Computing Generalized Method of Moments and Generalized Empirical Likelihood with R

Pierre Chaussé

Abstract

This paper shows how to estimate models by the generalized method of moments and the generalized empirical likelihood using the R package **gmm**. A brief discussion is offered on the theoretical aspects of both methods and the functionality of the package is presented through several examples in economics and finance. It is a modified version of **?** published in the Journal of Statistical Software. It has been adapted to the version 1.4-0.

Keywords: generalized empirical likelihood, generalized method of moments, empirical likelihood, continuous updated estimator, exponential tilting, exponentially tilted empirical likelihood, R.

1. Introduction

The generalized method of moments (GMM) has become an important estimation procedure in many areas of applied economics and finance since **?** introduced the two step GMM (2SGMM). It can be seen as a generalization of many other estimation methods like least squares (LS), instrumental variables (IV) or maximum likelihood (ML). As a result, it is less likely to be misspecified. The properties of the estimators of LS depend on the exogeneity of the regressors and the circularity of the residuals, while those of ML depend on the choice of the likelihood function. GMM is much more flexible since it only requires some assumptions about moment conditions. In macroeconomics, for example, it allows to estimate a structural model equation by equation. In finance, most data such as stock returns are characterized by heavy-tailed and skewed distributions. Because it does not impose any restriction on the distribution of the data, GMM represents a good alternative in this area as well. As a result of its popularity, most statistical packages like Matlab, Gauss or Stata offer tool boxes to use the GMM procedure. It is now possible to easily use this method in R with the new **gmm** package.

Although GMM has good potential theoretically, several applied studies have shown that the properties of the 2SGMM may in some cases be poor in small samples. In particular, the estimators may be strongly biased for certain choices of moment conditions. In response to this result, **?** proposed two other ways to compute GMM: the iterative GMM (ITGMM) and the continuous updated GMM (CUE)[1]. Furthermore, another family of estimation procedures inspired by **?**, which also depends only on moment conditions, was introduced by **?**. It is the generalized empirical likelihood (GEL). So far, this method has not reached the popularity

[1]See also **?** for a detailed presentation of most recent developments regarding GMM.

of GMM and it was not included in any statistical package until **gmm** was developed for R which also includes a GEL procedure.

Asymptotic properties of GMM and generalized empirical likelihood (GEL) are now well established in the econometric literature. **?** and **?** have compared their second order asymptotic properties. In particular, they show that the second order bias of the empirical likelihood (EL) estimator, which is a special case of GEL, is smaller than the bias of the estimators from the three GMM methods. Furthermore, as opposed to GMM, the bias does not increase with the number of moment conditions. Since the efficiency improves when the number of conditions goes up, this is a valuable property. However, these are only asymptotic results which do not necessarily hold in small sample as shown by **?**. In order to analyze small sample properties, we have to rely on Monte Carlo simulations. However, Monte Carlo studies on methods such as GMM or GEL depend on complicated algorithms which are often home made. Because of that, results from such studies are not easy to reproduce. The solution should be to use a common tool which can be tested and improved upon by the users. Because it is open source, R offers a perfect platform for such tool.

The **gmm** package allows to estimate models using the three GMM methods, the empirical likelihood and the exponential tilting, which belong to the family of GEL methods, and the exponentially tilted empirical likelihood which was proposed by **?**, Also it offers several options to estimate the covariance matrix of the moment conditions. Users can also choose between `optim`, if no restrictions are required on the coefficients of the model to be estimated, and either `nlminb` or `constrOptim` for constrained optimizations. The results are presented in such a way that R users who are familiar with `lm` objects, find it natural. In fact, the same methods are available for `gmm` and `gel` objects produced by the estimation procedures.

The paper is organized as follows. Section 2 presents the theoretical aspects of the GMM method along with several examples in economics and finance. Through these examples, the functionality of the **gmm** packages is presented in details. Section 3 presents the GEL method with some of the examples used in section 2. Section 4 concludes and Section 5 gives the computational details of the package.

2. Generalized method of moments

This section presents an overview of the GMM method. It is intended to help the users understand the options that the **gmm** package offers. For those who are not familiar with the method and require more details, see **?** and **?** for the method itself, **?** and **?** for the choice of the covariance matrix or **?**.

We want to estimate a vector of parameters $\theta_0 \in \mathbb{R}^p$ from a model based on the following $q \times 1$ vector of unconditional moment conditions:

$$E[g(\theta_0, x_i)] = 0, \tag{1}$$

where x_i is a vector of cross-sectional data, time series or both. In order for GMM to produce consistent estimates from the above conditions, θ_0 has to be the unique solution to $E[g(\theta, x_i)] = 0$ and be an element of a compact space. Some boundary assumptions on higher moments of $g(\theta, x_i)$ are also required. However, it does not impose any condition on the distribution of x_i, except for the degree of dependence of the observations when it is a vector of time series.

Several estimation methods such as least squares (LS), maximum likelihood (ML) or instrumental variables (IV) can also be seen as being based on such moment conditions, which make them special cases of GMM. For example, the following linear model:

$$Y = X\beta + u,$$

where Y and X are respectively $n \times 1$ and $n \times k$ matrices, can be estimated by LS. The estimate $\hat{\beta}$ is obtained by solving $\min_\beta \|u\|^2$ and is therefore the solution to the following first order condition:

$$\frac{1}{n} X'u(\beta) = 0,$$

which is the estimate of the moment condition $E(X_i u_i(\beta)) = 0$. The same model can be estimated by ML in which case the moment condition becomes:

$$E\left[\frac{dl_i(\beta)}{d\beta}\right] = 0,$$

where $l_i(\beta)$ is the density of u_i. In presence of endogeneity of the explanatory variable X, which implies that $E(X_i u_i) \neq 0$, the IV method is often used. It solves the endogeneity problem by substituting X by a matrix of instruments H, which is required to be correlated with X and uncorrelated with u. These properties allow the model to be estimated by the conditional moment condition $E(u_i|H_i) = 0$ or its implied unconditional moment condition $E(u_i H_i) = 0$. In general we say that u_i is orthogonal to an information set I_i or that $E(u_i|I_i) = 0$ in which case H_i is a vector containing functions of any element of I_i. The model can therefore be estimated by solving

$$\frac{1}{T} H'u(\beta) = 0.$$

When there is no assumption on the covariance matrix of u, the IV corresponds to GMM. If $E(X_i u_i) = 0$ holds, generalized LS with no assumption on the covariance matrix of u other than boundary ones is also a GMM method. For the ML procedure to be viewed as GMM, the assumption on the distribution of u must be satisfied. If it is not, but $E(dl_i(\theta_0)/d\theta) = 0$ holds, as it is the case for linear models with non normal error terms, the pseudo-ML which uses a robust covariance matrix can be seen as being a GMM method.

Because GMM depends only on moment conditions, it is a reliable estimation procedure for many models in economics and finance. For example, general equilibrium models suffer from endogeneity problems because these are misspecified and they represent only a fragment of the economy. GMM with the right moment conditions is therefore more appropriate than ML. In finance, there is no satisfying parametric distribution which reproduces the properties of stock returns. The family of stable distributions is a good candidate but only the densities of the normal, Cauchy and Lévy distributions, which belong to this family, have a closed form expression. The distribution-free feature of GMM is therefore appealing in that case.

Although GMM estimators are easily consistent, efficiency and bias depend on the choice of moment conditions. Bad instruments implies bad information and therefore low efficiency. The effects on finite sample properties are even more severe and are well documented in the literature on weak instruments. **?** show that the bias increases with the number of instruments but efficiency decreases. Therefore, users need to be careful when selecting the instruments. **?** gives a good review of recent developments on how to choose instruments in her introduction.

In general, the moment conditions $E(g(\theta_0, x_i)) = 0$ is a vector of nonlinear functions of θ_0 and the number of conditions is not limited by the dimension of θ_0. Since efficiency increases with the number of instruments q is often greater than p, which implies that there is no solution to

$$\bar{g}(\theta) \equiv \frac{1}{n}\sum_{i=1}^{n} g(\theta, x_i) = 0.$$

The best we can do is to make it as close as possible to zero by minimizing the quadratic function $\bar{g}(\theta)'W\bar{g}(\theta)$, where W is a positive definite and symmetric $q \times q$ matrix of weights. The optimal matrix W which produces efficient estimators is defined as:

$$W^* = \left\{ \lim_{n\to\infty} Var(\sqrt{n}\bar{g}(\theta_0)) \equiv \Omega(\theta_0) \right\}^{-1}. \tag{2}$$

This optimal matrix can be estimated by an heteroskedasticity and auto-correlation consistent (HAC) matrix like the one proposed by **?**. The general form is:

$$\hat{\Omega} = \sum_{s=-(n-1)}^{n-1} k_h(s)\hat{\Gamma}_s(\theta^*), \tag{3}$$

where $k_h(s)$ is a kernel, h is the bandwidth which can be chosen using the procedures proposed by **?** and **?**,

$$\hat{\Gamma}_s(\theta^*) = \frac{1}{n}\sum_{i} g(\theta^*, x_i)g(\theta^*, x_{i+s})'$$

and θ^* is a convergent estimate of θ_0. There are many choices for the HAC matrix. They depend on the kernel and bandwidth selection. Although the choice does not affect the asymptotic properties of GMM, very little is known about the impacts in finite samples. The GMM estimator $\hat{\theta}$ is therefore defined as:

$$\hat{\theta} = \arg\min_{\theta} \bar{g}(\theta)'\hat{\Omega}(\theta^*)^{-1}\bar{g}(\theta) \tag{4}$$

The original version of GMM proposed by **?** is called two-step GMM (2SGMM). It computes θ^* by minimizing $\bar{g}(\theta)'\bar{g}(\theta)$. The algorithm is therefore:

1- Compute $\theta^* = \arg\min_\theta \bar{g}(\theta)'\bar{g}(\theta)$

2- Compute the HAC matrix $\hat{\Omega}(\theta^*)$

3- Compute the 2SGMM $\hat{\theta} = \arg\min_\theta \bar{g}(\theta)'\left[\hat{\Omega}(\theta^*)\right]^{-1}\bar{g}(\theta)$

In order to improve the properties of 2SGMM, **?** suggest two other methods. The first one is the iterative version of 2SGMM (ITGMM) and can be computed as follows:

1- Compute $\theta^{(0)} = \arg\min_\theta \bar{g}(\theta)'\bar{g}(\theta)$

2- Compute the HAC matrix $\hat{\Omega}(\theta^{(0)})$

3- Compute the $\theta^{(1)} = \arg\min_\theta \bar{g}(\theta)'\left[\hat{\Omega}(\theta^{(0)})\right]^{-1}\bar{g}(\theta)$

4- If $\|\theta^{(0)} - \theta^{(1)}\| < tol$ stops, else $\theta^{(0)} = \theta^{(1)}$ and go to 2-

5- Define the ITGMM estimator $\hat{\theta}$ as $\theta^{(1)}$

where *tol* can be set as small as we want to increase the precision. In the other method, no preliminary estimate is used to obtain the HAC matrix. The latter is treated as a function of θ and is allowed to change when the optimization algorithm computes the numerical derivatives. It is therefore continuously updated as we move toward the minimum. For that, it is called the continuous updated estimator (CUE). This method is highly nonlinear. It is therefore crucial to choose a starting value that is not too far from the minimum. A good choice is the estimate from 2SGMM which is known to be root-n convergent. The algorithm is:

1- Compute θ^* using 2SGMM

2- Compute the CUE estimator defined as

$$\hat{\theta} = \arg\min_{\theta} \bar{g}(\theta)' \left[\hat{\Omega}(\theta)\right]^{-1} \bar{g}(\theta)$$

using θ^* as starting value.

According to **?** and **?**, 2SGMM and ITGMM are second order asymptotically equivalent. On the other hand, they show that the second order asymptotic bias of CUE is smaller. The difference in the bias comes from the randomness of θ^* in $\Omega(\theta^*)$. Iterating only makes θ^* more efficient. These are second order asymptotic properties. They are informative but may not apply in finite samples. In most cases, we have to rely on numerical simulations to analyze the properties in small samples.

Given some regularity conditions, the GMM estimator converges as n goes to infinity to the following distribution:

$$\sqrt{n}(\hat{\theta} - \theta_0) \xrightarrow{L} N(0, V),$$

where

$$V = E\left(\frac{\partial g(\theta_0, x_i)}{\partial \theta}\right)' \Omega(\theta_0)^{-1} E\left(\frac{\partial g(\theta_0, x_i)}{\partial \theta}\right)$$

Inference can therefore be performed on $\hat{\theta}$ using the assumption that it is approximately distributed as $N(\theta_0, \hat{V}/n)$.

If $q > p$, we can perform a J-test to verify if the moment conditions hold. The null hypothesis and the statistics are respectively $H0 : E[g(\theta, x_i)] = 0$ and:

$$n\bar{g}(\hat{\theta})'[\hat{\Omega}(\theta^*)]^{-1}\bar{g}(\hat{\theta}) \xrightarrow{L} \chi^2_{q-p}.$$

3. GMM with R

The **gmm** package can be loaded the usual way.

```
> library(gmm)
```

The main function is `gmm()` which creates an object of class `gmm`. Many options are available but in many cases they can be set to their default values. They are explained in details below through examples. The main arguments are `g` and `x`. For a linear model, `g` is a formula like `y~z1+z2` and `x` the matrix of instruments. In the nonlinear case, they are respectively the function $g(\theta, x_i)$ and its argument. The available methods are `coef`, `vcov`, `summary`, `residuals`, `fitted.values`, `plot`, `confint`. The model and data in a `data.frame` format can be extracted by the generic function `model.frame`.

3.1. Estimating the parameters of a normal distribution

This example[2], is not something we want to do in practice, but its simplicity allows us to understand how to implement the `gmm()` procedure by providing the gradient of $g(\theta, x_i)$. It is also a good example of the weakness of GMM when the moment conditions are not sufficiently informative. In fact, the ML estimators of the mean and the variance of a normal distribution are more efficient because the likelihood carries more information than few moment conditions.

For the two parameters of a normal distribution (μ, σ) we have the following vector of moment conditions:

$$E[g(\theta, x_i)] \equiv E\left[\begin{array}{c} \mu - x_i \\ \sigma^2 - (x_i - \mu)^2 \\ x_i^3 - \mu(\mu^2 + 3\sigma^2) \end{array}\right] = 0,$$

where the first two can be directly obtained by the definition of (μ, σ) and the last comes from the third derivative of the moment generating function evaluated at 0.

We first need to create a function $g(\theta, x)$ which returns an $n \times 3$ matrix:

```
> g1 <- function(tet,x)
+         {
+         m1 <- (tet[1]-x)
+         m2 <- (tet[2]^2 - (x - tet[1])^2)
+         m3 <- x^3-tet[1]*(tet[1]^2+3*tet[2]^2)
+         f <- cbind(m1,m2,m3)
+         return(f)
+         }
```

The following is the gradient of $\bar{g}(\theta)$:

$$G \equiv \frac{\partial \bar{g}(\theta)}{\partial \theta} = \left(\begin{array}{cc} 1 & 0 \\ 2(\bar{x} - \mu) & 2\sigma \\ -3(\mu^2 + \sigma^2) & -6\mu\sigma \end{array}\right).$$

If provided, it will be used to compute the covariance matrix of $\hat{\theta}$. It can be created as follows:

```
> Dg <- function(tet,x)
+         {
+         G <- matrix(c( 1,
+                                  2*(-tet[1]+mean(x)),
+                                  -3*tet[1]^2-3*tet[2]^2,0,
```

[2] Thanks to Dieter Rozenich for his suggestion.

```
+                                2*tet[2],-6*tet[1]*tet[2]),
+                              nrow=3,ncol=2)
+         return(G)
+         }
```

First we generate normally distributed random numbers:

```
> set.seed(123)
> n <- 200
> x1 <- rnorm(n, mean = 4, sd = 2)
```

We then run `gmm` using the starting values $(\mu_0, \sigma_0^2) = (0, 0)$

```
> print(res <- gmm(g1,x1,c(mu = 0, sig = 0), grad = Dg))

Method
 twoStep

Objective function value:  0.01287054

    mu     sig
3.8762  1.7887

Convergence code =  0
```

The `summary` method prints more results from the estimation:

```
> summary(res)

Call:
gmm(g = g1, x = x1, t0 = c(mu = 0, sig = 0), gradv = Dg)


Method:  twoStep

Kernel:  Quadratic Spectral

Coefficients:
     Estimate     Std. Error   t value      Pr(>|t|)
mu    3.8762e+00   1.2143e-01   3.1922e+01  1.3310e-223
sig   1.7887e+00   8.3299e-02   2.1474e+01  2.7439e-102

J-Test: degrees of freedom is 1
                J-test   P-value
Test E(g)=0:    2.57411  0.10863

Initial values of the coefficients
```

```
      mu      sig
4.022499 1.881766

#############
Information related to the numerical optimization
Convergence code =  0
Function eval. =  55
Gradian eval. =  NA
```

The section "Initial values of the coefficients" shows the first step estimates used to either compute the weighting matrix in the 2-step GMM or the fixed bandwidth in CUE or iterative GMM.

The J-test of over-identifying restrictions can also be extracted by using the method `specTest`:

```
> specTest(res)

 ##  J-Test: degrees of freedom is 1  ##

                J-test   P-value
Test E(g)=0:    2.57411  0.10863
```

A small simulation using the following function shows that ML produces estimators with smaller mean squared errors than GMM based on the above moment conditions. However, it is not GMM but the moment conditions that are not efficient, because ML is GMM with the likelihood derivatives as moment conditions.

```
> sim_ex <- function(n,iter)
+     {
+     tet1 <- matrix(0,iter,2)
+     tet2 <- tet1
+     for(i in 1:iter)
+       {
+       x1 <- rnorm(n, mean = 4, sd = 2)
+       tet1[i,1] <- mean(x1)
+       tet1[i,2] <- sqrt(var(x1)*(n-1)/n)
+       tet2[i,] <- gmm(g1,x1,c(0,0),grad=Dg)$coefficients
+       }
+     bias <- cbind(rowMeans(t(tet1)-c(4,2)),rowMeans(t(tet2)-c(4,2)))
+     dimnames(bias)<-list(c("mu","sigma"),c("ML","GMM"))
+     Var <- cbind(diag(var(tet1)),diag(var(tet2)))
+     dimnames(Var)<-list(c("mu","sigma"),c("ML","GMM"))
+     MSE <- cbind(rowMeans((t(tet1)-c(4,2))^2),rowMeans((t(tet2)-c(4,2))^2))
+     dimnames(MSE)<-list(c("mu","sigma"),c("ML","GMM"))
+     return(list(bias=bias,Variance=Var,MSE=MSE))
+     }
```

The following results can be reproduced with $n = 50$, $iter = 2000$ and by setting `set.seed(345)`:

	μ			σ		
	Bias	Variance	MSE	Bias	Variance	MSE
GMM	0.0020	0.0929	0.0928	-0,0838	0.0481	0.0551
ML	0.0021	0.0823	0.0822	-0.0349	0.0411	0.0423

3.2. Estimating the parameters of a stable distribution

The previous example showed that ML should be used when the true distribution is known. However, when the density does not have a closed form expression, we have to consider other alternatives. **?** propose to use indirect inference and perform a numerical study to compare it with several other methods. One of them is GMM for a continuum of moment conditions and was suggested by **?**. It uses the fact that the characteristic function $E(e^{ix_i\tau})$, where i is the imaginary number and $\tau \in \mathbb{R}$, has a closed form expression (for more details on stable distribution, see **?**). The **gmm** package does not yet deal with continuum of moment conditions but we can choose a certain grid $\{\tau_1, ..., \tau_q\}$ over a given interval and estimate the parameters using the following moment conditions:

$$E\left[e^{ix_i\tau_l} - \Psi(\theta;\tau_l)\right] = 0 \quad \text{for} \quad l = 1, ..., q \ ,$$

where $\Psi(\theta;\tau_l)$ is the characteristic function. There is more than one way to define a stable distribution and it depends on the choice of parametrization. We will follow the notation of **?** and consider stable distributions $S(\alpha, \beta, \gamma, \delta; 1)$, where $\alpha \in (0, 2]$ is the characteristic exponent and $\beta \in [-1, 1]$, $\gamma > 0$ and $\delta \in \mathbb{R}$ are respectively the skewness, the scale and the location parameters. The last argument defines which parametrization we use. The **fBasics** package of **?** offers a function to generate random variables from stable distributions and uses the same notation. This parametrization implies that:

$$\Psi(\theta;\tau_l) = \begin{cases} \exp\left(-\gamma^\alpha|\tau_l|^\alpha[1 - i\beta(\tan\frac{\pi\alpha}{2})(\text{sign}(\tau_l))] + i\delta\tau_l\right) & \text{for} \quad \alpha \neq 1 \\ \exp\left(-\gamma|\tau_l|[1 + i\beta\frac{2}{\pi}(\text{sign}(\tau_l))\log|\tau_l|] + i\delta\tau_l\right) & \text{for} \quad \alpha = 1 \end{cases} ,$$

The function `charStable` included in the package computes the characteristic function and can be used to construct $g(\theta, x_i)$. To avoid dealing with complex numbers, it returns the imaginary and real parts in separate columns because both should have zero expectation. The function is:

```
> g2 <- function(theta,x)
+         {
+         tau <- seq(1,5,length.out=10)
+         pm <- 1
+         x <- matrix(c(x),ncol=1)
+         x_comp <- x%*%matrix(tau,nrow=1)
+         x_comp <- matrix(complex(ima=x_comp),ncol=length(tau))
+         emp_car <- exp(x_comp)
+         the_car <- charStable(theta,tau,pm)
+         gt <- t(t(emp_car) - the_car)
+         gt <- cbind(Im(gt),Re(gt))
+         return(gt)
+         }
```

The parameters of a simulated random vector can be estimated as follows (by default, γ and δ are set to 1 and 0 respectively in `rstable`). For the example, the starting values are the ones of a normal distribution with mean 0 and variance equals to `var(x)`:

```
> library(fBasics)
> set.seed(345)
> x2 <- rstable(500,1.5,.5,pm=1)
> t0 <- c(alpha = 2, beta = 0, gamma = sd(x2)/sqrt(2), delta = 0)
> print(res <- gmm(g2,x2,t0))

Method
 twoStep

Objective function value:  0.1463447

   alpha      beta     gamma     delta
 1.00995  -0.10974   1.41403   1.82246

Convergence code =  1
```

The result is not very close to the true parameters. But we can see why by looking at the J-test that is provided by the `summary` method:

```
> summary(res)

Call:
gmm(g = g2, x = x2, t0 = t0)


Method:  twoStep

Kernel:  Quadratic Spectral

Coefficients:
       Estimate     Std. Error   t value      Pr(>|t|)
alpha   1.0100e+00   1.5596e-01   6.4756e+00   9.4420e-11
beta   -1.0974e-01   2.4425e-01  -4.4928e-01   6.5323e-01
gamma   1.4140e+00   1.4364e-01   9.8440e+00   7.2749e-23
delta   1.8225e+00   1.5699e+02   1.1609e-02   9.9074e-01

J-Test: degrees of freedom is 16
                J-test      P-value
Test E(g)=0:    7.3172e+01  2.7583e-09

Initial values of the coefficients
      alpha        beta       gamma       delta
 0.99537936 -0.02472724  1.13127832  3.13327073
```

```
#############
Information related to the numerical optimization
Convergence code =  1
Function eval. =  501
Gradian eval. =  NA
```

The null hypothesis that the moment conditions are satisfied is rejected. For nonlinear models, a significant J-test may indicate that we have not reached the global minimum. Furthermore, the standard deviation of the coefficient of δ indicates that the covariance matrix is nearly singular. Notice also that the convergence code is equal to 1, which indicates that the algorithm did not converge. We could try different starting values, increase the number of iterations in the control option of `optim` or use `nlminb` which allows to put restrictions on the parameter space. The former would work but the latter will allow us to see how to select another optimizer. The option `optfct` can be modified to use this algorithm instead of `optim`. In that case, we can specify the upper and lower bounds of θ.

```
> res2 <- gmm(g2,x2,t0,optfct="nlminb",lower=c(0,-1,0,-Inf),upper=c(2,1,Inf,Inf))
> summary(res2)

Call:
gmm(g = g2, x = x2, t0 = t0, optfct = "nlminb", lower = c(0,
    -1, 0, -Inf), upper = c(2, 1, Inf, Inf))


Method:  twoStep

Kernel:  Quadratic Spectral

Coefficients:
       Estimate     Std. Error   t value      Pr(>|t|)
alpha   1.3878e+00   1.5182e-01   9.1410e+00   6.1898e-20
beta    4.3438e-01   2.3447e-01   1.8526e+00   6.3940e-02
gamma   9.1035e-01   4.5515e-02   2.0001e+01   5.3936e-89
delta  -1.3086e-01   3.9070e-01  -3.3494e-01   7.3767e-01

J-Test: degrees of freedom is 16
                J-test    P-value
Test E(g)=0:    16.78635   0.39955

Initial values of the coefficients
     alpha       beta      gamma      delta
 1.7227120  0.1788339  0.8909130 -0.6108679

#############
Information related to the numerical optimization
Convergence code =  0
```

```
Function eval. =  89
Gradian eval. =  316
Message:  relative convergence (4)
```

We conclude this example by estimating the parameters for a vector of stock returns from the data set `Finance` that comes with the **gmm** package.

```
> data(Finance)
> x3 <- Finance[1:1500,"WMK"]
> t0<-c(alpha = 2, beta = 0, gamma = sd(x3)/sqrt(2),delta = 0)
> res3 <- gmm(g2,x3,t0,optfct="nlminb")
> summary(res3)


Call:
gmm(g = g2, x = x3, t0 = t0, optfct = "nlminb")


Method:  twoStep

Kernel:  Quadratic Spectral

Coefficients:
       Estimate   Std. Error  t value    Pr(>|t|)
alpha   1.901891   0.028799   66.040924   0.000000
beta    0.880640   0.397088    2.217743   0.026572
gamma   0.576878   0.013713   42.067192   0.000000
delta   0.070564   0.028779    2.451951   0.014208

J-Test: degrees of freedom is 16
                J-test      P-value
Test E(g)=0:    37.5342403   0.0017626

Initial values of the coefficients
    alpha      beta     gamma     delta
2.0000000 0.0000000 0.7028397 0.0000000

#############
Information related to the numerical optimization
Convergence code =  0
Function eval. =  24
Gradian eval. =  91
Message:  both X-convergence and relative convergence (5)
```

For this sub-sample, the hypothesis that the return follows a stable distribution is rejected. The normality assumption can be analyzed by testing $H_0 : \alpha = 2,\ \beta = 0$ using `linearHypothesis` from the **car** package:

```
> library(car)
> linearHypothesis(res3,cbind(diag(2),c(0,0),c(0,0)),c(2,0))

Linear hypothesis test

Hypothesis:
alpha = 2
beta = 0

Model 1: restricted model
Model 2: res3

  Df  Chisq Pr(>Chisq)
1
2  2 24.174  5.633e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

It is clearly rejected. The result is even stronger if the whole sample is used.

3.3. A linear model with iid moment conditions

We want to estimate a linear model with an endogeneity problem. It is the model used by **?** to compare several methods which deal with the many instruments problem. We want to estimate δ from:

$$y_i = \delta W_i + \epsilon_i$$

with $\delta = 0.1$ and

$$W_i = e^{-x_i^2} + u_i,$$

where $(\epsilon_i, u_i) \sim \ iidN(0, \Sigma)$ with

$$\Sigma = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}$$

Any function of x_i can be used as an instrument because it is orthogonal to ϵ_i and correlated with W_i. There is therefore an infinite number of possible instruments. For this example, (x_i, x_i^2, x_i^3) will be the selected instruments and the sample size is set to $n = 400$:

```
> library(mvtnorm)
> sig <- matrix(c(1,.5,.5,1),2,2)
> n <- 400
> e <- rmvnorm(n,sigma=sig)
> x4 <- rnorm(n)
> w <- exp(-x4^2) + e[,1]
> y <- 0.1*w + e[,2]
```

where `rmvnorm` is a multivariate normal distribution random generator which is included in the package **mvtnorm** (**?**). For a linear model, the g argument is a formula that specifies the right- and left-hand sides as for `lm` and x is the matrix of instruments:

```
> h <- cbind(x4, x4^2, x4^3)
> g3 <- y~w
```

By default, an intercept is added to the formula and a vector of ones to the matrix of instruments. It implies the following moment conditions:

$$E\begin{pmatrix}(y_i-\alpha-\delta W_i)\\(y_i-\alpha-\delta W_i)x_i\\(y_i-\alpha-\delta W_i)x_i^2\\(y_i-\alpha-\delta W_i)x_i^3\end{pmatrix}=0$$

In order the remove the intercept, -1 has to be added to the formula. In that case there is no column of ones added to the matrix of instruments. To keep the condition that the expected value of the error terms is zero, the column of ones needs to be included manually.

We know that the moment conditions of this example are iid. Therefore, we can add the option `vcov="iid"`. This option tells `gmm` to estimate the covariance matrix of $\sqrt{n}\bar{g}(\theta^*)$ as follows:

$$\hat{\Omega}(\theta^*)=\frac{1}{n}\sum_{i=1}^{n}g(\theta^*,x_i)g(\theta^*,x_i)'$$

However, it is recommended not to set this option to "iid" in practice with real data because one of the reasons we want to use GMM is to avoid such restrictions. Finally, it is not necessary to provide the gradient when the model is linear since it is already included in `gmm`. The first results are:

```
> summary(res <- gmm(g3,x=h))

Call:
gmm(g = g3, x = h)


Method:  twoStep

Kernel:  Quadratic Spectral

Coefficients:
             Estimate   Std. Error  t value    Pr(>|t|)
(Intercept)  0.0023635  0.1029250   0.0229631  0.9816797
w            0.1086418  0.1412907   0.7689238  0.4419386

J-Test: degrees of freedom is 2
                J-test   P-value
Test E(g)=0:    1.40661  0.49495

Initial values of the coefficients
(Intercept)           w
0.004819144 0.109485921
```

By default, the 2SGMM is computed. Other methods can be chosen by modifying the option "type". The second possibility is ITGMM:

```
> res2 <- gmm(g3,x=h,type='iterative',crit=1e-8,itermax=200)
> coef(res2)

(Intercept)           w
0.002318529 0.108705556
```

The procedure iterates until the difference between the estimates of two successive iterations reaches a certain tolerance level, defined by the option `crit` (default is 10^{-7}), or if the number of iterations reaches `itermax` (default is 100). In the latter case, a message is printed to indicate that the procedure did not converge.

The third method is CUE. As you can see, the estimates from ITGMM is used as starting values. However, the starting values are required only when `g` is a function. When `g` is a formula, the default starting values are the ones obtained by setting the matrix of weights equal to the identity matrix.

```
> res3 <- gmm(g3,x=h,res2$coef,type='cue')
> coef(res3)

(Intercept)           w
 0.01767331  0.08572386
```

It is possible to produce confidence intervals by using the method `confint`:

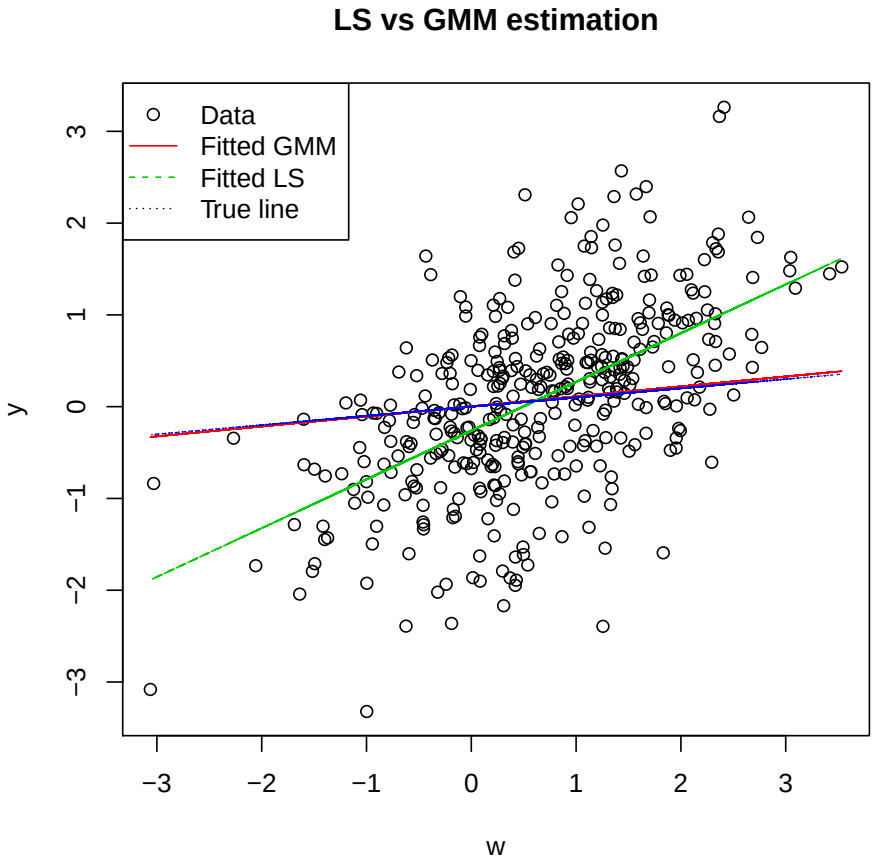
```
> confint(res3,level=.90)

                  0.05      0.95
(Intercept) -0.1549924 0.1903391
w           -0.1515377 0.3229854
```

Whether `optim` or `nlminb` is used to compute the solution, it is possible to modify their default options by adding `control=list()`. For example, you can keep track of the convergence with `control=list(trace=TRUE)` or increase the number of iterations with `control=list(maxit=1000)`. You can also choose the `BFGS` algorithm with `method="BFGS"` (see `help(optim)` for more details).

The methods `fitted` and `residuals` are also available for linear models. We can compare the fitted values of `lm` with the ones from `gmm` to see why this model cannot be estimated by LS.

```
> plot(w,y,main="LS vs GMM estimation")
> lines(w,fitted(res),col=2)
> lines(w,fitted(lm(y~w)),col=3,lty=2)
> lines(w,.1*w,col=4,lty=3)
> legend("topleft",c("Data","Fitted GMM","Fitted LS","True line"),pch=c(1,NA,NA,NA),col=1:
```



The LS seems to fit the model better. But the graphics hides the endogeneity problem. LS overestimates the relationship between y and w because it does not take into account the fact that some of the correlation is caused by the fact that y_i and w_i are positively correlated with the error term ϵ_i.

Finally, the `plot` method produces some graphics to analyze the properties of the residuals. It can only be applied to `gmm` objects when `g` is a formula because when `g` is a function, residuals are not defined.

3.4. Estimating the AR coefficients of an ARMA process

The estimation of auto-regressive coefficients of ARMA(p,q) processes is better performed by ML or nonlinear LS. But in Monte Carlo experiments, it is often estimated by GMM to study its properties. It gives a good example of linear models with endogeneity problems in which the moment conditions are serially correlated and possibly conditionally heteroskedastic. As opposed to the previous example, the choice of the HAC matrix becomes an important issue.

We want to estimate the AR coefficients of the following process:

$$X_t = 1.4X_{t-1} - 0.6X_{t-2} + u_t$$

where $u_t = 0.6\epsilon_{t-1} - 0.3\epsilon_{t-2} + \epsilon_t$ and $\epsilon_t \sim iidN(0,1)$. This model can be estimated by GMM using any X_{t-s} for $s > 2$, because they are uncorrelated with u_t and correlated with X_{t-1} and

X_{t-2}. However, as s increases the quality of the instruments decreases since the stationarity of the process implies that the auto-correlation goes to zero. For this example, the selected instruments are $(X_{t-3}, X_{t-4}, X_{t-5}, X_{t-6})$ and the sample size equals 400. The ARMA(2,2) process is generated by the function `arima.sim`:

```
> t <- 400
> set.seed(345)
> x5 <- arima.sim(n=t,list(ar=c(1.4,-0.6),ma=c(0.6,-0.3)))
> x5t<-cbind(x5)
> for (i in 1:6) x5t<-cbind(x5t,lag(x5,-i))
> x5t<-na.omit(x5t)
> g4<-x5t[,1]~x5t[,2]+x5t[,3]
> res<-gmm(g4,x5t[,4:7])
> summary(res)

Call:
gmm(g = g4, x = x5t[, 4:7])


Method:  twoStep

Kernel:  Quadratic Spectral

Coefficients:
             Estimate     Std. Error   t value      Pr(>|t|)
(Intercept)  -1.0548e-01   7.9814e-02  -1.3216e+00   1.8630e-01
x5t[, 2]      1.2599e+00   1.2382e-01   1.0176e+01   2.5485e-24
x5t[, 3]     -5.1839e-01   9.6723e-02  -5.3595e+00   8.3458e-08

J-Test: degrees of freedom is 2
                J-test   P-value
Test E(g)=0:    0.29450  0.86308

Initial values of the coefficients
(Intercept)    x5t[, 2]    x5t[, 3]
 -0.1000513   1.2544985  -0.5136757
```

The optimal matrix, when moment conditions are based on time series, is an HAC matrix which is defined by equation (3). Several estimators of this matrix have been proposed in the literature. Given some regularity conditions, they are asymptotically equivalent. However, their impacts on the finite sample properties of GMM estimators may differ. The **gmm** package uses the **sandwich** package to compute these estimators which are well explained by **?** and **?**. We will therefore briefly summarize the available options.

The option `kernel` allows to choose between five kernels: Truncated, Bartlett, Parzen, Tukey-Hanning and Quadratic spectral[3]. By default, the Quadratic Spectral kernel is used as it was

[3]The first three have been proposed by **?**, **?** and **?** respectively and the last two, applied to HAC estimation, by **?**. But the latter gives a good review of all five.

shown to be optimal by ? with respect to some mean squared error criterion. In most statistical packages, the Bartlett kernel is used for its simplicity. It makes the estimation of large models less computationally intensive. It may also make the gmm algorithm more stable numerically when dealing with highly nonlinear models, especially with CUE. We can compare the results with different choices of kernel:

```
> res2 <- gmm(g4,x=x5t[,4:7],kernel="Truncated")
> coef(res2)

(Intercept)    x5t[, 2]    x5t[, 3]
 -0.1026734   1.2603526  -0.5183035

> res3 <- gmm(g4,x=x5t[,4:7],kernel="Bartlett")
> coef(res3)

(Intercept)    x5t[, 2]    x5t[, 3]
 -0.1046965   1.2609165  -0.5192982

> res4 <- gmm(g4,x=x5t[,4:7],kernel="Parzen")
> coef(res4)

(Intercept)    x5t[, 2]    x5t[, 3]
 -0.1048764   1.2618772  -0.5200277

> res5<- gmm(g4,x=x5t[,4:7],kernel="Tukey-Hanning")
> coef(res5)

(Intercept)    x5t[, 2]    x5t[, 3]
 -0.1047404   1.2606906  -0.5191553
```

The similarity of the results is not surprising since the matrix of weights should only affect the efficiency of the estimator. We can compare the estimated standard deviations using the method vcov:

```
> diag(vcov(res2))^.5

(Intercept)    x5t[, 2]    x5t[, 3]
 0.08103396  0.12448129  0.09709397

> diag(vcov(res3))^.5

(Intercept)    x5t[, 2]    x5t[, 3]
 0.07861272  0.12254082  0.09540993

> diag(vcov(res4))^.5
```

```
(Intercept)    x5t[, 2]    x5t[, 3]
 0.07969795  0.12399019  0.09687048
```

```
> diag(vcov(res5))^.5
```

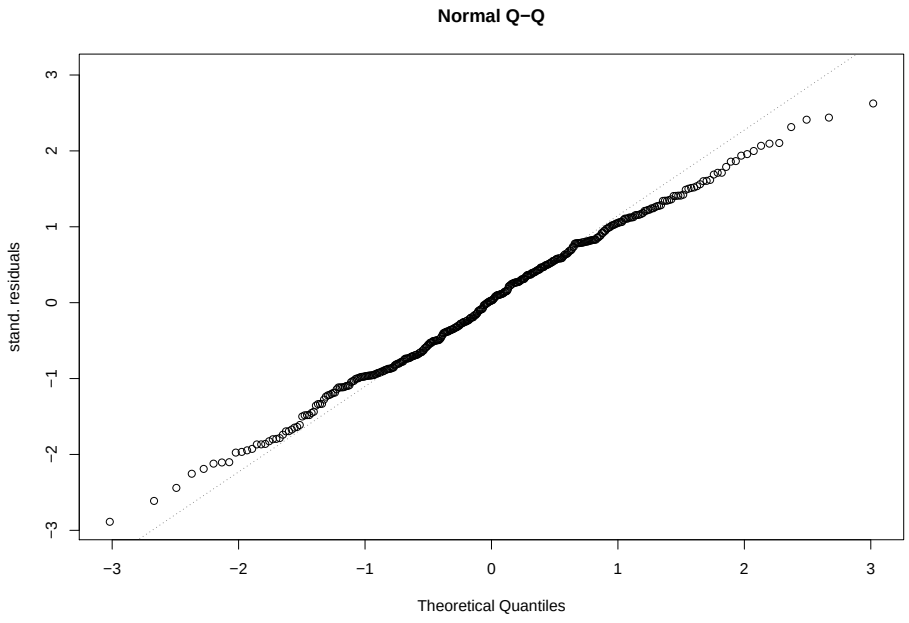
```
(Intercept)    x5t[, 2]    x5t[, 3]
 0.08004569  0.12300914  0.09620664
```

which shows, for this example, that the Bartlett kernel generates the estimates with the smallest variances. However, it does not mean it is better. We have to run simulations and compute the true variance if we want to compare them. In fact, we do not know which one produces the most accurate estimate of the variance.

The second options is for the bandwidth selection. By default it is the automatic selection proposed by `?`. It is also possible to choose the automatic selection of `?` by adding `bw=bwNeweyWest` (without quotes because `bwNeweyWest` is a function). A prewhitened kernel estimator can also be computed using the option `prewhite=p`, where p is the order of the vector auto-regressive (VAR) used to compute it. By default, it is set to `FALSE`. `?` show that a prewhitened kernel estimator improves the properties of hypothesis tests on parameters.

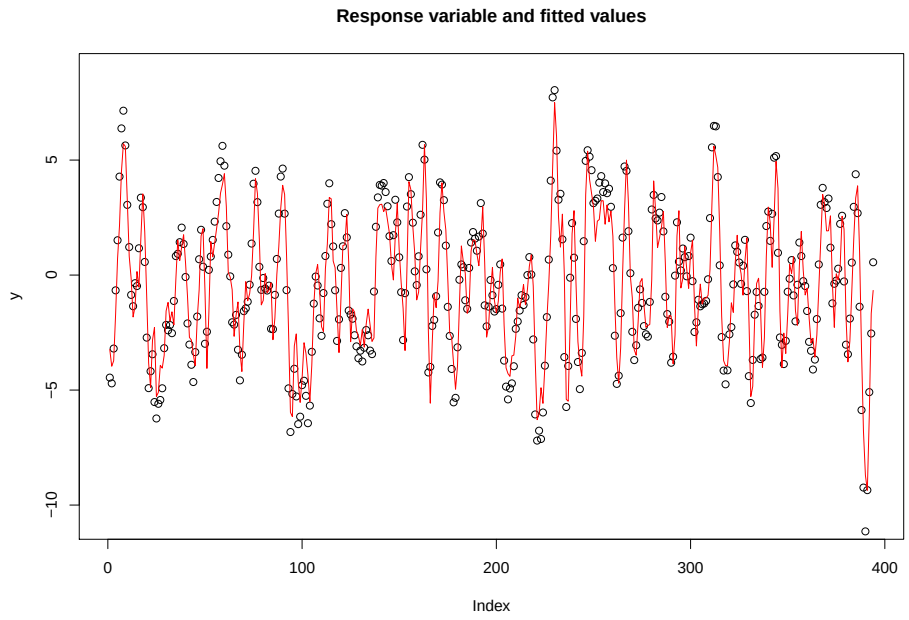
Finally, the `plot` method can be applied to `gmm` objects to do a Q-Q plot of the residuals:

```
> plot(res,which=2)
```



or to plot the observations with the fitted values:

```
> plot(res,which=3)
```



3.5. Estimating a system of equations: CAPM

We want to test one of the implications of the capital asset pricing model (CAPM). This example comes from **?**. It shows how to apply the **gmm** package to estimate a system of equations. The theory of CAPM implies that $\mu_i - R_f = \beta_i(\mu_m - R_f)\ \forall i$, where μ_i is the expected value of stock i's return, R_f is the risk free rate and μ_m is the expected value of the market porfolio's return. The theory can be tested by running the following regression:

$$(R_t - R_f) = \alpha + \beta(R_{mt} - R_f) + \epsilon_t,$$

where R_t is a $N \times 1$ vector of observed returns on stocks, R_{mt} if the observed return of a proxy for the market portfolio, R_f is the interest rate on short term government bonds and ϵ_t is a vector of error terms with covariance matrix Σ_t. When estimated by ML or LS, Σ is assumed to be fixed. However, GMM allows ϵ_t to be heteroskedastic and serially correlated. One implication of the CAPM is that the vector α should be zero. It can be tested by estimating the model with $(R_{mt} - R_f)$ as instruments, and by testing the null hypothesis $H_0 :\ \alpha = 0$.

The data, which are included in the package, are the daily returns of twenty selected stocks from January 1993 to February 2009, the risk-free rate and the three factors of Fama and French[4]. The following test is performed using the returns of 5 stocks and a sample size of 500[5].

```
> data(Finance)
> r <- Finance[1:500,1:5]
```

[4]The symbols of the stocks taken from http://ca.finance.yahoo.com/ are ("WMK", "UIS", "ORB", "MAT", "ABAX","T", "EMR", "JCS", "VOXX", "ZOOM", "ROG", "GGG", "PC", "GCO", "EBF", "F", "FNM", "NHP", "AA", "TDW"). The four other series can be found on K. R. French's web site: http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html

[5]The choice of sample size is arbitrary. The purpose is to show how to estimate a system of equations not to test the CAPM. Besides, the β's seem to vary over time. It is therefore a good practice to estimate the model using short periods.

```
> rm <- Finance[1:500,"rm"]
> rf <- Finance[1:500,"rf"]
> z <- as.matrix(r-rf)
> zm <- as.matrix(rm-rf)
> res <- gmm(z~zm,x=zm)
> coef(res)

 WMK_(Intercept)  UIS_(Intercept)  ORB_(Intercept)  MAT_(Intercept)
    -0.006175863     -0.040071898      0.034540959      0.030904524
ABAX_(Intercept)           WMK_zm           UIS_zm           ORB_zm
    -0.100401093      0.265160711      1.191251310      1.468351230
          MAT_zm           ABAX_zm
     0.944597286      0.945732586

> R <- cbind(diag(5),matrix(0,5,5))
> c <- rep(0,5)
> linearHypothesis(res,R,c,test = "Chisq")

Linear hypothesis test

Hypothesis:
WMK_((Intercept) = 0
UIS_((Intercept) = 0
ORB_((Intercept) = 0
MAT_((Intercept) = 0
ABAX_((Intercept) = 0

Model 1: restricted model
Model 2: z ~ zm

  Df  Chisq Pr(>Chisq)
1
2  5 0.6678     0.9847
```

where the asymptotic chi-square is used since the default distribution requires a normality assumption. The same test could have been performed using the names of the coefficients:

```
> test <- paste(names(coef(res)[1:5])," = 0",sep="")
> linearHypothesis(res,test)
```

Another way to test the CAPM is to estimate the restricted model ($\alpha = 0$), which is over-identified, and to perform a J-test. Adding -1 to the formula removes the intercept. In that case, a column of ones has to be added to the matrix of instruments:

```
> res2<-gmm(z~zm-1,cbind(1,zm))
> specTest(res2)
```

```
 ##  J-Test: degrees of freedom is 5  ##

                J-test   P-value
Test E(g)=0:    0.66770  0.98469
```

which confirms the non-rejection of the theory.

3.6. Testing the CAPM using the stochastic discount factor representation

In some cases the theory is directly based on moment conditions. When it is the case, testing the validity of these conditions becomes a way of testing the theory. **?** present several GMM applications in finance and one of them is the stochastic discount factor (SDF) representation of the CAPM. The general theory implies that $E(m_t R_{it}) = 1$ for all i, where m_t is the SDF and R_{it} the gross return $(1 + r_{it})$. It can be shown that if the CAPM holds, $m_t = \theta_0 + \theta_0 R_{mt}$ which implies the following moment conditions:

$$E\Big[R_{it}(\theta_0 - \theta_1 R_{mt}) - 1\Big] = 0 \quad \text{for} \quad i = 1, ..., N$$

which can be tested as follows:

```
> g5 <- function(tet, x) {
+      gmat <- (tet[1] + tet[2] * (1 + c(x[, 1]))) * (1 + x[, 2:6]) -  1
+      return(gmat)
+  }
> res_sdf <- gmm(g5, x = as.matrix(cbind(rm, r)), c(0, 0))
> specTest(res_sdf)

 ##  J-Test: degrees of freedom is 3  ##

                J-test   P-value
Test E(g)=0:    0.64794  0.88537
```

which is consistent with the two previous tests.

3.7. Estimating continuous time processes by discrete time approximation

This last example also comes from **?**. We want to estimate the coefficients of the following continuous time process which is often used in finance for interest rates:

$$dr_t = (\alpha + \beta r_t)dt + \sigma r_t^{\gamma} dW_t,$$

where W_t is a standard Brownian motion. Special cases of this process are the Brownian motion with drift ($\beta = 0$ and $\gamma = 0$), the Ornstein-Uhlenbeck process ($\gamma = 0$) and the Cox-Ingersoll-Ross or square root process ($\gamma = 1/2$). It can be estimated using the following discrete time approximation:

$$r_{t+1} - r_t = \alpha + \beta r_t + \epsilon_{t+1}$$

with

$$E_t \epsilon_{t+1} = 0, \quad \text{and} \quad E_t(\epsilon_{t+1}^2) = \sigma^2 r_t^{2\gamma}$$

Notice that ML cannot be used to estimate this model because the distribution depends on γ. In particular, it is normal for $\gamma = 0$ and gamma for $\gamma = 1/2$. It can be estimated by GMM using the following moment conditions:

$$E[g(\theta, x_t)] \equiv E \begin{pmatrix} \epsilon_{t+1} \\ \epsilon_{t+1} r_t \\ \epsilon_{t+1}^2 - \sigma^2 r_t^{2\gamma} \\ (\epsilon_{t+1}^2 - \sigma^2 r_t^{2\gamma}) r_t \end{pmatrix} = 0$$

The related `g` function, with $\theta = \{\alpha, \beta, \sigma^2, \gamma\}$ is:

```
> g6 <- function(theta, x) {
+      t <- length(x)
+      et1 <- diff(x) - theta[1] - theta[2] * x[-t]
+      ht <- et1^2 - theta[3] * x[-t]^(2 * theta[4])
+      g <- cbind(et1, et1 * x[-t], ht, ht * x[-t])
+      return(g)
+  }
```

In order to estimate the model, the vector of interest rates needs to be properly scaled to avoid numerical problems. The transformed series is the annualized interest rates expressed in percentage. Also, the starting values are obtained using LS and some options for `optim` need to be modified.

```
> rf <- Finance[,"rf"]
> rf <- ((1 + rf/100)^(365) - 1) * 100
> dr <- diff(rf)
> res_0 <- lm(dr ~ rf[-length(rf)])
> tet0 <- c(res_0$coef, var(residuals(res_0)), 0)
> names(tet0) <- c("alpha", "beta", "sigma^2", "gamma")
> res_rf <- gmm(g6, rf, tet0, control = list(maxit = 1000, reltol = 1e-10))
> coef(res_rf)

       alpha         beta      sigma^2        gamma
 0.010674189 -0.002068407  0.006490192  0.459478821
```

3.8. Comments on models with panel data

The **gmm** package is not directly built to easily deal with panel data. However, it is flexible enough to make it possible in most cases. To see that, let us consider the following model (see **?** for more details):

$$y_{it} = x_{it}\beta + a_i + \epsilon_{it} \quad \text{for} \quad i = 1, ..., N \quad \text{and} \quad t = 1, ..., T,$$

where x_{it} is $1 \times k$, β is $k \times 1$, ϵ_{it} is an error term and a_i is an unobserved component which is specific to individual i. If a_i is correlated with x_{it}, it can be removed by subtracting the average of the equation over time, which gives:

$$(y_{it} - \bar{y}_i) = (x_{it} - \bar{x}_i)\beta + (\epsilon_{it} - \bar{\epsilon}_i) \quad \text{for} \quad i = 1, ..., N \quad \text{and} \quad t = 1, ..., T,$$

which can be estimated by `gmm`. For example, if there are 3 individuals the following corresponds to the GMM fixed effects estimation:

```
> y <- rbind(y1-mean(y1),y2-mean(y2),y3-mean(y3))
> x <- rbind(x1-mean(x1),x2-mean(x2),x3-mean(x3))
> res <- gmm(y~x,h)
```

However, if a_i is not correlated with x_{it}, the equation represents a random effects model. In that case, it is more efficient not to remove a_i from the equation because of the information it carries about the individuals. The error terms are then combined in a single one, $\eta_{it} = (a_i + \epsilon_{it})$ to produce the linear model:

$$y_{it} = x_{it}\beta + \eta_{it}$$

This model cannot be efficiently estimated by OLS because the presence of the common factor a_i at each period implies that η_{it} is serially correlated. However, GMM is well suited to deal with such specifications. The following will therefore produce a GMM random effects estimation:

```
> y <- rbind(y1,y2,y3)
> x <- rbind(x1,x2,x3)
> res <- gmm(y~x,h)
```

The package **plm** of **?** offers several functions to manipulate panel data. It could therefore be combined with **gmm** when estimating such models. It also offers a way to estimate them with its own GMM algorithm for panel data.

3.9. GMM and the sandwich package

In the **gmm** package, the estimation of the optimal weighting matrices are obtained using the **sandwich** package of **?**. For example, the weighting matrix of the two-step GMM defined as:

$$W = \left[\lim_{n\to\infty} Var(\sqrt{n}\bar{g})\right]^{-1}$$

is estimated as follows:

```
> gt <- g(t0, x)
> V <- kernHAC(lm(gt~1),sandwich = FALSE)
> W <- solve(V)
```

where $t0$ is any consistent estimate. As long as the optimal matrix is used, the covariance matrix of the coefficients can be estimated as follows:

$$(\hat{G}'\hat{V}\hat{G})^{-1}/n,$$

where $\hat{G} = d\bar{g}(\hat{\theta})/d\theta$ and $\hat{V}$ is obtained using `kernHAC()`. It is not a sandwich covariance matrix and is computed using the `vcov()` method included in **gmm**. However, if any other weighting matrix is used, say W, the estimated covariance matrix of the coefficients must then be estimated as follows:

$$(\hat{G}'W\hat{G})^{-1}\hat{G}'W\hat{V}W\hat{G}(\hat{G}'W\hat{G})^{-1}/n.$$

A `bread()` and `estfun()` methods are available for `gmm` objects which allows to compute the above matrix using the **sandwich** package. The `bread()` method computes $(\hat{G}'W\hat{G})^{-1}$ while the `estfun()` method returns a $T \times q$ matrix with the t^{th} row equals to $g(\hat{\theta}, x_t)W\hat{G}$. The `meatHAC()` method applied to the latter produces the right meat. Let us consider the example of section (3.4). Suppose we want to use the identity matrix to eliminate one source of bias, at the cost of lower efficiency. In that case, a consistent estimate of the covariance matrix is

$$(\hat{G}'\hat{G})^{-1}\hat{G}'\hat{V}\hat{G}(\hat{G}'\hat{G})^{-1}/n,$$

which can be computed as:

```
> print(res<-gmm(g4,x5t[,4:7],wmatrix="ident"))

Method
 twoStep

Objective function value:  0.002559527

(Intercept)     x5t[, 2]     x5t[, 3]
  -0.087257     1.285166    -0.530806

> diag(vcovHAC(res))^.5

[1] 0.08814116 0.18227836 0.12303848
```

which is more robust than using `vcov()`:

```
> diag(vcov(res))^.5

(Intercept)    x5t[, 2]    x5t[, 3]
 0.08837765  0.18296414  0.12348230
```

Notice that it is possible to fixe W. Therefore, the above results can also be obtained as:

```
> print(res<-gmm(g4,x5t[,4:7], weightsMatrix = diag(5)))

Method
 One step GMM with fixed W

Objective function value:  0.002559527

(Intercept)     x5t[, 2]     x5t[, 3]
  -0.087257     1.285166    -0.530806
```

In this case, the choice of the type of GMM is irrelevant since the weighting matrix is fixed.

4. Generalized empirical likelihood

The GEL is a new family of estimation methods which, as GMM, is based on moment conditions. It follows **?** who developed the idea of empirical likelihood estimation which was meant to improve the confidence regions of estimators. We present here a brief discussion on the method without going into too much details. For a complete review, see **?**, **?** or **?**.

The estimation is based on

$$E(g(\theta_0, x_i)) = 0,$$

which can be estimated in general by

$$\tilde{g}(\theta) = \sum_{i=1}^{n} p_i g(\theta, x_i) = 0,$$

where p_i is called the implied probability associated with the observation x_i. For the GEL method, it is assumed that $q > p$ because otherwise it would correspond to GMM. Therefore, as it is the case for GMM, there is no solution to $\bar{g}(\theta) = 0$. However, there is a solution to $\tilde{g}(\theta) = 0$ for some choice of the probabilities p_i such that $\sum_i p_i = 1$. In fact, there is an infinite number of solutions since there are $(n + q)$ unknowns and only $q + 1$ equations. GEL selects among them the one for which the distance between the vector of probabilities p and the empirical density $1/n$ is minimized. The empirical likelihood of **?** is a special case in which the distance is the likelihood ratio. The other methods that belong to the GEL family of estimators use different metrics. If the moment conditions hold, the implied probabilities carry a lot of information about the stochastic properties of x_i. For GEL, the estimations of the expected value of the Jacobian and the covariance matrix of the moment conditions, which are required to estimate θ, are based on p_i while in GMM they are estimated using $1/n$. **?** show that this difference explains partially why the second order properties of GEL are better.

Another difference between GEL and GMM is how they deal with the fact that $g(\theta, x_i)$ can be a conditionally heteroskedastic and weakly dependent process. GEL does not require to compute explicitly the HAC matrix of the moment conditions. However, if it does not take it into account, its estimators may not only be inefficient but may also fail to be consistent. **?** proposes to replace $g(\theta, x_i)$ by:

$$g^w(\theta, x_i) = \sum_{s=-m}^{m} w(s) g(\theta, x_{i-s})$$

where $w(s)$ are kernel based weights that sum to one (see also **?** and **?**). The sample moment conditions become:

$$\tilde{g}(\theta) = \sum_{i=1}^{n} p_i g^w(\theta, x_i) = 0 \tag{5}$$

The estimator is defined as the solution to the following constrained minimization problem:

$$\begin{aligned}
\hat{\theta}_n &= \arg\min_{\theta, p_i} \sum_{i=1}^{n} h_n(p_i), && (6)\\
&\text{subject to} && (7)\\
&\sum_{i=1}^{n} p_i g^w(\theta, x_i) = 0 \quad \text{and} && (8)\\
&\sum_{i=1}^{n} p_i = 1, && (9)
\end{aligned}$$

where $h_n(p_i)$ has to belong to the following Cressie-Read family of discrepancies:

$$h_n(p_i) = \frac{[\gamma(\gamma+1)]^{-1}[(np_i)^{\gamma+1} - 1]}{n}.$$

? showed that the empirical likelihood method (EL) of **?** ($\gamma = 0$) and the exponential tilting of **?** ($\gamma = -1$) belong to the GEL family of estimators while **?** show that it is also the case for the continuous updated estimator of **?** ($\gamma = 1$). What makes them part of the same GEL family of estimation methods is the existence of a dual problem which is defined as:

$$\hat{\theta} = \arg\min_{\theta} \left[\max_{\lambda} P_n(\theta, \lambda) = \frac{1}{n} \sum_{i=1}^{n} \rho\left(\lambda' g^w(\theta, x_i)\right) \right] \tag{10}$$

where λ is the Lagrange multiplier associated with the constraint (8) and $\rho(v)$ is a strictly concave function normalized so that $\rho'(0) = \rho''(0) = -1$. It can be shown that $\rho(v) = \ln(1-v)$ corresponds to EL , $\rho(v) = -\exp(v)$ to ET and to CUE if it is quadratic.

The equivalence of the primal and dual problems can easily be verified by showing that they both share the same following first order conditions:

$$\sum_{i=1}^{n} p_i g^w(\theta, x_i) = 0, \tag{11}$$

$$\sum_{i=1}^{n} p_i \lambda' \left(\frac{\partial g^w(\theta, x_i)}{\partial \theta} \right) = 0, \tag{12}$$

with

$$p_i = \frac{1}{n} \rho'\left(\lambda' g^w(\theta, x_i)\right). \tag{13}$$

Equation (10) represents a saddle point problem. The solution is obtained by solving simultaneously two optimization problems. We can solve for θ by minimizing $P_n(\theta, \lambda(\theta))$, where $\lambda(\theta)$ is the solution to $\arg\max_\lambda P_n(\theta, \lambda)$ for a given θ. Therefore an optimization algorithm needs to be called inside the $P_n(\theta, \lambda)$ function. It makes the GEL very hard to implement numerically. For example, **?**, who analyzes the small sample properties of GEL, uses an iterative procedure based on the Newton method for λ and a grid search for θ in order to confidently reach the absolute minimum. Using such iterative procedures for λ makes the problem less computationally demanding and does not seem to affect the properties of the estimator of θ_0.

Indeed, **?** show that going beyond two iterations for λ does not improve the second order asymptotic properties of the estimator of θ_0.

The function `gel` offers two options. By default, $\lambda(\theta)$ is obtained by the following iterative method:

$$\lambda_l = \lambda_{l-1} - \left[\frac{1}{n}\sum_{i=1}^{n}\rho''(\lambda'_{l-1}g_t)g_t g'_t\right]^{-1}\left[\frac{1}{n}\sum_{i=1}^{n}\rho'(\lambda'_{l-1}g_i)g_i\right]$$

starting with $\lambda = 0$, which corresponds to its asymptotic value. The algorithm stops when $\|\lambda_l - \lambda_{l-1}\|$ reaches a certain tolerance level. The second option is to let `optim` solve the problem. Then, as for `gmm`, the minimization problem is solved either by `optim`, `nlminb` or `constrOptim`.

In order to test the over-identifying restrictions, **?** proposes three tests which are all asymptotically distributed as a χ^2_{q-p}. The first one is the J-test:

$$n\bar{g}^w(\hat{\theta})'[\hat{\Omega}(\hat{\theta})]^{-1}\bar{g}^w(\hat{\theta}),$$

the second is a Lagrange multiplier test (LM):

$$LM = n\hat{\lambda}'\hat{\Omega}(\hat{\theta})\hat{\lambda}$$

and the last one is a likelihood ratio test (LR):

$$LR = 2\sum_{i=1}^{n}\left[\rho\left(\hat{\lambda}'g^w(\hat{\theta}, x_i)\right) - \rho(0)\right]$$

5. GEL with R

5.1. Estimating the parameters of a normal distribution

For this example, we can leave the option `smooth` at its default value, which is `FALSE`, because of the iid properties of x. A good starting value is very important for GEL. The best choice is the sample mean and the standard deviation. By default the option `type` is set to `EL`. The same methods that apply to `gmm` objects, can also be applied to `gel` objects.

```
> tet0 <- c(mu = mean(x1), sig = sd(x1))
> res_el <- gel(g1,x1,tet0)
> summary(res_el)

Call:
gel(g = g1, x = x1, tet0 = tet0)


Type of GEL:  EL

Kernel:
```

```
Coefficients:
     Estimate  Std. Error  t value   Pr(>|t|)
mu    3.99342   0.13111    30.45750   0.00000
sig   1.85533   0.09030    20.54722   0.00000

Lambdas:
           Estimate  Std. Error  t value   Pr(>|t|)
Lambda[1]  -0.68604   0.29237    -2.34650   0.01895
Lambda[2]  -0.14129   0.06022    -2.34650   0.01895
Lambda[3]  -0.01179   0.00503    -2.34649   0.01895

 Over-identifying restrictions tests: degrees of freedom is 1
         statistics  p-value
LR test  5.051897    0.024599
LM test  5.506010    0.018951
J test   5.506010    0.018951

Convergence code for the coefficients:  0

Convergence code for the lambdas:  0
```

Each Lagrange multiplier represents a shadow price of the constraint implied by moment condition. A binding constraint will produce a multiplier different from zero. Therefore, its value informs us on the validity of the moment condition. In the above results, the λ's are significantly different from zero which would normally suggest that the moment conditions associated with them are violated. As a result, the LM test also rejects the null hypothesis since it is based on the λ's. Notice that `summary` reports two convergence codes, one for λ et another for θ.

The ET and CUE estimates can be obtained as follows:

```
> res_et <- gel(g1,x1,tet0,type="ET")
> coef(res_et)

      mu      sig
3.982436 1.819943

> res_cue <- gel(g1,x1,tet0,type="CUE")
> coef(res_cue)

      mu      sig
3.940642 1.781967
```

A fourth method is available which is called the exponentially tilted empirical likelihood (ETEL) and was proposed by **?**. However, it does not belong to the family of GEL estimators. It solves the problem of misspecified models. In such models there may not exist any pseudo

value to which $\hat{\theta}$ converges as the sample size increases. ETEL uses the $\rho()$ of ET to solve for λ and the $\rho()$ of EL to solve for θ. `?` shows that ETEL shares the same asymptotic properties of EL without having to impose restrictions on the domain of $\rho(v)$ when solving for λ.

```
> res_etel <- gel(g1,x1,c(mu=1,sig=1),type="ETEL")
> coef(res_etel)
```

```
      mu      sig
4.634932 2.419449
```

The type ETEL is experimental for now. Although it is supposed to be more stable because no restrictions are required to solve for λ, once we substitute $\lambda(\theta)$ in the EL objective function to estimate θ, we still need to restrict $\lambda' g_t$ to avoid having NA's. The solution used in gel() is to obtain $\lambda(\theta)$ with `constrOptim` with the restriction $\lambda' gt > 1$ even if it is not required by ET ($\rho(v) = -\exp(v)$). It is however sensitive to starting values. That's the reason why we used different ones above.

5.2. Estimating the AR coefficients of an ARMA process

Because the moment conditions are weakly dependent, we need to set the option `smooth=TRUE`. Before going to the estimation procedure, we need to understand the relationship between the smoothing kernel and the HAC estimator. The reason why we need to smooth the moment function is that GEL estimates the covariance matrix of $\bar{g}(\theta, x_t)$, as if we had iid observations, using the expression $(1/T)\sum_{t=1}^{T}(g_t g_t')$. We can show that substituting g_t by g_t^w in this expression results in an HAC estimator. However, the relationship between the smoothing kernel and the kernel that appears in the HAC estimator is not obvious. For example, we can show that if the smoothing kernel is Truncated, then the kernel in the HAC estimator is the Bartlett. Let us consider the truncated kernel with a bandwidth of 2. This implies that $w(s) = 1/5$ for $|s| \leq 2$ and 0 otherwise. Then, the expression for the covariance matrix becomes:

$$\begin{aligned}
\frac{1}{T}\sum_{t=1}^{T} g_t^w (g_t^w)' &= \frac{1}{T}\sum_{t=1}^{T}\left(\sum_{s=-2}^{2}\frac{1}{5}g_{t+s}\right)\left(\sum_{l=-2}^{2}\frac{1}{5}g_{t+l}'\right), \\
&= \frac{1}{25}\sum_{s=-2}^{2}\sum_{l=-2}^{2}\left(\frac{1}{T}\sum_{t=1}^{T} g_{t+s}g_{t+l}'\right), \\
&= \frac{1}{25}\sum_{s=-2}^{2}\sum_{l=-2}^{2}\hat{\Gamma}_{s-l}, \\
&= \frac{1}{25}\sum_{s=-4}^{4}(5-|s|)\hat{\Gamma}_s, \\
&= \sum_{s=-4}^{4}\left(\frac{1}{5}-\frac{|s|}{25}\right)\hat{\Gamma}_s, \\
&= \sum_{s=-T+1}^{T-1} k_5(s)\hat{\Gamma}_s,
\end{aligned}$$

where $k_5(s)$ is the Bartlett kernel with a bandwidth of 5 defined as

$$K_5(s) = \begin{cases} 1/5 - |s|/25 & \text{if} \quad |s| \leq 5 \\ 0 & \text{otherwise} \end{cases} .$$

See **?** for more details. The model will therefore be estimated using the kernel Truncated. The GMM estimate with the identity matrix is selected as starting value.

```
> tet0 <- gmm(g4,x=x5t[,4:7],wmatrix="ident")$coef
> res <- gel(g4,x=x5t[,4:7],tet0,smooth=TRUE,kernel="Truncated")
> summary(res)

Call:
gel(g = g4, x = x5t[, 4:7], tet0 = tet0, smooth = TRUE, kernel = "Truncated")


Type of GEL:  EL

Kernel:

Coefficients:
             Estimate  Std. Error  t value   Pr(>|t|)
(Intercept)  -0.10018   0.09236    -1.08464   0.27808
x5t[, 2]      1.25067   0.13016     9.60840   0.00000
x5t[, 3]     -0.51150   0.10130    -5.04943   0.00000

Lambdas:
                           Estimate  Std. Error  t value   Pr(>|t|)
h.(Intercept)               0.00505   0.01228     0.41086   0.68117
h.x5t.x5t.x5t.lag(x5, -i)  -0.00029   0.05035    -0.00574   0.99542
h.x5t.x5t.lag(x5, -i)       0.02762   0.16399     0.16840   0.86626
h.x5t.lag(x5, -i)          -0.06938   0.23074    -0.30069   0.76365
h.lag(x5, -i)               0.05694   0.13024     0.43720   0.66197

 Over-identifying restrictions tests: degrees of freedom is 2
         statistics  p-value
LR test  0.22900     0.89181
LM test  0.22850     0.89204
J test   0.23321     0.88994

Convergence code for the coefficients:  0

Convergence code for the lambdas:  0
```

The `specTest` method applied to a `gel` object computes the three tests proposed by **?**:

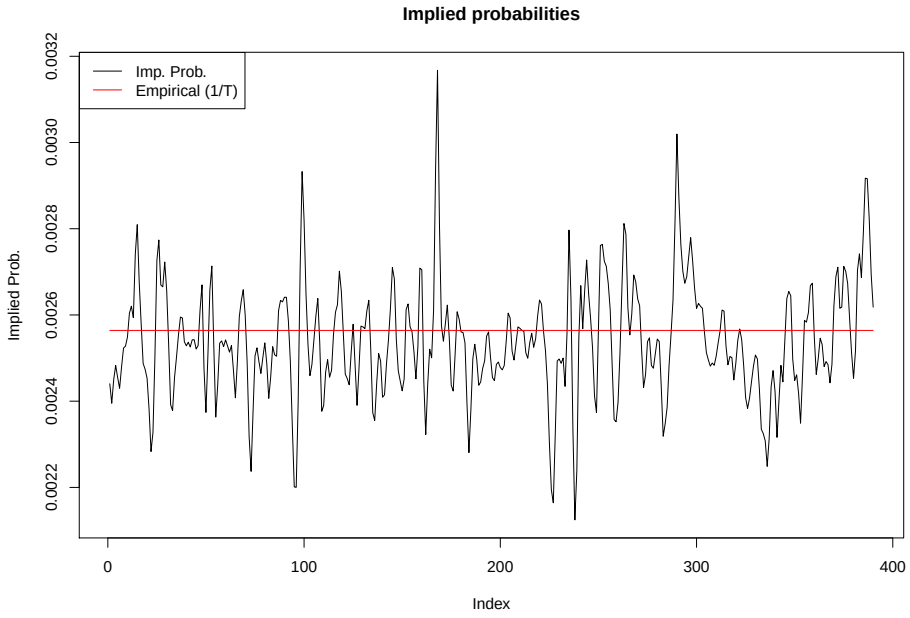
```
> specTest(res)
```

```
 ##  Over-identifying restrictions tests: degrees of freedom is 2  ##

         statistics  p-value
LR test  0.22900     0.89181
LM test  0.22850     0.89204
J test   0.23321     0.88994
```

The `plot` method produces one more graphics when applied to a `gel` object. It shows the implied probabilities along with the empirical density $(1/T)$. It allows to see which observations have more influence:

```
> plot(res,which=4)
```



We can also select `optfct="nlminb"` or `constraint=TRUE` in order to impose restrictions on the coefficients. The former sets lower and upper bounds for the coefficients, while the latter imposes linear constraints using the algorithm `constrOptim`. In this example we want the sum of the AR coefficients to be less than one. `constrOptim` imposes the constraint $ui\theta - ci \geq 0$. Therefore, we need to set:

```
> ui=cbind(0,-1,-1)
> ci <- -1
```

and rerun the estimation as

```
> res <- gel(g4,x=dat5[,4:7],tet0,smooth=TRUE,kernel="Truncated",
+ constraint=TRUE, ui=ui,ci=ci)
```

The result, which is not shown, is identical.

They are also many option to compute the λ's. From version 1.4-0, the default algorithm is `nlminb` because the gradient and Hessian matrix are well defined analytically which speed up convergence. The other choices are `optim` or "iter" which uses a Newton method to solve the first order condition. If the option optlam is set to "optim" and the type is EL, `contrOptim` is selected automatically to restrict $\lambda' g_t$ to be less than 1. It is also possible to change the default values in the control list of the optimizer with the option LambdaControl (see `?nlminb` or `?optim`). Here are some examples:

```
> res <- gel(g4,x=dat5[,4:7],tet0,smooth=TRUE, optlam="optim")
> res <- gel(g4,x=dat5[,4:7],tet0,smooth=TRUE, optlam="optim",
+ LambdaControl=list(trace=TRUE, parscale=rep(.1,5)))
```

5.3. Comments

The GEL method is very unstable numerically. This fact has been reported many times in the recent literature. The method has been included in the **gmm** package because recent theoretical evidence suggests that it may produce better estimators than GMM. Because R is an open source statistical package, it offers a good platform to experiment with numerical properties of estimators.

6. Conclusion

The **gmm** package offers complete and flexible algorithms to estimate models by GMM and GEL. Several options are available which allow to choose among several GMM and GEL methods and many different HAC matrix estimators. In order to estimate the vector of parameters, users can select their preferred optimization algorithm depending on whether inequality constraints are required. For the vector of Lagrange multiplier of GEL, it can be computed by an iterative procedure based on the Newton method which increases the speed of convergence and reduce the instability of the estimation procedure. It could then easily be used by those who are interested in studying the numerical properties of both methods.

The package also offers an interface which is comparable to the least squares method `lm`. Linear model are estimated using formula and methods such as `summary`, `vcov`, `coef`, `confint`, `plot`. `residuals` or `fitted` are available for the objects of class `gmm` and `gel`. R users will therefore have little difficulty in using the package.

7. Computational Details

The package **gmm** is written entirely in R and S3-classes with methods are used. It can be found on the comprehensive R archive network (CRAN,http://CRAN.R-project.org/). It is also hosted on R-Forge (http://r-forge.r-project.org/projects/gmm). It is shipped with a NAMESPACE. The version used to produce this paper is 1.4-0. It depends on the **sandwich** package of **?**, which is used to compute de HAC matrices. The packages **car** (**?**), **mvtnorm** (**?**), **fBasics** (**?**), **MASS** (**?**), **timeDate** (**?**) and **timeSeries** (**?**) are suggested in order to reproduce the examples.

Acknowledgments

I am grateful to the three anonymous referees of the Journal of Statistical Software for great comments on the paper and the package. I also want to thank Achim Zeileis for his suggestions regarding the way the **sandwich** package can be used within `gmm`.

Affiliation:

Pierre Chausé
Department of Economics
University of Waterloo
Waterloo (Ontario), Canada
E-mail: pierre.chausse@uqam.ca
URL: http://www.er.uqam.ca/nobel/k34115/