BCEAO - 20 au 25 janvier 2019

2 - ECM et SARIMA

Dominique Ladiray et Alain Quartier-la-Tente
(dominique.ladiray@insee.fr et alain.quartier@yahoo.fr)

Sommaire

Modèles à corrections d'erreurs

De nombreux packages sont disponibles pour faire des estimations avec des modèles ECM $\rightarrow$ on va utiliser `ecm`.

Modèle général :

$$\Delta y = \beta_0 + \beta_1 \Delta x_{1,t} + ... + \beta_i \Delta x_{i,t} + \gamma(y_{t-1} - (\alpha_1 x_{1,t-1} + ... + \alpha_i x_{i,t-1}))$$

Modèle estimé :

$$\Delta y = \beta_0 + \underbrace{\beta_1 \Delta x_{1,t} + ... + \beta_i \Delta x_{i,t}}_{\text{court terme ("transient term")}} + \gamma y_{t-1} + \underbrace{\gamma_1 x_{1,t-1} + ... + \gamma_i x_{i,t-1}}_{\text{long terme ("equilibrium term")}}$$

Modèles à corrections d'erreurs

Estimation à partir de la fonction `ecm()` qui a 4 paramètres :

- `y` la variable d'intérêt (un `data.frame`)
- `xeq` les variables de long terme (un `data.frame`)
- `xtr` les variables de court terme (un `data.frame`)
- `includeIntercept` booléen indiquant si l'on ajoute ou non une constante

```
library(ecm)
donnees_ts <- ts.intersect(raffinage, brent)
data <- data.frame(donnees_ts) # il faut des data.frame
model <- ecm(y = data["raffinage"],
             xeq = data["brent"],
             xtr = data["brent"],
             includeIntercept=TRUE)
```

Modèles à correction d'erreur

```
summary(model)
```

```
##
## Call:
## lm(formula = dy ~ ., data = x, weights = weights)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7703 -0.7356  0.0397  0.7718  4.3810
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.02129    0.93933   4.281 2.59e-05 ***
## deltabrent   0.44470    0.01228  36.208  < 2e-16 ***
## brentLag1    0.05975    0.01344   4.447 1.28e-05 ***
## yLag1       -0.10334    0.02357  -4.385 1.67e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Modèles à correction d'erreur

Pour obtenir de manière dynamique les prévisions, pas de solution disponible dans le package... voir programme `TP/3_1 - Modèle ecm.R` pour une solution (on fait une boucle pour mettre à jour de manière dynamique en récupérant à chaque fois la prévision en m+1)

Autres packages R ne font que des VECM : `tsDyn` avec la fonction `VECM()` par exemple... Mais permettent d'avoir les prévisions de manière dynamique

Sommaire

Modèles SARIMA

```
library(forecast)
arima_auto <- auto.arima(ipch_benin[,"ensemble"])
arima_auto
```

```
## Series: ipch_benin[, "ensemble"]
## ARIMA(2,1,1)(0,0,2)[12] with drift
##
## Coefficients:
##          ar1      ar2      ma1    sma1    sma2   drift
##       0.6855  -0.1198  -0.7307  0.0809  0.0899  0.1592
## s.e.  0.1653   0.0709   0.1581  0.0630  0.0608  0.0430
##
## sigma^2 estimated as 0.9209:  log likelihood=-347.09
## AIC=708.19   AICc=708.64   BIC=732.95
```

Modèles ARIMA

```
# On peut aussi redéfinir le modèle à main :
arima_fixe <- Arima(ipch_benin[,"ensemble"],
                    order = c(2,1,1), seasonal = c(0,0,2),
                    include.drift = TRUE)
summary(arima_fixe)
```

```
## Series: ipch_benin[, "ensemble"]
## ARIMA(2,1,1)(0,0,2)[12] with drift
##
## Coefficients:
##          ar1      ar2      ma1    sma1    sma2   drift
##       0.6855  -0.1198  -0.7307  0.0809  0.0899  0.1592
## s.e.  0.1653   0.0709   0.1581  0.0630  0.0608  0.0430
##
## sigma^2 estimated as 0.9209:  log likelihood=-347.09
## AIC=708.19   AICc=708.64   BIC=732.95
##
## Training set error measures:
```

Modèles ARIMA

```
accuracy(arima_auto)
```

```
##                       ME      RMSE       MAE           MPE
## Training set -0.0003855297 0.9463808 0.7132021 4.88716e-05 0.8
##                      ACF1
## Training set -0.005317493
```

```
forecast(arima_auto, h = 10)
```

```
##          Point Forecast     Lo 80    Hi 80    Lo 95    Hi 95
## Apr 2018       101.5592 100.32942 102.7891 99.67838 103.4401
## May 2018       101.7273 100.02679 103.4277 99.12661 104.3279
## Jun 2018       101.8454  99.87819 103.8126 98.83682 104.8539
## Jul 2018       101.9287  99.77828 104.0792 98.63990 105.2175
## Aug 2018       101.9232  99.62586 104.2205 98.40972 105.4367
## Sep 2018       101.9448  99.51823 104.3714 98.23367 105.6560
## Oct 2018       102.0983  99.55248 104.6442 98.20479 105.9919
## Nov 2018       102.3853  99.72680 105.0438 98.31948 106.4511
## Dec 2018       102.5334  99.76731 105.2994 98.30304 106.7637
```

Modèles SARIMA

```r
library(RJDemetra)
arima_auto_jd <- regarima_x13(ipch_benin[,"ensemble"])
summary(arima_auto_jd)
```

```
## y = regression model + arima (0, 1, 0, 0, 1, 1)
##
## Model: RegARIMA - X13
## Estimation span: from 1-1997 to 3-2018
## Log-transformation: no
## Regression model: no mean, no trading days effect, no leap yea
##
## Coefficients:
## ARIMA:
##             Estimate Std. Error T-stat Pr(>|t|)
## BTheta(1)  -0.9145     0.0355 -25.76   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

Modèles SARIMA

```
#Pour obtenir les prévisions :
arima_auto_jd$forecast
```

```
##                fcst   fcsterr
## Apr 2018 102.0625 0.8362706
## May 2018 102.6104 1.1809875
## Jun 2018 102.5594 1.4466714
## Jul 2018 102.2266 1.6701685
## Aug 2018 101.4578 1.8673051
## Sep 2018 101.2930 2.0455303
## Oct 2018 101.6243 2.2094252
## Nov 2018 102.1102 2.3619749
## Dec 2018 102.3557 2.5052527
## Jan 2019 102.7234 2.6411578
## Feb 2019 102.2990 2.7699502
## Mar 2019 102.6144 2.8930147
## Apr 2019 103.4769 3.0328435
## May 2019 104.0249 3.1658775
```

Modèles SARIMA

```
#Pour avoir une plus grande période, il faut refaire une specification
arima_auto_jd_spec <- regarima_spec_x13(arima_auto_jd,
                      # -3 pour trois années, équivalent ) 12*3
                                        fcst.horizon = -3)

arima_auto_jd <- regarima(ipch_benin[,"ensemble"],
                          arima_auto_jd_spec)
arima_auto_jd$forecast
```

```
##                fcst   fcsterr
## Apr 2018 102.0625 0.8362706
## May 2018 102.6104 1.1809875
## Jun 2018 102.5594 1.4466714
## Jul 2018 102.2266 1.6701685
## Aug 2018 101.4578 1.8673051
## Sep 2018 101.2930 2.0455303
## Oct 2018 101.6243 2.2094252
## Nov 2018 102.1102 2.3619749
## Dec 2018 102.3557 2.5052527
## Jan 2019 102.7234 2.6411578
## Feb 2019 102.2990 2.7699502
## Mar 2019 102.6144 2.8930147
## Apr 2019 103.4769 3.0328435
```

Modèles SARIMA

```
arima_fixe_jd_spec <-
  regarima_spec_x13(arima_auto_jd,
                    automdl.enabled = FALSE,
                    arima.mu = FALSE,
                    arima.p = 0, arima.d = 1, arima.q = 0,
                    arima.bp = 0, arima.bd = 1, arima.bq = 1)
arima_auto_jd <- regarima(ipch_benin[,"ensemble"],
                          arima_fixe_jd_spec)
arima_auto_jd
```

```
## y = regression model + arima (0, 1, 0, 0, 1, 1)
## Log-transformation: no
## Coefficients:
##           Estimate Std. Error
## BTheta(1)  -0.9145      0.036
##
##              Estimate Std. Error
## LS (1-2012)     4.379      0.824
## AO (4-2015)    -2.747      0.587
## AO (6-2009)    -2.322      0.582
##
##
## Residual standard error: 0.8265 on 237 degrees of freedom
```

Sommaire

Modèles ARDL

dynlm permet de faire facilement des modèles linéaires avec des transformations

```
library(dynlm)
# Modèle sans aucun sens économique
mod <- dynlm(ensemble ~  transport +
               L(transport, 12) + # lag d'ordre 12
               diff(transport,1), # On différencie
             data = ipch_benin)
# Équivalent à :
mod <- dynlm(ensemble ~  transport +
               lag(transport, -12) + # lag d'ordre 12
               diff(transport,1), # On différencie
             data = ipch_benin)
```

Modèles ARDL

```
summary(mod)
```

```
##
## Time series regression with "ts" data:
## Start = 1998(1), End = 2018(3)
##
## Call:
## dynlm(formula = ensemble ~ transport + lag(transport, -12) +
##     diff(transport, 1), data = ipch_benin)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.3295 -1.7062  0.0818  1.4791  5.8110
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)         40.05389    0.58882  68.024   <2e-16 ***
## transport            0.38936    0.02230  17.463   <2e-16 ***
                                                      ***
                                                      0
```

Modèles ARDL

```
predict(mod)
```

```
##   jan 1998   fév 1998   mar 1998   avr 1998   mai 1998   jui 1998   j
##   63.40777   63.73444   65.62796   64.37006   64.17943   63.98067   6
##   sep 1998   oct 1998   nov 1998   déc 1998   jan 1999   fév 1999   m
##   63.88444   63.93878   63.98123   64.96601   64.62967   64.56988   6
##   mai 1999   jui 1999   jul 1999   aoû 1999   sep 1999   oct 1999   n
##   64.54029   64.32416   64.24145   64.14488   64.10496   64.22332   6
##   jan 2000   fév 2000   mar 2000   avr 2000   mai 2000   jui 2000   j
##   64.26620   65.28563   66.39045   66.61324   66.60389   68.49304   6
##   sep 2000   oct 2000   nov 2000   déc 2000   jan 2001   fév 2001   m
##   68.94051   69.12514   69.72781   70.30039   70.15012   71.01452   7
##   mai 2001   jui 2001   jul 2001   aoû 2001   sep 2001   oct 2001   n
##   72.34986   73.71502   73.41947   73.07522   73.45787   73.22699   7
##   jan 2002   fév 2002   mar 2002   avr 2002   mai 2002   jui 2002   j
##   73.26592   73.35590   73.48456   73.47600   73.32740   73.99958   7
##   sep 2002   oct 2002   nov 2002   déc 2002   jan 2003   fév 2003   m
##   73.60858   73.64230   73.55058   73.57115   73.44422   74.15588   7
```

Modèles ARDL

```
library(dynamac)
model <- dynardl(ensemble ~ transport + alimentaires,
        lags = list("ensemble"= 1,"transport" = 1),
        diffs = c("alimentaires"),
        ec = TRUE, simulate = FALSE,
        data = data.frame(ipch_benin))
```

```
## [1] "Error correction (EC) specified; dependent variable to be
```

Modèles ARDL

```
summary(model)
```

```
##
## Call:
## lm(formula = as.formula(paste(paste(dvnamelist), "~", paste(co
##     collapse = "+"), collapse = " ")))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.8583 -0.2671 -0.0674  0.1511  5.2282
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -1.127981   0.512643  -2.200  0.02870 *
## l.1.ensemble      0.037092   0.012519   2.963  0.00334 **
## d.1.alimentaires  0.363401   0.016656  21.818  < 2e-16 ***
## l.1.transport    -0.025457   0.007662  -3.323  0.00103 **
## ---
```