Generalized Empirical Likelihood with R

Pierre Chaussé

November 6, 2019

Abstract

This an extention of the gmmS4 vignette, to explain how to use the package for generalized empirical likelihood estimation.

Contents

1 A very brief review of the GEL method

We present how to use the package to estimate models by the Generalized Empirical Likelihood method (GEL) (see Newey and Smith (2004) for the iid case and Anatolyev (2005) for weakly dependent processes). We assume that the reader has read the gmmS4 vignette in which many classes and methods needed are defined. We first describe the method without going into too much details. The author can refer to the above papers for a detailed description, or Chaussé (2010) who explains GEL estimation using the gmm (Chaussé (2019))package.

The estimation is based on the following moment conditions

$$\mathrm{E}[g_i(\theta)] = 0,$$

For the iid case, the estimator is defined as the solution to either

$$\hat{\theta} = \arg\min_{\theta, p_i} \sum_{i=1}^n h_n(p_i)$$

subject to,

$$\sum_{i=1}^n p_i g_i(\theta) = 0$$

and

$$\sum_{i=1}^n p_i = 1,$$

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

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

or

$$\hat{\theta} = \arg\min_{\theta} \left[\max_{\lambda} \frac{1}{n} \sum_{i=1}^n \rho\left(\lambda' g_i(\theta)\right) \right] \tag{1}$$

The first is the primal and the second is the dual problem, the latter being preferred in general to define GEL estimators. The vector λ is the Lagrange multiplier associated with the first constraint in the primal problem. Its estimator plays an important role in testing the validity of the moment conditions. $\rho(v)$ 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 Empirical Likelihood (EL) of Owen (2001) , $\rho(v) = -\exp(v)$ to the Exponential Tilting (ET) of Kitamura and Stutzer (1997), and $\rho(v) = (-v - v^2/2)$ to the Continuous Updated GMM estimator (CUE) of Hansen et al. (1996). In the context of GEL, the CUE is also known at the Euclidean Empirical Likelihood (EEL), because it corresponds to $h_n(p_i)$ being the Euclidean distance.

Once the solution is obtained for θ and λ, the implied probabilities can be computed as follows

$$\hat{p}_i = \frac{\rho'(\hat{\lambda}' g_i(\hat{\theta}))}{\sum_{j=1}^n \rho'(\hat{\lambda}' g_j(\hat{\theta}))} \tag{2}$$

If we relax the iid assumption, the problem is identical, but the moment function must be smoothed using a kernel method. Smith (2001) proposes to replace $g_i(\theta)$ by:

$$g_i^w(\theta) = \sum_{s=-m}^{m} w(s) g_{i-s}(\theta)$$

where $w(s)$ are kernel based weights that sum to one (see also Kitamura and Stutzer (1997) and Smith (2001).

2 An S4 class object for GEL models

All classes for GEL models inherit directly from "gmmModels" classes. It is just a gmmModel class with two additional slots: "gelType" and "wSpec". The first is a list with the name of the GEL method and a function for $\rho(v)$, if not already available in the package. The second slot is used to store information about the smoothing of $g_i(\theta)$ in the case of non-iid observations. As for gmmModels, "gelModels" is a union class for "linearGel", "nonlinearGel", "formulaGel" and "functionGel". They inherit directly from the associated GMM model class. For example, "linearGel" is a class that contains a "linearGmm" class object and the two additional slots. The constructor is the gelModel() function. We use the same models presented in the gmmS4 vignette:

```
library(gmm4)
data(simData)
```

- Linear models:

```
lin <- gelModel(y~x1+x2, ~x2+z1+z2, data=simData, vcov="iid", gelType="EL")
lin

## GEL Model: Type  EL
## *******************************
## Moment type: linear
## No Smoothing required
##
## Number of regressors: 3
## Number of moment conditions: 4
## Number of Endogenous Variables: 1
## Sample size:  50
```

- Formula-type models:

```
theta0=c(mu=1,sig=1)
x <- simData$x1
dat <- data.frame(x=x, x2=x^2, x3=x^3, x4=x^4)
gform <- list(x~mu,
              x2~mu^2+sig,
              x3~mu^3+3*mu*sig,
              x4~mu^4+6*mu^2*sig+3*sig^2)
form <- gelModel(gform, NULL, gelType="EEL", theta0=theta0, vcov="MDS", data=dat)
form

## GEL Model: Type  EEL
## *******************************
## Moment type: formula
## No Smoothing required
##
## Number of regressors: 2
## Number of moment conditions: 4
## Number of Endogenous Variables: 0
## Sample size:  50
```

- function:

```
fct <- function(theta, x)
    cbind(x-theta[1], (x-theta[1])^2-theta[2],
          (x-theta[1])^3, (x-theta[1])^4-3*theta[2]^2)
dfct <- function(theta, x)
    {
```

```
        m1 <- mean(x-theta[1])
        m2 <- mean((x-theta[1])^2)
        m3 <- mean((x-theta[1])^3)
        matrix(c(-1, -2*m1, -3*m2, -4*m3,
                 0, -1, 0, -6*theta[2]), 4, 2)
    }
theta0=c(mu=1,sig2=1)
func <- gelModel(fct, simData$x3, theta0=theta0, grad=dfct, vcov="iid", gelType="ET")
func

## GEL Model: Type  ET
## *******************************
## Moment type: function
## No Smoothing required
##
## Number of regressors: 2
## Number of moment conditions: 4
## Sample size:  50
```

- Non-linear models:

```
theta0=c(b0=1, b1=1, b2=1)
gform <- y~exp(b0+b1*x1+b2*x2)
nlin <- gelModel(gform, ~z1+z2+z3+x2, theta0=theta0, vcov="MDS", data=simData,
                 gelType="HD")
nlin

## GEL Model: Type  HD
## *******************************
## Moment type: nonlinear
## No Smoothing required
##
## Number of regressors: 3
## Number of moment conditions: 5
## Number of Endogenous Variables: 2
## Sample size:  50
```

It is also posssible to convert an existing gmmModels to gelModels:

```
nlin <- gmmModel(gform, ~z1+z2+z3+x2, theta0=theta0, vcov="MDS", data=simData)
nlin <- gmmToGel(nlin, gelType="HD")
```

2.1 The rhoXXX functions

The package provides $\rho(v)$ for ET, EL, EEL and HD. The function names are respectively "rhoET", "rhoEL", "rhoEEL" and "rhoHD". For any other GEL, users can pass user specific $\rho(v)$ function to the rhoFct argument of the gelModel(). The following example shows how the function must be built:

```
rhoEL

## function (gmat, lambda, derive = 0, k = 1)
## {
##     lambda <- c(lambda) * k
##     gmat <- as.matrix(gmat)
##     gml <- c(gmat %*% lambda)
##     switch(derive + 1, log(1 - gml), -1/(1 - gml), -1/(1 - gml)^2)
## }
```

```
## <bytecode: 0x558336cfcc18>
## <environment: namespace:gmm4>
```

Therefore, the function must be in the form $f(X, \lambda, d, k)$, where the first argument is an $n \times q$ matrix, the second argument is a vector of q elements, the third is an integer that indicates the order of derivative to return, and the last is a scalar that depends on the kernel used to smooth the moment function. We will discuss that in more details below. The function must return the vector $\rho(kX\lambda)$, $\rho'(kX\lambda)$ or $\rho''(kX\lambda)$, when derive is 0, 1, or 2, where $\rho'(v)$ and $\rho''(v)$ are the first and second derivative of $\rho(v)$.

2.2 The Lagrange multiplier solver

The function getLambda() function solve the maximization problem

$$\max_{\lambda} \frac{1}{n} \sum_{i=1}^{n} \rho(\lambda' X_i)$$

It is used to solve problem (1). The arguments are:

```r
args(getLambda)

## function (gmat, lambda0 = NULL, gelType = NULL, rhoFct = NULL,
##     tol = 1e-07, maxiter = 100, k = 1, method = "BFGS", algo = c("nlminb",
##         "optim", "Wu"), control = list())
## NULL
```

The first argument is the $n \times q$ matrix X. The second argument is the starting value for λ. If set to NULL, a vector of zeroes is used. The third argument is the GEL type, which is either "ET", "EL", "EEL", "REEL" or "HD". If the rhoFct is provided, getType is ignored. The argument control is used to pass a list of arguments to optim(), nlminb() or constrOptim(). The argument "k" is the scalar described in the previous subsection. To describe all other arguments, we are presenting the options by type of GEL:

- EL: There are three possible options for EL. The default is "nlminb", in which case, the arguments tol, maxiter and method are ignored. If algo is set to "optim", the solution is obtained using constrOptim(), with the restriction $(1 - \lambda' X_i) > 0$ for all i. If algo is set to "Wu", the Wu (2005) algorithm is used. The argument tol and maxit are used to control the stopping rule.
- HD: Identical to EL, except that the "Wu" algorithm is not available for that type of GEL.
- EEL: There is an analytical solution, so all other arguments are ignored.
- REEL: This is EEL with a non-negativity constraint on all implied probabilities. In that case, the maxiter can be used to control the number of iterations.
- Others: When rhoFct is provided or the type is ET, the solution is obtained by either "nlminb" or "optim". In that case, the algorithms are controlled through the control argument.

Here are a few example using the simulated dataset (convergence code of 0 means that the algorithm converged with no detected problem):

```r
X <- simData[c("x3","x4","z5")]
(res <- getLambda(X, gelType="EL"))$lambda

## [1] -0.04178415 -0.06745195 -0.02075063

res$convergence$convergence

## [1] 0
```

```r
(res <- getLambda(X, gelType="EL", algo="Wu"))$lambda

## [1] -0.04178415 -0.06745195 -0.02075063

res$convergence$convergence

## [1] 0
```

```r
(res <- getLambda(X, gelType="ET",control=list(maxit=2000)))$lambda

## [1] -0.04171694 -0.06804520 -0.02120255

res$convergence$convergence

## [1] 0
```

The following shows that we can provide getLambda() with a rhoFct instead:

```r
(res <- getLambda(X, rhoFct=rhoEEL))$lambda

## [1] -0.04119158 -0.06769691 -0.02148683

res$convergence$convergence

## [1] 0
```

Although we rarely call the function directly, it is good to understand how it works, because it is an important part of the estimation procedure.

3 Methods for gelModels Classes

We saw above that any "gelModels" is a class that contains one of the "gmmModels" class object. Therefore, many "gmmModels" methods can be applied to "gelModels" through this direct inheritance. when it is the case, we will specify "gmmModels inherited method".

- *kernapply*: In the case of weakly dependent moment conditions, we saw above that the moment function must be smoothed using the following expression:

  $$g_t^w(\theta) = \sum_{s=-m}^{m} w(s) g_{t-s}(\theta)$$

  When a GEL model is defined with vcov="HAC", the specification of the kernel is stored in the "wSpec" slot of the object. For example, we can define the linear model above with the HAC specification:

  ```r
  linHAC <- gelModel(y~x1+x2, ~x2+z1+z2, data=simData, vcov="HAC", gelType="EL")
  linHAC

  ## GEL Model: Type  EL
  ## *******************************
  ## Moment type: linear
  ## Smoothing: Truncated kernel and Andrews bandwidth (1.413)
  ## Number of regressors: 3
  ## Number of moment conditions: 4
  ## Number of Endogenous Variables: 1
  ## Sample size:  50
  ```

  The optimal bandwidth is computed when the model is created, and remains the same during the estimation process, unless another one is specified. The above model shows that the default kernel is the "Truncated" one, and the default bandwidth is based on Andrews (1991). The bandwidth is not based on the smoothing kernel, but on the implied kernel for the HAC estimation. Smith (2001) shows that when $g_t(\theta)$ is replaced by $g_t^w(\theta)$, $V = \sum_{i=1}^n g_t^w(\theta)g_t^w(\theta)'/n$ is an HAC estimator of the covariance matrix of $\sqrt{n}\bar{g}(\theta)$, with Bartlett kernel when the smoothing kernel is the Truncated, and with Parzen kernel when the smoothing kernel is the Bartlett. The optimal bandwidth above is therefore based on the Bartlett kernel.

It is possible to modify the specifications of the kernel and bandwidth through the argument vcovOptions (See help(vcovHAC) from the sandwich package for all possible options). Notice that the kernel type that is passed is the kernel used for the HAC estimation, not the smoothing of $g_t(\theta)$. See in the following example that the Parzen kernel is selected, which implies a Bartlett kernel for the smoothing of $g_t(\theta)$.

```
linHAC2 <- gelModel(y~x1+x2, ~x2+z1+z2, data=simData, vcov="HAC",
                    gelType="EL",
                    vcovOptions=list(kernel="Parzen", bw="NeweyWest", prewhite=1))
linHAC2

## GEL Model: Type  EL
## *******************************
## Moment type: linear
## Smoothing: Bartlett kernel and NeweyWest bandwidth (4.736)
## Number of regressors: 3
## Number of moment conditions: 4
## Number of Endogenous Variables: 1
## Sample size:  50
```

It is also possible to set the bandwidth to a fix number:

```
linHAC3 <- gelModel(y~x1+x2, ~x2+z1+z2, data=simData, vcov="HAC",
                    gelType="EL",
                    vcovOptions=list(kernel="Parzen", bw=3, prewhite=1))
linHAC3

## GEL Model: Type  EL
## *******************************
## Moment type: linear
## Smoothing: Bartlett kernel and Fixed  bandwidth (3)
## Number of regressors: 3
## Number of moment conditions: 4
## Number of Endogenous Variables: 1
## Sample size:  50
```

The *kernapply* method, which is defined as an S3 method in the stats package, uses the information contained in the "wSpec" slot to compute the $n \times q$ matrix of moment with the ith row being $g_t^w(\theta)'$. A theta is required, because we need to evaluate $g_t(\theta)$.

```
gw <- kernapply(linHAC, theta=c(1,1,1))$smoothx
head(gw)

##   (Intercept)         x2         z1         z2
## 2  -10.098571 -124.39829 -10.587446 -30.814727
## 3   -9.941592 -122.40371 -12.084930 -30.151628
## 4   -6.403760  -48.78897  -1.999816 -17.523432
## 5   -6.022626  -37.63422  -1.821833  -8.477109
## 6   -8.604733  -62.82294  -6.329339 -12.859888
## 7   -9.562394  -78.47840  -7.311563 -15.964863
```

The function also returns the weights, bandwidth, kernel name and the scalars k_1 and k_2 that are needed for the asymptotic properties of the estimators (see Anatolyev (2005)). If the argument smooth is set to FALSE, the optimal bandwidth is computed and no smoothing is done. By default, the first step GMM estimator with the identity matrix is used, unless theta is provided.

```
kernapply(linHAC, smooth=FALSE)

## $k
## [1] 2 2
##
## $w
## unknown
## coef[-1] = 0.3333
## coef[ 0] = 0.3333
## coef[ 1] = 0.3333
##
## $bw
## [1] 1.412821
##
## $kernel
## [1] "Truncated"
```

There is also a *kernapply* method for "gmmModels" classes. For now, it only returns the optimal bandwidth, the weights, the kernel names and the k_i's.

```
kernapply(as(linHAC, "gmmModels"))

## $k
## [1] 2 2
##
## $w
## unknown
## coef[-1] = 0.3333
## coef[ 0] = 0.3333
## coef[ 1] = 0.3333
##
## $bw
## [1] 1.412821
##
## $kernel
## [1] "Truncated"
```

- *residuals (gmmModels inherited method)*: Only defined for linearGel and nonlinearGel. It returns $\varepsilon(\theta)$:
- *Dresiduals (gmmModels inherited method)*: Only for "linearGel" and "nonlinearGel", it returns the $n \times k$ matrix $d\varepsilon(\theta)/d\theta$:
- *model.matrix (gmmModels inherited method)*: For linearGel and nonlinearGel only. For both classes, it ca be used to get the matrix of instruments:
- *modelResponse (gmmModels inherited method)*: For linear model only, it returns the vector of response. It is not defined for nonlinearGel classes because the left hand side is not always defined.
- *"[" (gmmModels inherited method)*: It creates a new object of the same class with a subset of moment conditions:
- *as*: "linearGel" can be converted into a "nonlinearGel" or "functionGmm". Also, "nonlinearGel" and "formulaGel" can be converted to "functionGel"

```
as(lin, "nonlinearGel")

## GEL Model: Type  EL
```

```
## *******************************
## Moment type: nonlinear
## No Smoothing required
##
## Number of regressors: 3
## Number of moment conditions: 4
## Number of Endogenous Variables: 1
## Sample size:  50
```

```r
as(nlin, "functionGel")
```

```
## GEL Model: Type  HD
## *******************************
## Moment type: function
## No Smoothing required
##
## Number of regressors: 3
## Number of moment conditions: 5
## Sample size:  50
```

- *subset (gmmModels inherited method)*: As for the S3 method, it creates the same class of object with a subset of the sample.
- *evalMoment*: It computes the $n \times q$ matrix of moments, with the i^{th} row being $g_i(\theta)'$, when the "vcov" slot is not "HAC", and $g_i^w(\theta)'$ when it is.

```r
gt <- evalMoment(linHAC, theta=1:3)
```

- *evalDMoment*: It computes the $p \times k$ matrix of derivatives of the sample mean of $g_i(\theta)$. It calls the next method when the slot "vcov" is not "HAC". Otherwise, a numerical derivative is computed using numericDeric().
- *momentVcov*: It calls the next method when the slot "vcov" is not "HAC". Otherwise, it returns

$$V = \frac{1}{n}\sum_{i=1}^{n} g_t^w(\theta) g_t^w(\theta)'$$

- *update*: This method is used to modify existing objects. For now, only the covariance structure, the gelType and rhoFct can be modified.

```r
update(lin, vcov="HAC", gelType="ET")
```

```
## GEL Model: Type  ET
## *******************************
## Moment type: linear
## Smoothing: Truncated kernel and Andrews bandwidth (1.413)
## Number of regressors: 3
## Number of moment conditions: 4
## Number of Endogenous Variables: 1
## Sample size:  50
```

4 Restricted models

As for "gmmModels", restrictions can be imposed on the coefficients. The union class for all restricted GEL models is "rgelModels". The classes are "rlinearGel", "rformulaGel", "rfunctionGel" and "rnonlinearGel". Each one contains its unrestricted model plus additional slots that specify the constraints. See the gmmS4 vignette for more details. The constructor is the *restModel* method.

```r
lin2 <- gelModel(y~x1+x2+x3+z1, ~x1+x2+z1+z2+z3+z4, data=simData,
                 gelType="EL", vcov="MDS")
rlin2 <- restModel(lin2, c("x1=x2", "x3=0"))
rlin2

## GEL Model: Type  EL
## *******************************
## Moment type: rlinear
## No Smoothing required
##
## Number of regressors: 3
## Number of moment conditions: 7
## Number of Endogenous Variables: 1
## Sample size:  50
## Constraints:
##   x1 - x2 = 0
##   x3 = 0
## Restricted regression:
##  y = (Intercept)+(x1+x2)+z1
```

All methods described in the previous section also apply to the restricted models. For example,

```r
e <- residuals(rlin2, theta=c(1,1,1)) ## we now have 3 coefficients
gt <- evalMoment(rlin2, theta=c(1,1,1))
```

To recover the full coefficient vector, we use the *coef* method:

```r
coef(rlin2, theta=1:3)

## (Intercept)          x1          x2          x3          z1
##           1           2           2           0           3
```

5 The *solveGel* Method

The main method to estimate a model by GEL methods is *solveGel*. The available signatures are:

```r
showMethods("solveGel")

## Function: solveGel (package gmm4)
## object="gelModels"
## object="linearGel"
##     (inherited from: object="gelModels")
```

The arguments are

- theta0: The initial value for the minimization algorithm. It is required if the model does not have a theta0 slot. If it has a theta0 slot, it is used by default unless a new theta0 is provided.
- lambda0: The initial value to pass to the lambda solver. By default, it is set to 0, which is its asymptotic value in case of correctly specified models.
- lamSlv: An optional function to solve for lambda. By default, getLambda() is used. The function must have the following form:

```r
mylSolve <- function(gmat, lambda0, gelType=NULL, rhoFct=NULL, k=1, ...)
    {
        lambda <-  rep(0,ncol(gmat))
        obj <-  sum(colMeans(gmat)^2)
        list(lambda=lambda, convergence=0, obj=obj)
    }
```

Therefore, it must return a list with lambda, convergence and obj. In the above example, λ is set to a vector of zeros and the returned obj is $\bar{g}(\theta)'\bar{g}(\theta)$. The solution will therefore be the one step GMM with the weighting matrix equals to the identity.

```
solveGel(lin,c(0,0,0), lamSlv=mylSolve)

## $theta
##  (Intercept)           x1           x2
## -0.226153103  1.009952150  0.002381384
##
## $convergence
## [1] 0
##
## $lambda
## (Intercept)          x2          z1          z2
##           0           0           0           0
##
## $lconvergence
## [1] 0
```

To present a more realistic example, suppose we want to estimate the model using the exponentially tilted empirical likelihood (ETEL) method of Schennach (2007), we could write a function that solves the lambda using ET, and returns the empirical log-likelihood ratio:

```
mylSolve <- function(gmat, lambda0=NULL, gelType=NULL, rhoFct=NULL, k=1, ...)
    {
        gmat <- as.matrix(gmat)
        res  <-  getLambda(gmat, lambda0, gelType="ET", k=k)
        gml <- c(gmat%*%res$lambda)
        w <- exp(gml)
        w <- w/sum(w)
        n <- nrow(gmat)
        res$obj <- mean(-log(w*n))
        res
    }
etelFit <- solveGel(lin,c(1,1,0), lamSlv=mylSolve)
etelFit$theta

## (Intercept)          x1          x2
##   1.3266917   0.8504703  -0.1030545
```

That's equivalent to setting gelType to "ETEL":

```
solveGel(update(lin, gelType="ETEL"), c(1,1,0))$theta

## (Intercept)          x1          x2
##   1.3266917   0.8504703  -0.1030545
```

- coefSlv: A character string that indicates the name of the minimization solver used to obtain $\hat{\theta}$. By default, "optim" is use. The other options are "nlminb" and "constrOptim".
- lControl: A list of options for the lambda solver. It is passed to getLambda() or lamSlv() if provided. For example, we can use the Wu method for EL as follows:

```
solveGel(lin, c(1,1,0), lControl=list(algo="Wu"))$theta

## (Intercept)          x1          x2
##   1.3272620   0.8505003  -0.1031604
```

- tControl: A list of control for the coefSlv function. We could, for example, use the following options:

```r
solveGel(lin, c(1,1,0), lControl=list(algo="Wu"),
         tControl=list(method="BFGS", control=list(maxit=2000, reltol=1e-9)))$theta

## (Intercept)          x1          x2
##   1.3271920   0.8505091  -0.1031434
```

In that particular case, the list is directly passed to optim().

The method returns a list with: theta=$\hat{\theta}$, lambda=$\hat{\lambda}$, convergence = convergence message and code for θ, and lconvergence = convergence message and code for λ.

6 The *modelFit* Method

This is the main estimation method. It returns an object of class "gelfit". The arguments are:

- object: Any object that belongs to the union class "gelModels"
- getType: Optional type of GEL if we want to estimate the model with a type that is different from the one defined in the object.
- rhoFct: Optional $\rho(v)$ function if we want to estimate the model with a function that is different from the one defined in the object.
- initTheta: Method to obtain the starting value for θ. By default, the one step GMM is used. The other option is to use the one included in the object.
- theta0: The stating value for θ. If provided, the argument initTheta is ignored.
- lambda0: Starting value for λ. By default, a vector of zeros is used.
- vcov: Logical argument that specifes if the covariance matrices of $\hat{\theta}$ and $\hat{\lambda}$ should be computed? It is FALSE by default.
- ... : Additional argument that is passed to the *gelSolve* method.

In general, it works fine with the default arguments:

```r
modelFit(lin)

## GEL Model: Type  EL
## *******************************
## Moment type: linear
## No Smoothing required
##
## Number of regressors: 3
## Number of moment conditions: 4
## Number of Endogenous Variables: 1
## Sample size:  50
##
## Estimation:  EL
## Convergence Theta:  0
## Convergence Lambda:  0
## coefficients:
## (Intercept)          x1          x2
##   1.3270715   0.8505249  -0.1031374
## lambdas:
## (Intercept)          x2          z1          z2
##  0.03175148  0.01673016 -0.02751524 -0.07550335
```

References

S. Anatolyev. Gmm, gel, serial correlation, and asymptotic bias. *Econometrica*, 73:983–1002, 2005.

D. W. K. Andrews. Heteroskedasticity and autocorrelation consistent covariance matrix estimation. *Econometrica*, 59:817–858, 1991.

Pierre Chaussé. Computing generalized method of moments and generalized empirical likelihood with r. *Journal of Statistical Software*, 34(11):1–35, 2010. URL `http://www.jstatsoft.org/v34/i11/`.

Pierre Chaussé. *gmm: Generalized Method of Moments and Generalized Empirical Likelihood*, 2019. R package version 1.6-3.

L. P. Hansen, J. Heaton, and A. Yaron. Finit-sample properties of some alternative gmm estimators. *Journal of Business and Economic Statistics*, 14:262–280, 1996.

Y. Kitamura and M. Stutzer. An information-theoretic alternative to generalized method of moments estimation. *Econometrica*, 65(5):861–874, 1997.

W. K. Newey and R. J. Smith. Higher order properties of gmm and generalized empirical likelihood estimators. *Econometrica*, 72:219–255, 2004.

A. B. Owen. *Empirical Likelihood*. Chapman and Hall, 2001.

S. M. Schennach. Point estimation with exponentially tilted empirical likelihood. *Econometrica*, 35(2): 634–672, 2007.

R. J. Smith. Gel criteria for moment condition models. *Working Paper, University of Bristol*, 2001.

C. Wu. Algorithms and R codes for the pseudo empirical likelihood method in survey sampling. *Survey Methodology*, 31(2):239, 2005.

Appendix

A Some extra codes