(fcst, xlab="Year")
```



Data usconsumption

Forecasting efficiency with a VAR_5

```
print(sqrt(mean(us_test[,1]-fcst$forecast$consumption$mean)^2))
```

```
## [1] 0.02851079
```

```
print(sqrt(mean(us_test[,2]-fcst$forecast$income$mean)^2))
```

```
## [1] 0.2568282
```

Data usconsumption

Forecasting of consumption and income separately

```
mod1=auto.arima(us_app[,1])
pred1=forecast(mod1,h =8)
mod2=auto.arima(us_app[,2])
pred2=forecast(mod2,h =8)
print(sqrt(mean(us_test[,1]-pred1$mean)^2))
```

```
## [1] 0.04460754
```

```
print(sqrt(mean(us_test[,2]-pred2$mean)^2))
```

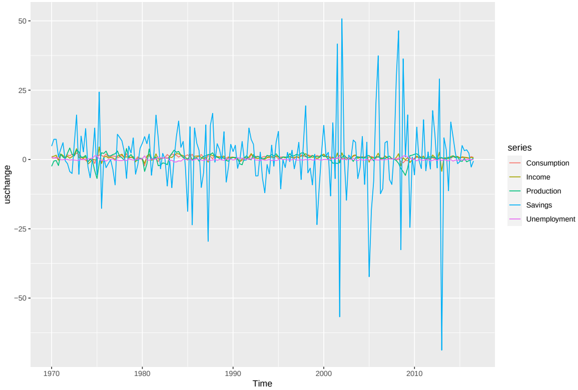
```
## [1] 0.6388838
```

Quality of prediction is lower when each times series is used separaterly.

Exercice: Data uschange

Try to find the best forecasting model for the 5 uschange time series

```r
autoplot(uschange)
```



Forecasting efficiency will be evaluated on 2016 data, and compare to forecasting each time series separately.

To go further

Facebook develop a kind of automatic time series modelling which appear to be relatively efficient (but I never test it). If you are interested in fast forecast, you can test it : https://cran.r-project.org/web/packages/prophet/index.html