Times series forecasting

Advanced forecasting methods

Julien JACQUES

Université Lumière Lyon 2

Neural network models

Time series regression models

Dynamic regression model

Grouped time series models: VAR 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 :

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

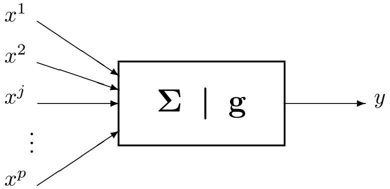


Figure 1: Neuron representation

- 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*:

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

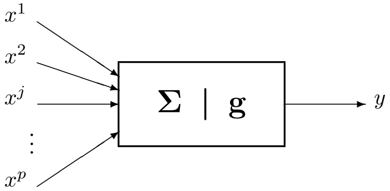


Figure 2: Neuron representation

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

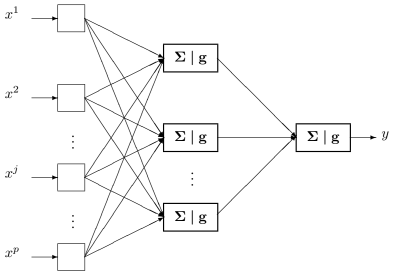

x^1
x^2
x^j
x^p
$\mathbf{\Sigma} \mid \mathbf{g}$
$\mathbf{\Sigma} \mid \mathbf{g}$
$\mathbf{\Sigma} \mid \mathbf{g}$
$\mathbf{\Sigma} \mid \mathbf{g}$
y

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 $x_{t-1}, \dots, x_{t-p}$
  - 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...).

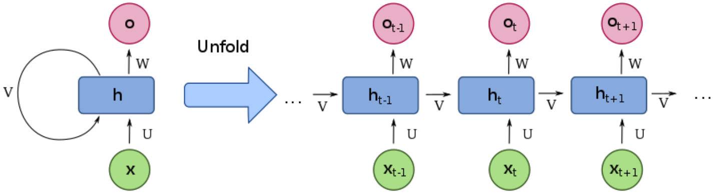


Figure 4: Neuron representation

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:

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

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

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

More Neural Network

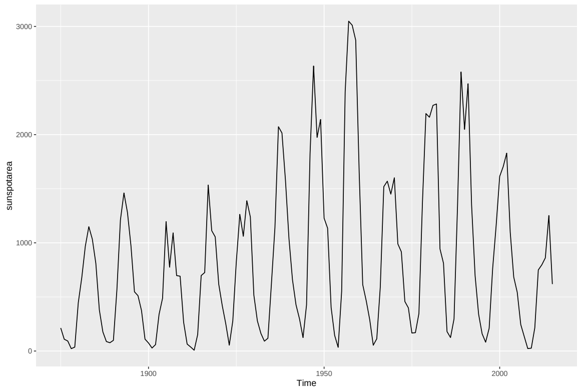
To my experience, such models: - are efficient when the series is hard to forecast, with no evident model behind and when usual model are not efficient - need time series of large sizes

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:

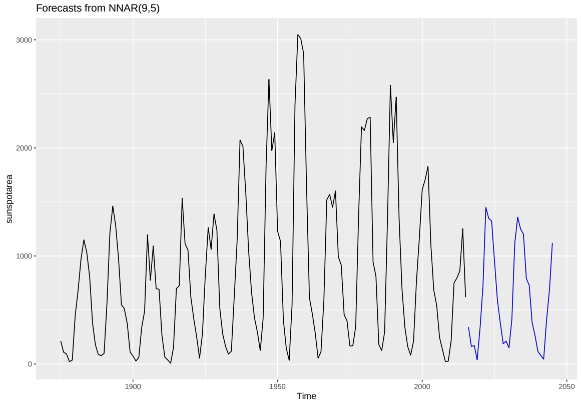
```
fit=nnetar(sunspotarea)
print(fit)
```

```
## 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 10583
```

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 ;-)

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:

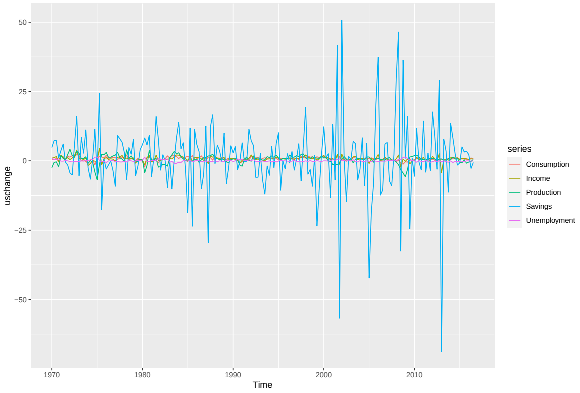
$$x_t = \underbrace{c + \beta_0 t}_{\text{trend}} + \underbrace{\delta_2 d_{2t} + \ldots + \delta_T d_{Tt}}_{\text{seasonal effect}} + \underbrace{\beta_1 z_t + \ldots + \beta_k z_{kt}}_{\text{covariates}} + \epsilon_t.$$

  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

```
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

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

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

```
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.

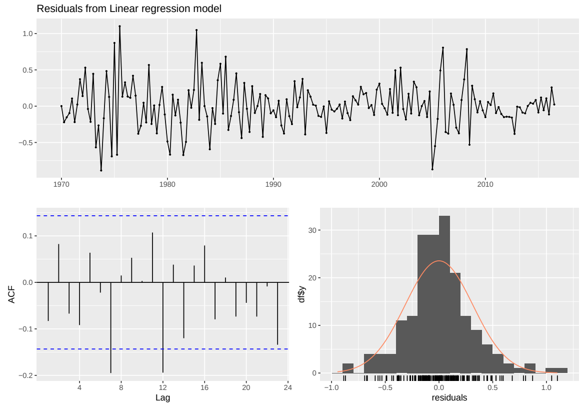
To the best of my knowledge, such procedures are not available for the `tslm` function.

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* = 23.979, df = 3, p-value = 2.524e-05
##
## Model df: 9.   Total lags used: 12
```

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 $ARIMA_{1,0,2}$:

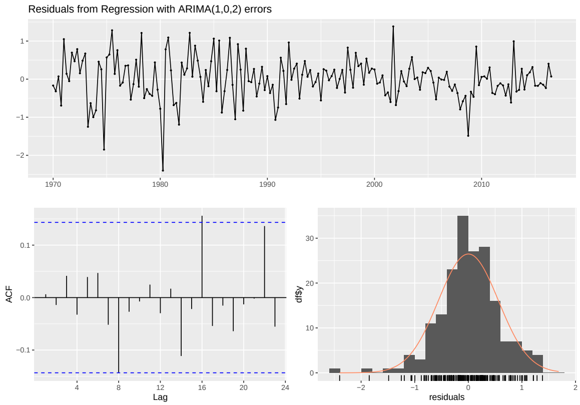
```
fit=Arima(uschange[,'Consumption'],xreg=uschange[,'Income'],order=c(1,0,2))
summary(fit)
```

```
## 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 estimated as 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:

```
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 = 3, p-value = 0.117
##
## Model df: 5.   Total lags used: 8
```

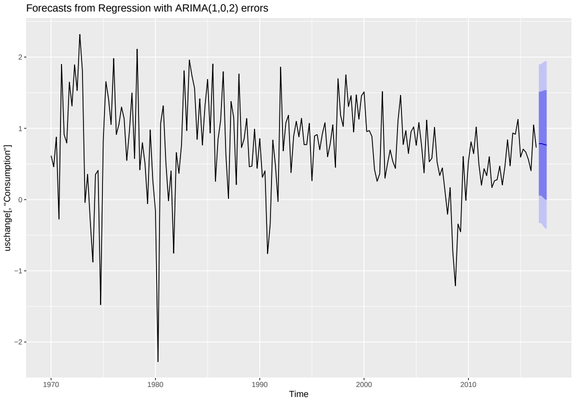
It seems that all the auto-correlations of the residuals have been modelled with this model.

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 !

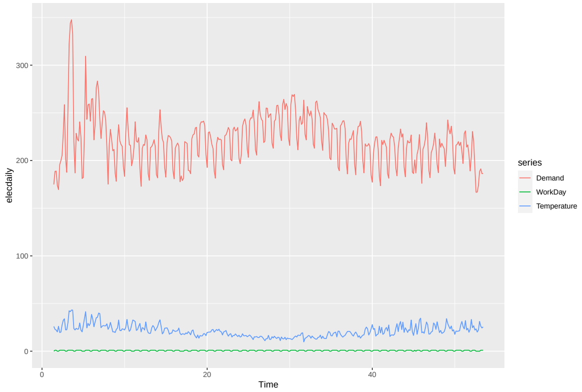
```
autoplot(forecast(fit,xreg=rep(mean(uschange[,2]),4)))
```



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 :

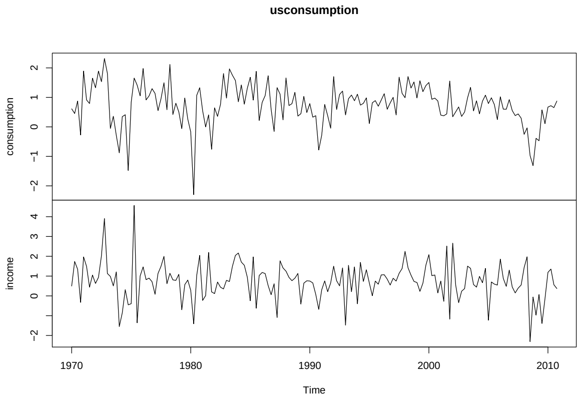
$$\begin{aligned} X_{1,t} &= c_1 + \epsilon_{1,t} + a_{1,1}X_{1,t-1} + a_{1,2}X_{2,t-1}, \\ X_{2,t} &= c_2 + \epsilon_{2,t} + a_{2,1}X_{1,t-1} + a_{2,2}X_{2,t-1}, \end{aligned}$$

- High order model can be also considered VAR_p

Data usconsumption

We will work with data usconsumption

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



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 $1 \leq p \leq lag.max$

```r
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

```
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 = NULL,exogen=NULL)
summary(var)
```

```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: consumption, income
## Deterministic variables: const
## Sample size: 151
## Log Likelihood: -308.714
## Roots of the characteristic polynomial:
## 0.7556 0.7556 0.7317 0.7274 0.7274 0.5929 0.5929 0.5861 0.586
## Call:
## VAR(y = us_app, p = 5, type = "const", exogen = NULL)
##
##
## 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

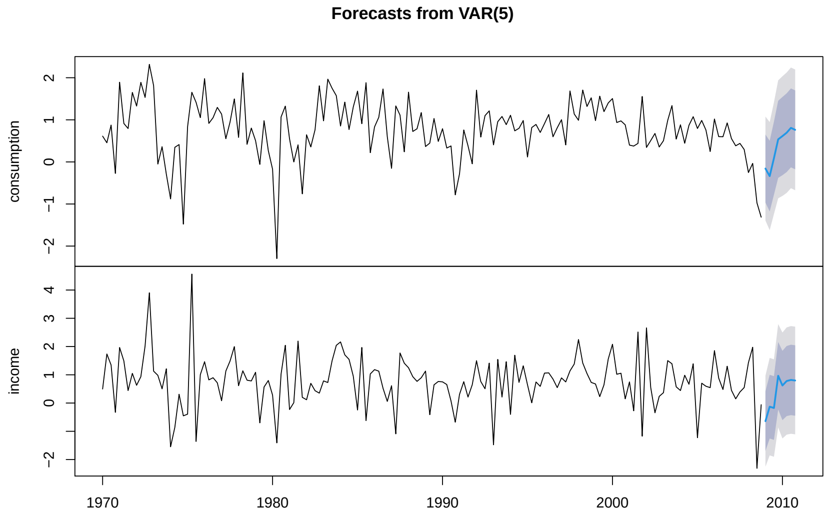
```
serial.test(var, lags.pt=10, type="PT.asymptotic")
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object var
## Chi-squared = 13.11, df = 20, p-value = 0.8726
```

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] 0.03271834
```

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

```
## [1] 0.2508865
```

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))
```

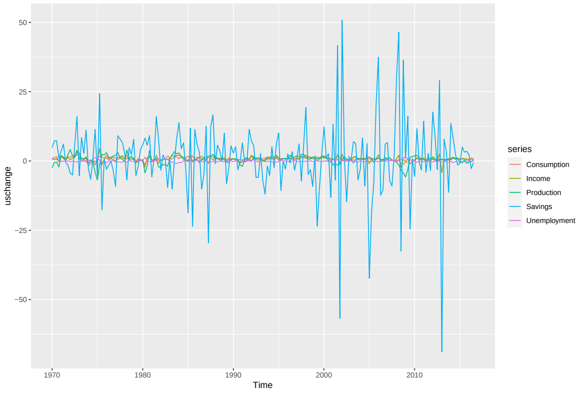
```
## [1] 0.6388838
```

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

```
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