Instrumental Variables Tools for the Case of Weak or Many Instruments

Pierre Chausse*

Abstract

This vignette explains the different tools included in the package to deal with the weak or the many instruments problem. For example, it presents estimation methods like the LIML or its modified version proposed by Fuller (1977) method and some improved inference methods for TSLS and GMM. It is in early stage of development, so comments and recommendations are welcomed.

Important: This document is incomplete (so is the package for what is covered here).

1 The model

We only consider linear models for the moment. Let the following be the model of interest:

$$y = X_1\beta_1 + X_2\beta_2 + u \equiv X\beta + u$$

where y and u are $n \times 1$, X_1 is $n \times k_1$, X_2 is $n \times k_2$, β_1 is $k_1 \times 1$, β_2 is $k_2 \times 1$, X is $n \times k$ and β is $k \times 1$, with $k = k_1 + k_2$. We assume that the intercept is included in X_1. Suppose that X_2 is the matrix of endogenous variables. Then, we want to instrument them with Z_2, a $n \times l_2$ matrix, where $l_2 \geq k_2$. The matrix of exogenous variables that are included and excluded is $Z = [X_1, Z_2]$, a $n \times q$ matrix with $q = k_1 + l_2$. The reduced form for X_2, or the first stage regression, is therefore:

$$X_2 = Z\Pi + e\,,$$

where Π is $q \times k_2$ and e is $n \times k_2$.

2 K-class Estimator and LIML

The K-Class methods need to be added to the package if we want to develop tools for models with weak and/or many instruments. The reason is that estimations and tests based on the limited information maximum likelihood (LIML), which is K-Class method, has shown to perform well in these cases.

To my knowledge, many of the methods proposed here have not been implemented in R yet. However, some procedures are implemented in the `ivmodel` package of Kang et al. (2023). Some of our procedures have been influenced by the package, so we use it when needed to compare our results.

2.1 The method

A K-Class estimator is the solution to

$$X'(I - \kappa M_z)(y - X\beta) = 0\,,$$

*University of Waterloo, pchausse@uwaterloo.ca

where $M_z = I - P_z$ and P_z is the projection matrix $Z(Z'Z)^{-1}Z'$. It is therefore represented as a just-identified IV with the instrument $W_\kappa = (I - \kappa M_z)X$. Note that $M_z X_1 = 0$, which implies the following matrix of instruments:

$$\begin{aligned} W_\kappa &= \begin{bmatrix}(I-\kappa M_z)X_1 & (I-\kappa M_z)X_2\end{bmatrix} \\ &= \begin{bmatrix}X_1 & (I-\kappa M_z)X_2\end{bmatrix} \\ &= \begin{bmatrix}X_1 & (X_2 - \kappa\hat{e})\end{bmatrix} \end{aligned},$$

where $\hat{e} = M_z X_2$ is the matrix of residuals from the first stage regression. Note that the model is just-identified only when $l_2 > k_2$. The above representation is just a convenient way of defining the method. In fact, we can also represent the two-stage least squares (TSLS) method , over-identified or not, as a just-identified IV with $W = [X_1 \hat{X}_2]$, where $\hat{X}_2 = P_z X_2 \equiv X_2 - \hat{e}$. Therefore, TSLS is a K-Class estimator with $\kappa = 1$. We can also see that the least squares estimator can be obtained by setting κ to 0. The solution can be written as follows:

$$\hat{\beta}_\kappa = (W_\kappa' X)^{-1} W_\kappa' y\,.$$

We can compute the standard errors using the asymptotic properties of just identified IV. In the case of iid errors (no heteroskedasticity), the variance can be estimated as:

$$\hat{\Sigma}_{\kappa,iid} = \hat{\sigma}^2 (W_\kappa' X)^{-1} W_\kappa' W_\kappa (W_\kappa' X)^{-1}\,,$$

where $\hat{\sigma}^2$ is the estimated variance of u. Note that the bread of the covariance matrix is symmetric, which is not the case in general for just-identified IV. Also, we can simplify the expression to $\hat{\sigma}^2(W_\kappa' X)^{-1}$ only when κ is equal to 0 or 1. For other values it is not possible because $(I - \kappa M_z)(I - \kappa M_z) \neq (I - \kappa M_z)$. In the case of heteroskedastic errors, the covariance matrix can be estimated as follows:

$$\hat{\Sigma}_{\kappa,HC} = (W_\kappa' X)^{-1} \hat{\Omega}_{\kappa,HC} (W_\kappa' X)^{-1}\,,$$

where $\hat{\Omega}_{HC}$ is an HCCM estimator of the variance of $W_\kappa' u$. For example, we can obtain the HC0 estimator with the following $\hat{\Omega}$:

$$\hat{\Omega}_{\kappa,HC0} = \sum_{i=1}^{n} \hat{u}_i^2 W_{\kappa,i} W_{\kappa,i}'\,,$$

where $\hat{u}_i = y_i - X_i' \hat{\beta}_\kappa$.

2.2 The LIML method

We do not justify how κ is defined for the LIML method. For more details, see Davidson and MacKinnon (2004). Let $Y = [y\ X_2]$ be the $n \times (1 + k_2)$ matrix with all endogenous variables from the model. Then, κ_{liml} is defined as the smallest eigenvalue of:

$$(Y' M_z Y)^{-1/2} Y' M_1 Y (Y' M_z Y)^{-1/2}\,,$$

where $M_1 = I - P_1$ and $P_1 = X_1(X_1'X_1)^{-1}X_1'$. We can show that it is equivalent to finding the smallest eigenvalue of $(Y'M_zY)^{-1}Y'M_1Y$. An alternative to the LIML method was proposed by Fuller (1977). The method is also a K-Class method with $\kappa_{ful} = \kappa_{liml} - \alpha/(n - q)$. The Fuller method happens to have better properties than LIML.

2.3 Computing $\hat{\kappa}$

We want to use the data used by Card (1993). The dataset is included in the `ivmodel` package. The endogenous variable is education (`educ`) and the two instruments we consider are `near4` and `near2`. The other included exogenous variables are experience (`exper`), experience squared (`expersq`) and a set of binary variables. In the following, the `ivmodel` object is generate. It contains the `\kappa` for LIML and Fuller:

```
library(ivmodel)
data(card.data)
## from the ivmodel examples
Z <- card.data[,c("nearc4","nearc2")]
Y <- card.data[,"lwage"]
D <- card.data[,"educ"]
Xname <- c("exper", "expersq", "black", "south", "smsa", "reg661",
           "reg662", "reg663", "reg664", "reg665", "reg666", "reg667",
           "reg668", "smsa66")
X <- card.data[,Xname]
mod <- ivmodel(Y=Y,D=D,Z=Z,X=X)
```

We can see the κ's using the following commands:

```
c(LIML=mod$LIML$k, Fuller=mod$Fuller$k)
```

```
##     LIML   Fuller
## 1.000409 1.000075
```

We can create a `linearModel` object with the same specifications as follows. By default, `ivmodel` model assumes homoskedasticity, so we set the argument `vcov` to `"iid"`:

```
library(momentfit)
g <- reformulate(c("educ", Xname), "lwage")
h <- reformulate(c(c("nearc4","nearc2"), Xname))
mod2 <- momentModel(g, h, data=card.data, vcov="iid")
```

The `getK` function generates $\hat{\kappa}$ for the original LIML and the modified one. No effort is done to make it efficient for now. The modified LIML is $\hat{\kappa} - \alpha/(n-k)$, where k is the number of exogenous variables (included and excluded).

We can compare the values with the ones computed by `ivmodel`. They are identical:

```
getK(mod2)
```

```
##     LIML   Fuller
## 1.000409 1.000075
```

Note that the function `getK` has three arguments: `object`, which is the model object, `alpha`, which is use to compute κ_{ful} and `returnRes`. When the latter is set to `TRUE` (the default is `FALSE`), the function returns a list of two elements: the above vector of κ and the matrix of first stage residuals $M_z X_2$. The latter is used by the K-Class function to generate the matrix of instruments W_κ. By setting it to `TRUE`, it avoids having to recompute it.

We can also have more than one endogenous regressor. For this model, we can interact `educ` with, say, `exper`, which is like having a second endogenous variable. The package can recognize that `educ:exper` is endogenous because it is not part of the set of instruments. The following is the new model:

```
g2 <- reformulate(c("educ", "educ:exper", Xname), "lwage")
h2 <- reformulate(c(c("nearc4","nearc2", "nearc2:exper", "nearc4:exper"), Xname))
mod3 <- momentModel(g2, h2, data=card.data)
getK(mod3)
```

```
##     LIML   Fuller
## 1.000702 1.000368
```

Note that $\kappa_{liml} = 1$ for just-identified models. When it is the case, `getK` does not compute the residuals and only returns the vector of κ no matter how we set the argument `returnRes`. The following model is just identified:

```r
h3 <- reformulate(c(c("nearc4"), Xname))
mod4 <- momentModel(g, h3, data=card.data)
getK(mod4)
```

```
##     LIML   Fuller
## 1.000000 0.999666
```

2.4 Computing the K-Class estimators

The function that computes the K-Class estimator is `kclassfit`. The arguments are: `object`, the model object, `k`, the value of κ, `type`, the type of κ to compute when `k` is missing (`"LIML"` or `"Fuller"`) and `alpha`, the parameter of the Fuller method (the default is 1). Note first that the estimator is a TSLS estimator when `k=1` and a LSE when it is equal to 0. The package already has a `tsls` method for `linearModel` objects, which is what `kclassfit` calls when `k=1`. For the LSE, a new method was created to facilitate the estimation of model objects by least squares. The method is `lse`:

```r
lse(mod2)
```

```
## Model based on moment conditions
## *********************************
## Moment type: linear
## Covariance matrix: iid
## Number of regressors: 16
## Number of moment conditions: 17
## Number of Endogenous Variables: 1
## Sample size:  3010
##
## Estimation: Least Squares
##
## Coefficients:
## (Intercept)        educ        exper      expersq        black        south
##   4.7393766   0.0746933   0.0848320   -0.0022870   -0.1990123   -0.1479550
##        smsa      reg661       reg662       reg663       reg664       reg665
##   0.1363845  -0.1185698  -0.0222026    0.0259703   -0.0634942    0.0094551
##      reg666      reg667       reg668       smsa66
##   0.0219476  -0.0005887  -0.1750058    0.0262417
```

It is an object of class `lsefit` that contains the `lm` object from the estimation. Therefore, the `kclassfit` function returns an object of class `lsefit` when `k=0` and `tlsl` when `k=1`. For any other value, which includes LIML and Fuller, the function returns an object of class `kclassfit`. The object contains a `gmmfit` object, generated by the estimation of the artificially created just-identified model, the name of the method, the value of κ and the original model.

```r
(liml <- kclassfit(mod2))
```

```
## Model based on moment conditions
## *********************************
## Moment type: linear
## Covariance matrix: iid
## Number of regressors: 16
```

```
## Number of moment conditions: 17
## Number of Endogenous Variables: 1
## Sample size:  3010
##
## Estimation: LIML (k = 1.000409)
## coefficients:
##  (Intercept)          educ         exper       expersq         black
##  3.221269443   0.164027756   0.121689917  -0.002362359  -0.116870463
##        south          smsa        reg661        reg662        reg663
## -0.142791708   0.097738480  -0.101656724   0.001630403   0.048731041
##       reg664        reg665        reg666        reg667        reg668
## -0.054724308   0.055061606   0.074061888   0.042413909  -0.199985585
##       smsa66
##  0.014116798
```

```
(fuller <- kclassfit(mod2, type="Fuller"))
```

```
## Model based on moment conditions
## *********************************
## Moment type: linear
## Covariance matrix: iid
## Number of regressors: 16
## Number of moment conditions: 17
## Number of Endogenous Variables: 1
## Sample size:  3010
##
## Estimation: Fuller (k = 1.000075)
## coefficients:
##   (Intercept)           educ          exper        expersq          black
##  3.319304e+00   1.582588e-01   1.193098e-01  -2.357495e-03  -1.221749e-01
##         south           smsa         reg661         reg662         reg663
## -1.431251e-01   1.002341e-01  -1.027489e-01   9.134797e-05   4.726123e-02
##        reg664         reg665         reg666         reg667         reg668
## -5.529064e-02   5.211649e-02   7.069652e-02   3.963694e-02  -1.983725e-01
##        smsa66
##  1.489978e-02
```

We see that the LIML and Fuller estimates I get are identical to the ones from the `ivmodel` package.

```
print(mod$LIML$point.est,digits=10)
```

```
##          Estimate
## [1,] 0.1640277561
```

```
print(coef(liml)[2], digits=10)
```

```
##         educ
## 0.1640277561
```

```
print(mod$Fuller$point.est,digits=10)
```

```
##          Estimate
## [1,] 0.1582588323
```

```
print(coef(fuller)[2], digits=10)
```

```
##         educ
## 0.1582588323
```

Note that the argument `k` can be the output of `getK` with `returnRes=TRUE`. This is a way of avoiding recomputing the κ and the first stage residuals. This is useful when we want to compute the LIML and Fuller for the same model. For example, the following is the fast version of what we did above.

```
resK <- getK(mod2, 1, TRUE)
(liml <- kclassfit(mod2, resK))
```

```
## Model based on moment conditions
## *********************************
## Moment type: linear
## Covariance matrix: iid
## Number of regressors: 16
## Number of moment conditions: 17
## Number of Endogenous Variables: 1
## Sample size:  3010
##
## Estimation: LIML (k = 1.000409)
## coefficients:
##  (Intercept)          educ         exper       expersq         black
##  3.221269443   0.164027756   0.121689917  -0.002362359  -0.116870463
##        south          smsa        reg661        reg662        reg663
## -0.142791708   0.097738480  -0.101656724   0.001630403   0.048731041
##       reg664        reg665        reg666        reg667        reg668
## -0.054724308   0.055061606   0.074061888   0.042413909  -0.199985585
##       smsa66
##  0.014116798
```

```
(fuller <- kclassfit(mod2, resK, type="Fuller"))
```

```
## Model based on moment conditions
## *********************************
## Moment type: linear
## Covariance matrix: iid
## Number of regressors: 16
## Number of moment conditions: 17
## Number of Endogenous Variables: 1
## Sample size:  3010
##
## Estimation: Fuller (k = 1.000075)
## coefficients:
##   (Intercept)           educ          exper        expersq          black
##  3.319304e+00   1.582588e-01   1.193098e-01  -2.357495e-03  -1.221749e-01
##         south           smsa         reg661         reg662         reg663
## -1.431251e-01   1.002341e-01  -1.027489e-01   9.134797e-05   4.726123e-02
##        reg664         reg665         reg666         reg667         reg668
## -5.529064e-02   5.211649e-02   7.069652e-02   3.963694e-02  -1.983725e-01
##        smsa66
##  1.489978e-02
```

2.5 Inference

Since the `kclassfit` object contains a just-identified `gmmfit` object, we can do inference as if it was an IV. The `summary` method for `kclassfit` objects is in fact the same as for `gmmfit` objects, but it contains additional information about the original model and the method. It returns an object of class `summaryKclass`.

```r
(s <- summary(liml))
```

```
## Model based on moment conditions
## *********************************
## Moment type: linear
## Covariance matrix: iid
## Number of regressors: 16
## Number of moment conditions: 16
## Number of Endogenous Variables: 1
## Sample size:  3010
##
## Estimation: LIML (k = 1.00040942731651)
## Sandwich vcov: TRUE
## coefficients:
##                Estimate  Std. Error t value  Pr(>|t|)
## (Intercept)  3.22126944  0.98048104  3.2854 0.0010184 **
## educ         0.16402776  0.05763981  2.8457 0.0044309 **
## exper        0.12168992  0.02482322  4.9023 9.474e-07 ***
## expersq     -0.00236236  0.00035189 -6.7133 1.903e-11 ***
## black       -0.11687046  0.05656732 -2.0660 0.0388245 *
## south       -0.14279171  0.02879080 -4.9596 7.063e-07 ***
## smsa         0.09773848  0.03329490  2.9355 0.0033297 **
## reg661      -0.10165672  0.04410858 -2.3047 0.0211838 *
## reg662       0.00163040  0.03468374  0.0470 0.9625071
## reg663       0.04873104  0.03349713  1.4548 0.1457294
## reg664      -0.05472431  0.03968009 -1.3791 0.1678523
## reg665       0.05506161  0.04942349  1.1141 0.2652459
## reg666       0.07406189  0.05544273  1.3358 0.1816059
## reg667       0.04241391  0.05143408  0.8246 0.4095836
## reg668      -0.19998559  0.05348458 -3.7391 0.0001847 ***
## smsa66       0.01411680  0.02278641  0.6195 0.5355691
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##  Anderson and Rubin
##                  Statistics  df   pvalue
## Test E(g)=0:         1.2321   1  0.26699
##
##
## Instrument strength based on the F-Statistics of the first stage OLS
## educ : F( 1 ,  2994 ) =  13.42398  (P-Vavue =  0.0002527353 )
```

Note that the specification test is based on Anderson and Rubin. It is a likelihood ratio test equal to $n\log(\hat{\kappa})$ and is distributed as a chi-square with the degrees of freedom equal to the number of over-identifying restrictions. It calls the `specTest` method for `kclassfit` objects:

```r
specTest(liml)
```

```
##
##  Anderson and Rubin
##                  Statistics  df   pvalue
## Test E(g)=0:         1.2321   1  0.26699
```

We can compare the standard error we get here and the one we get from the `ivmodel` package. Note that only inference about the coefficient of the endogenous variable is provided by `ivmodel`.

```
s@coef["educ",]
```

```
##    Estimate  Std. Error     t value    Pr(>|t|)
## 0.164027756 0.057639810 2.845737283 0.004430873
```

```
mod$LIML$std.err
```

```
##      Std. Error
## [1,] 0.05549507
```

The result is quite different. But we can see why. In the following I recompute the standard error using the formula $\hat{\sigma}^2(W_\kappa' X)^{-1}$. We now get the same result. As mentioned before, this expression is only valid for $\kappa = 1$.

```
spec <- modelDims(mod2)
u <- residuals(liml)
sig <- sum(u^2)/(spec$n-spec$k)
W <- model.matrix(liml@model, "instruments")
myX <- model.matrix(liml@model)
sqrt(diag(sig*solve(t(W)%*%myX)))[2]
```

```
## [1] 0.05549507
```

For Heteroskedastic errors. We have to redefine the models.

```
mod <- ivmodel(Y=Y,D=D,Z=Z,X=X,heteroSE=TRUE)
mod2 <- momentModel(g, h, data=card.data, vcov="MDS")
liml <- kclassfit(mod2, resK)
summary(liml)@coef["educ",]
c(mod$LIML$point.est, mod$LIML$std.err)
```

The above code is not run because the `ivmodel` is very inefficient to compute the meat matrix. It is done using a loop. It you run the code you should get identical point estimate and both standard errors are equal to 0.0576098.

3 Weak Instruments (For later use)

3.1 Data Generating Process

The following function is used to generate dataset with k instruments and different level of strength. The DGP is

$$
\begin{aligned}
y_1 &= \beta y_2 + u \\
y_2 &= \pi' Z + e \,,
\end{aligned}
$$

where $Z \in \mathbb{R}^k$, $\mathrm{Var}(u) = \mathrm{Var}(e) = 1$, $\mathrm{Cor}(e,u) = \rho$, $\pi_i = \eta$ for all $i = 1, ..., k$ and $Z \sim N(0, I)$. The R^2 of the first stage regression is therefore equal to

$$
R^2 = \frac{k\eta^2}{k\eta^2 + 1} \,,
$$

which implies

$$
\eta = \sqrt{\frac{R^2}{k(1 - R^2)}}
$$

We can therefore set R^2 and k and let the function get η.

```
getIVDat <- function(n, R2, k, rho, b0=0)
{
    eta <- sqrt(R2/(k*(1-R2)))
    Z <- sapply(1:k, function(i) rnorm(n))
    sigma <- chol(matrix(c(1,rho,rho,1),2,2))
    err <- cbind(rnorm(n), rnorm(n))%*%sigma
    y2 <- rowSums(Z)*eta+err[,2]
    y1 <- b0*y2 + err[,1]
    dat <- data.frame(y1=y1, y2=y2, u=err[,1], e=err[,2])
    for (i in 1:k) dat[[paste("Z",i,sep="")]] <- Z[,i]
    dat
}
```

```
library(momentfit)
set.seed(112233)
k <- 10
rho <- .3
R2 <- .001
g <- y1~y2
n <- 500
h <- reformulate(paste("Z", 1:k, sep=""))
dat <- getIVDat(n, R2, k, rho)
m <- momentModel(g, h, data=dat, vcov="MDS")
```

References

Davidson, R., and J. G. MacKinnon. 2004. *Econometric Theory and Methods*. New York: Oxford University Press.

Fuller, W. A. 1977. "Some Properties of a Modification of the Limited Information Estimator." *Econometrica* 45: 939–53.

Kang, Hyunseung, Yang Jiang, Qingyuan Zhao, and Dylan Small. 2023. *Ivmodel: Statistical Inference and Sensitivity Analysis for Instrumental Variables Model*. https://CRAN.R-project.org/package=ivmodel.