The R packages VGAM and VGAMextra handling systems of cointegrated time series (Bivariate case)

by Victor Miranda
PhD Candidate, The University of Auckland
Supervisor: Thomas Yee.

March 1, 2018

VGLMs/VGAMs possess infrastructure to handle systems of cointegrated time series (SCTSs). At present, I have developed software to maneuver two $I(1)$, or cointegrated order–1, time series following the two–step approach introduced by Engle and Granger (1987). This methodology involves unit root test as preamble to specify an error–correction model (ECMs) to accommodate long–stochastic trends.

Along this line, Pfaff (2011) presents a compendium of techniques to handle cointegrated time series under the same approach with examples in R, including ECMs. However, the R code choices presented seems not to handle the general case: when the off–diagonal elements of the covariance matrix are non–zero. As we will see later in this document and compared to Pfaff (2011), VGLMs become a natural choice accommodating ECMs operating the general case, and also open further areas for development.

Firstly, we say that the components of a p–dimensional vector $\boldsymbol{y}_t = (y_{1,t}, \ldots, y_{p,t})$ are cointegrated of order d, b, if all the components of $\boldsymbol{y}_t$ are integrated of order d, i.e., $y_{i,t} \sim I(d)$, and there exists a non–zero vector, $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_p)^T$ such that

$$\boldsymbol{\alpha}^T \boldsymbol{y}_t \sim I(d-b). \tag{1}$$

This is denoted $\boldsymbol{y}_t \sim \mathrm{CI}(d, b)$.

As mentioned, we will consider the case $d = b = 1$, and $p = 2$. That is, $\boldsymbol{y}_t = (y_{1,t}, y_{2,t})^T$, $y_{i,t} \sim I(1)$. Thus simulate $\boldsymbol{y}_t = (y_{1,t}, y_{2,t})^T$, two random walks, $n = 280$:

$$\begin{aligned} y_{1,t} &= y_{1,t-1} + \varepsilon_{1,t}, \\ y_{2,t} &= \beta_0 + \beta_1 y_{1,t} + \beta_2 y_{2,t-1} + \varepsilon_{2,t}, \end{aligned} \tag{2}$$

where $\boldsymbol{\varepsilon}_t = (\varepsilon_{1,t}, \varepsilon_{2,t})^T \sim N_2(\mathbf{0}, \mathbf{V})$, $\mathbf{V} = \begin{pmatrix} \sigma^2_{\varepsilon_{1,t}} & \sigma_{\varepsilon_{1,t}}\sigma_{\varepsilon_{2,t}}\rho \\ \sigma_{\varepsilon_{1,t}}\sigma_{\varepsilon_{2,t}}\rho & \sigma^2_{\varepsilon_{2,t}} \end{pmatrix}$, with, e.g.,

$\sigma_{\varepsilon_{1,t}} = \exp(\log(1.5))$, $\sigma_{\varepsilon_{2,t}} = \exp(0)$, $\rho = 0.75$, $(\beta_0, \beta_1, \beta_2)^T = (0.0, 2.5, -0.32)^T$.

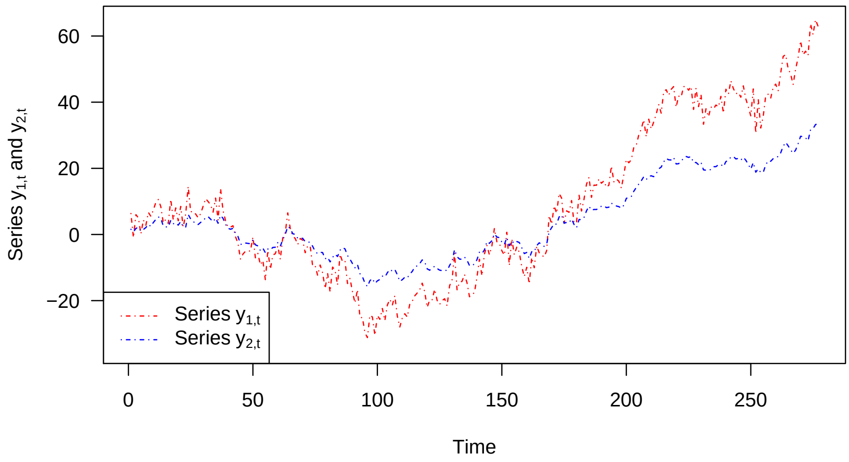

Figure 1. Non–stationary (simulated) series $y_{1,t}$ and $y_{2,t}$ from Model (2).

The initial values, $y_{1,1}$ and $y_{2,1}$, are $\varepsilon_{1,1}$ and $\varepsilon_{2,1}$ correspondingly. Both, $y_{1,t}$ and $y_{2,t}$ can be seen from Figure 1. The R code to generate this data is:

```
> set.seed(2017081901)
> nn <- 380
> warm.up <- 100
> rho <- 0.75 * 1
> # Gaussian noise1
> s2u <- exp(log(1.5))
> #ut <- rnorm(nn, 0, s2u)
> # Gaussian noise2
> s2w <- exp(0)
> #wt <- rnorm(nn, 0, s2w)
> my.errors <- rbinorm(nn, mean1 = 0, mean2 = 0, var1 = s2u, var2 = s2w, cov12 = rho)
> ut <- my.errors[, 1]
> wt <- my.errors[, 2]
> yt <- xt <- numeric(nn)
> xt[1] <- ut[1]
> yt[1] <- wt[1]
> coint.coefs <- c(0.0, 2.5, -0.32)
>
> for (ii in 2:nn) {
    xt[ii] <-  xt[ii - 1] + ut[ii]
    yt[ii] <- coint.coefs[1] + coint.coefs[2] * xt[ii]  +
                       coint.coefs[3] * yt[ii - 1] + wt[ii]
  }
> xt <- xt[-c(1:warm.up)]
> yt <- yt[-c(1:warm.up)]
>
> ## Update errors
> ut <- my.errors[-c(1:warm.up), 1]
> wt <- my.errors[-c(1:warm.up), 2]
```

To model the dynamic behaviour of (2), I will use an order(u, v) error–correction model [ECM(u, v)], whose general form is given by (equations 4.5a and 4.5b, Ch. 4 in

Pfaff, 2011):

$$
\begin{aligned}
\Delta y_{2,t} = \; & \phi_0 + \gamma_1 \widehat{z}_{t-1} + \sum_{i=1}^{u} \phi_{1,i} \Delta y_{1,t-i} + \sum_{j=1}^{v} \phi_{2,j} \Delta y_{2,t-j} + \varepsilon_{2,t}, \\
\Delta y_{1,t} = \; & \psi_0 + \gamma_2 \widehat{z}_{t-1} + \sum_{i=1}^{u} \psi_{1,i} \Delta y_{2,t-i} + \sum_{j=1}^{v} \psi_{2,j} \Delta y_{1,t-j} + \varepsilon_{1,t},
\end{aligned} \tag{3}
$$

$t = 1, \ldots, T$, where $\boldsymbol{\varepsilon}_t = (\varepsilon_{1,t}, \varepsilon_{2,t})^T \overset{\text{iid}}{\sim} N_2(\mathbf{0}, \boldsymbol{\Sigma})$, $\widehat{z}_t$ are the residuals of the static regression $y_{2,t} \sim y_{1,t}$, and $\Delta(\cdot) = (1 - L)(\cdot)$. The adjustment rate of the error from the long–run equilibrium is determined by γ_1, expected to be negative, if the system converges from its long–run equilibrium path. Note, while the vector–error terms $\boldsymbol{\varepsilon}_t$ are iid, its components may be correlated in the same time period (contemporaneous correlation).

From the two–step methodology (Engle and Granger, 1987), we firstly must verify that $y_{1,t}$ and $y_{2,t}$ are cointegrated. For this, the residuals, $\widehat{z}_t$, of the static regression $y_{2,t} \sim y_{1,t}$ must conform with stationary conditions (Engle and Granger, 1987). This may be checked via the KPSS unit root test (Kwiatkowski et al., 1991) implemented in `KPSS.test()`, from the package VGAMextra. Here, the null hypothesis states that residuals $\widehat{z}_t$ conform with stationary conditions. The results, shown below, confirm no unit roots for $\{\widehat{z}_t\}$, that is, $y_{1,t}$ and $y_{2,t}$ are cointegrated.

```
>     errors.coint   <- residuals(lm(yt ~ xt))
>     kpss.VGAMextra <- KPSS.test(x = errors.coint, type.H0 = "trend")


 H0: trend Stationarity vs. H1: Unit root.
     Test statistic: 0.078777

p-value: 0.25348
Upper tail percentiles:
                 10%    5%  2.5%    1%
Critical value  0.119 0.146 0.176 0.216
```

Alternatively, we can fit an AR(1) with no intercept over $\{\widehat{z}_t\}$, say $\widehat{z}_t = \beta_1 \widehat{z}_{t-1} + w_t$, $\{w_t\}$ white noise, and then verify whether $\alpha = 1$ is a root of the polynomial $\alpha - \widehat{\beta}_1 = 0$. For this, I will use the family function `ARff()`, with the argument `noChecks = FALSE`. The latter internally calls the function `checkTS.VGAMextra()` that computes the polynomial roots requested. The code is given next:

```
>     data.errors    <- data.frame(est.errors = errors.coint)
>     fit.root.test <- vglm(est.errors ~ 1, family = ARff(order = 1,
                                                       nodrift = TRUE,
                                                       noChecks = FALSE),
                            data = data.errors, trace = FALSE)
```

```
Checks on stationarity / invertibility successfully performed.
No roots lying inside the unit circle.
Further details within the 'summary' output.
```

Once the unit root hypothesis has been rejected for $\{\widehat{z}_t\}$, an ECM(u, v) may be specified. Interestingly, assuming bivariate Normal errors, $\boldsymbol{\varepsilon}_t = (\varepsilon_{1,t}, \varepsilon_{2,t})^T$, the aforecited ECM$(u, v)$ (cf. (3)) can be seen as a VGLM with two–responses, $\Delta y_{1,t \ |\Phi_{t-1}} = \Delta y_{1,t}$, and $\Delta y_{2,t \ |\Phi_{t-1}} = \Delta y_{2,t}$, following the bivariate Normal distribution, and fitting linear models over the conditional means $\mathbb{E}(\Delta y_{1,t \ |\Phi_{t-1}}) = \mu_{\Delta y_{1,t}}$, and $\mathbb{E}(\Delta y_{2,t \ |\Phi_{t-1}}) = \mu_{\Delta y_{2,t}}$. The result is a new class of VGLMs, called VGLM-ECMs, with following statistical structure:

$$
\begin{aligned}
(\Delta y_{1,t}, \Delta y_{2,t})^T &\sim N_2((\mu_{\Delta y_{1,t}}, \mu_{\Delta y_{2,t}})^T, \boldsymbol{\Sigma}) \qquad (4)\\
\mu_{\Delta y_{2,t}} &= \phi_0 + \gamma_1 \widehat{z}_{t-1} + \sum_{i=1}^{u} \phi_{1,i} \Delta y_{1,t-i} + \sum_{j=1}^{v} \phi_{2,j} \Delta y_{2,t-j} + \boldsymbol{\beta_1}^T \boldsymbol{x} | \Phi_{t-1},\\
\mu_{\Delta y_{1,t}} &= \psi_0 + \gamma_2 \widehat{z}_{t-1} + \sum_{i=1}^{u} \psi_{1,i} \Delta y_{2,t-i} + \sum_{j=1}^{v} \psi_{2,j} \Delta y_{1,t-j} + \boldsymbol{\beta_2}^T \boldsymbol{x} | \Phi_{t-1},
\end{aligned}
$$

with $\boldsymbol{\Sigma} = \begin{pmatrix} \sigma^2_{\varepsilon_{1,t}} & \sigma_{\varepsilon_{1,t},\ \varepsilon_{2,t}} \\ \sigma_{\varepsilon_{1,t}, \varepsilon_{2,t}} & \sigma^2_{\varepsilon_{2,t}} \end{pmatrix}$, $\sigma_{\varepsilon_{1,t},\ \varepsilon_{2,t}} = \sigma_{\varepsilon_{1,t}} \cdot \sigma_{\varepsilon_{2,t}} \cdot \rho$, and five linear predictors:

$$
\boldsymbol{\eta}_{coint} = (\mu_{\Delta y_{2,t}}, \mu_{\Delta y_{1,t}}, \log \sigma^2_{\boldsymbol{\varepsilon}_{1,t}}, \log \sigma^2_{\boldsymbol{\varepsilon}_{2,t}}, \sigma_{\varepsilon_{1,t},\ \varepsilon_{2,t}})^T.
$$

Here, $\boldsymbol{x}|\Phi_{t-1}$ denotes an additional set of explanatories with informaiton up to time $t-1$, i.e., Φ_{t-1}, while $\boldsymbol{\beta}_1^T$ and $\boldsymbol{\beta}_2^T$ are vectors with coefficients to be estimated.

Within the VGLM/VGAM framework, VGLM–ECMs as that from (4) are described by the family function ECM.EngleGran() from VGAMextra. For illustrative purposes, an VGLM–ECM(2, 2), or simply ECM(2, 2), with no covariates $\boldsymbol{x}|\Phi_{t-1}$, will be fitted to the cointegrated series (2). The R code is presented in Table 1.

Table 1. R code to fit an ECM(2, 2) to (2) using VGLMs and VGLM-ECMs.

```
> coint.Data <- data.frame(y1 = xt, y2 = yt)
> ### Using the VGLM--family function ECM.EngleGran()
> fit.coint1 <- vglm(cbind(y1, y2) ~ 1,
                     ECM.EngleGran(ecm.order = c(2, 2),
                                   resids.pattern = "neither",
                                   zero = c("var", "cov")),
                     trace = TRUE, data = coint.Data)
> my.coefs <- fitted.values(fit.coint1)
```

Note,

1. The cointegrating vector, $\boldsymbol{\alpha}$ (cf. (1)) is estimated directly by `ECM.EngleGran()`, and shown right after the Fisher scoring iterations, as follows:

```
> coint.Data <- data.frame(y1 = xt, y2 = yt)
> fit.coint1 <- vglm(cbind(y1, y2) ~ 1,
                     ECM.EngleGran(ecm.order = c(2, 2),
                                   resids.pattern = "neither",
                                   zero = c("var", "cov")), # Default
                     trace = TRUE, data = coint.Data)


VGLM    linear loop  1 :  loglikelihood = -943.83068
VGLM    linear loop  2 :  loglikelihood = -794.57369
VGLM    linear loop  3 :  loglikelihood = -769.48295
VGLM    linear loop  4 :  loglikelihood = -769.45813
VGLM    linear loop  5 :  loglikelihood = -769.45813

Co-integrated vector:
 betaY2  betaY1
 1.0000 -1.9077

Final sample size: 277


> my.coefs <- fitted.values(fit.coint1)
```

   Here, `resids.pattern = "neither"` indicates that residuals are to be computed from he regression $y_2 \sim y_1$, by MLE.

2. Several choices are available to estimate the cointegrated vector $\boldsymbol{\alpha}$ from `ECM.EngleGran()`. It can be obtained by linear regression depending upon argument `resids.pattern`, as follows:

   1) $y_{2,t} = \alpha_0 + \alpha_1 y_{1,t} + z_t$, then $\boldsymbol{\alpha} = (1, -\alpha_0, -\alpha_1)^T$, if `resids.pattern = "intercept"`,

   2) $y_{2,t} = \alpha_1 y_{1,t} + \alpha_2 t + z_t$, then $\boldsymbol{\alpha} = (1, -\alpha_1, -\alpha_2)^T$, if `resids.pattern = "trend"`,

   3) $y_{2,t} = \alpha_1 y_{1,t} + z_t$, then $\boldsymbol{\alpha} = (1, -\alpha_1)^T$, if `resids.pattern = "neither"`, or else,

   4) $y_{2,t} = \alpha_0 + \alpha_1 y_{1,t} + \alpha_2 t + z_t$, then $\boldsymbol{\alpha} = (1, -\alpha_0, -\alpha_1, -\alpha_2)^T$, if `resids.pattern = "both"`.

   For further details see `ECM.EngleGran()` help documentation.

The estimated coefficients are retrieved from the object `fit.coint1`, producing the next output

```
> coef(fit.coint1, matrix = TRUE)

                 Diff1     Diff2 loge(var1) loge(var2)  cov12
(Intercept)   0.117883  0.283723     0.3816     2.6047 4.3502
ErrorsLag1    0.033623 -0.965105     0.0000     0.0000 0.0000
diffy1Lag1   -0.201602  1.581448     0.0000     0.0000 0.0000
```

```
diffy1Lag2    0.032389  0.078887      0.0000      0.0000 0.0000
diffy2Lag1    0.069592 -1.010039      0.0000      0.0000 0.0000
diffy2Lag2    0.038764  0.080395      0.0000      0.0000 0.0000
```

As expected, $\widehat{\gamma}_1 \approx -0.965$ is negative in sign (and close to unity), assuring the system convergence to its long–run equilibrium path. Overall, results show that $y_{1,t}$ and $y_{2,t}$ are two cointegrated $I(1)$–variables guaranteeing *Granger causality* in one direction. More precisely, one series may be predicted with help of the other.

The above methodology may be collated with that dispatched in Pfaff (2011) by embedding the artificial data (2) into the R code provided. Here, the authors consider a linear model with intercept to estimate the residuals. Hence, we modify the performance of `ECM.EngleGran()` correspondingly by setting `resids.pattern = "intercept"`, as follows:

```
> fit.coint2 <- vglm(cbind(y1, y2) ~ 1,
                     ECM.EngleGran(ecm.order = c(2, 2),
                                   resids.pattern = "intercept", # To match Pfaff (2011)
                                   zero = c("var", "cov")), # Default
                     trace = TRUE, data = coint.Data)

VGLM    linear loop  1 :  loglikelihood = -943.70537
VGLM    linear loop  2 :  loglikelihood = -793.78785
VGLM    linear loop  3 :  loglikelihood = -768.45283
VGLM    linear loop  4 :  loglikelihood = -768.42765
VGLM    linear loop  5 :  loglikelihood = -768.42765

Co-integrated vector:
     betaY2 (Intercept)      betaY1
   1.000000   -0.041269   -1.906651

Final sample size: 277

> coef(fit.coint2, matrix = TRUE)

                Diff1     Diff2 loge(var1) loge(var2)  cov12
(Intercept)  0.119537  0.247060    0.38157     2.6047 4.3509
ErrorsLag1   0.037807 -0.974324    0.00000     0.0000 0.0000
diffy1Lag1  -0.206083  1.592211    0.00000     0.0000 0.0000
diffy1Lag2   0.032160  0.081888    0.00000     0.0000 0.0000
diffy2Lag1   0.071070 -1.013447    0.00000     0.0000 0.0000
diffy2Lag2   0.038036  0.081532    0.00000     0.0000 0.0000
```

Now, we run the code from Pfaff (2011) using the generated data. Note that the differences and lagged values required need to be computed and named firstly. The following R code is an option for this:

```
> ### Regression of yt on xt, save residuals. Compute Order--1 differences.
> errors.coint <- residuals(lm(yt ~ xt))  # Residuals from the static regression yt ~ xt
> difx1 <- diff(ts(xt), lag = 1, differences = 1)  # First difference for xt
> dify1 <- diff(ts(yt), lag = 1, differences = 1)  # First difference for yt
>
> ### Set up the dataset (coint.data), including Order-2 lagged differences.
> coint.data <- data.frame(embed(difx1, 3), embed(dify1, 3))
```

```
> colnames(coint.data) <- c("difx1", "difxLag1", "difxLag2",
                            "dify1", "difyLag1", "difyLag2")
>
> ### Remove unutilized lagged errors accordingly.
> errors.cointLag1 <- errors.coint[1:(nn - warm.up - 3)]
> coint.data <- transform(coint.data, errors.cointLag1 = errors.cointLag1)
>
> ## Use lm() to regress 'dy2' on 'errors.cointLag1', and oreder-2 differences
> ecm.reg <- lm(dify1 ~ errors.cointLag1 + difxLag1 + difxLag2 +
                  difyLag1 + difyLag2, data = coint.data)
> coef(ecm.reg)

     (Intercept) errors.cointLag1         difxLag1         difxLag2
         0.24706         -0.97432          1.59221          1.93958
        difyLag1         difyLag2
        -1.01345         -0.89279
```

Finally, the coefficients from both fits can be compared:

```
> ECMcoefs <- data.frame(VGLM = coef(fit.coint2, matrix = TRUE)[, 2],
                         Pfaf = coef(ecm.reg))
> head(ECMcoefs, 10)

                 VGLM     Pfaf
(Intercept)  0.247060  0.24706
ErrorsLag1  -0.974324 -0.97432
diffy1Lag1   1.592211  1.59221
diffy1Lag2   0.081888  1.93958
diffy2Lag1  -1.013447 -1.01345
diffy2Lag2   0.081532 -0.89279
```

Further improvements are to be incorporated over time, .e.g, employing the class of Reduced–Rank VGLMs (Yee, 2015, RR–VGLMs) to aid the number of coefficients as u and v increase, or to implement VGLM family functions to readily handle Vector ECMs (VECMs) for multiple cointegrate time series.

On the other hand, there is a number of advantages conferred by VGLM-ECMs and `ECM.EngleGran()`. Firstly, the inclusion of covariates that may be integrated to model the volatility involved, i.e., variance and covariances equations, in addition to the mean models. For this, one can use constraint matrices on the parameters Yee (2015), or simply set up the argument `zero`. Secondly, `ECM.EngleGran()` also provides estimates of the variance–covariance structure of the disturbances, ε_t. In our artificial example (2), it is given by

```
> coef(fit.coint2, matrix = TRUE)

                Diff1     Diff2 loge(var1) loge(var2)  cov12
(Intercept)  0.119537  0.247060    0.38157     2.6047 4.3509
ErrorsLag1   0.037807 -0.974324    0.00000     0.0000 0.0000
diffy1Lag1  -0.206083  1.592211    0.00000     0.0000 0.0000
diffy1Lag2   0.032160  0.081888    0.00000     0.0000 0.0000
diffy2Lag1   0.071070 -1.013447    0.00000     0.0000 0.0000
diffy2Lag2   0.038036  0.081532    0.00000     0.0000 0.0000
```

The package tsDyn is an alternative, however, it is restricted to Covariagtes EIMs exact no need to compute covs

References

R. Engle and C. Granger. Co–integration and error correction: Representation, estimation and testing. *Econometrica*, 55(2):98–1007, 1987.

D. Kwiatkowski, P. Phillips, P. Schmidt, and Y. Shin. Testing the null hypothesis of stationarity against the alternative of a unit root. *Journal of Econometrics*, 54:159–178, 1991.

V. Miranda and T. Yee. Vector generalized linear time series models. Manuscript in preparation, 2018.

B. Pfaff. *Analysis of integrated and cointegrated time series with R*. Springer, Seattle, Washington, USA, 2011.

R Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2017. URL `https://www.R-project.org/`.

R. Rigby and D. Stasinopouls. Generalized additive models for location, scale and shape, (with discussion). *Applied Statistics*, 54(3):507–554, 2005.

T. Yee. *Vector generalized linear and additive models with an implementation in R*. Springer, New York, USA, 2015.