# 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 $n \times k_2$, $\beta_1$ is $k_1 \times 1$, $\beta_2$ is $k_2 \times 1$, $X$ is $n \times k$ and $\beta$ 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:

$$X_2 = X_1\Pi_1 + Z_2\Pi_2 + e \equiv Z\Pi + e\,,$$

where $\Pi_1$ is $k_1 \times k_2$, $\Pi_2$ is $l_2 \times k_2$, $\Pi$ 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 $\kappa$ 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 $\kappa$ 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 $\kappa$ 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, $\kappa_{liml}$ 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 $(Y' M_z Y)^{-1} Y' M_1 Y$. An alternative to the LIML method was proposed by Fuller (1977). The method is also a K-Class method with $\kappa_{ful} = \kappa_{liml} - \alpha/(n - q)$, where $\alpha$ is parameter. It usually set to=1. 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 $\kappa$'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 $\kappa_{ful}$ 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 $\kappa$ 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 $W_\kappa$. 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 $\kappa$ no matter how we set the argument `returnRes`. The following model is just identified:

```
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 $\kappa$, `type`, the type of $\kappa$ to compute when `k` is missing (`"LIML"`, `"Fuller"` or `"BTSLS"` for the biased corrected TSLE of Nagar (1959)) 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`:

```
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, Fuller and BTSLS ($\kappa = n/(n - l_2 + 2)$), 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 $\kappa$ and the original model.

```
(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
```

```
(btsls <- kclassfit(mod2, type="BTSLS"))
```

```
## 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:  Two-Stage Least Squares
## coefficients:
##   (Intercept)          educ          exper        expersq          black
##  3.3396868121  0.1570593700   0.1188148807  -0.0023564836  -0.1232777953
##         south          smsa         reg661         reg662         reg663
## -0.1431944615  0.1007530001  -0.1029759964  -0.0002286491   0.0469556243
##        reg664         reg665         reg666         reg667         reg668
```

```
## -0.0554083884   0.0515041450   0.0699968047   0.0390595603  -0.1980370807
##        smsa66
##  0.0150625816
```

Note that the biased-adjusted TSLS is just TSLS because the adjustment only affects the method when the number of excluded instruments is not equal to 2. We see in the following 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 $\kappa$ 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`.

```
(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:

```
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

## 3.1 Testing for weak instrument: Stock and Yogo (2005)

This test and the critical values are for model with homoskedastic errors. The test is the smallest eigenvalue of the following expression (Cragg and Donald (1993)):

$$\hat{\Sigma}_e^{-1/2}\Big[X_2'M_1Z_2(Z_2'M_1Z_2)^{-1}Z_2'M_1X_2\Big]\hat{\Sigma}_e^{-1/2} = \hat{\Sigma}_e^{-1/2}[M_1X_2]'[Z_2\hat{\Pi}_2]\hat{\Sigma}_e^{-1/2}$$

where $\hat{\Sigma}_e = \hat{e}'\hat{e}/(n - l_2 - k_1)$. If the number of included endogenous variables $k_2$ is equal to 1, this is just the F statistic for the null hypothesis $H_0 : \Pi_2 = 0$. For $k_2 > 1$, it is a test of rank reduction. Under the null the rank of $\Pi_2$ is $k_2 - 1$ and under the alternative it is equal to $k_2$. The function `CDtest`, which stands for Cragg and Donald test, computes this statistic. By using the `momentStrength` method, which computes the first stage F statistics for each included endogenous variable, we can see they are both the same when $k_2 = 1$:

```
(CD2 <- CDtest(mod2, print=FALSE))
```

```
## [1] 7.893096
```

```
momentStrength(mod2)
```

```
## $strength
##         Stats df1  df2           pv
## educ 7.893096   2 2993 0.0003811364
##
## $mess
## [1] "Instrument strength based on the F-Statistics of the first stage OLS"
```

However, it does not return a p-value like the F-test computed by `momentStrength`. Instead, it comes with the critical values computed by Stock and Yogo (2005). If we let the function `CDtest` print the result (the default), we see the statistics and the critical values that are relevant to our model (they depend on the number of included endogenous and excluded exogenous variables).

```
CDtest(mod2)
```

```
## Cragg and Donald Test for Weak Instruments
## ****************************************
## Number of included Endogenous: 1
## Number of excluded Exogenous: 2
## The test is not robust to heteroskedasticity
## Statistics: 7.893
##
## Stock and Yogo (2005) critical values
## ***********************************
## Target size for TSLS:
##  size=0.1  size=0.15   size=0.2  size=0.25
##     19.93      11.59       8.75       7.25
##
## Target relative bias for Fuller-K:
## bias=0.05   bias=0.1   bias=0.2   bias=0.3
##     15.60      12.38       7.93       6.62
##
## Target size for LIML:
##  size=0.1  size=0.15   size=0.2  size=0.25
##      8.68       5.33       4.42       3.92
```

We reject the null hypothesis that the instruments are weak if the statistic is greater than the critical value of interest. To understand the critical values, let's first consider the ones under "Target size for TSLS". If are willing to accept a wrong size of at most 10% for hypothesis tests on coefficients at 5%, the statistic must exceed 19.93 for the instruments to be considered strong enough. Since the statistic for `mod2` does not, we should expect a higher size distortion. In fact, our statistic is equal to 7.8931, so we can expect the size to be as high as 25% since the statistic is greater than 7.25. Under "Target size for LIML", we have the same critical values but for models estimated by LIML. We see that the size distortion is not as severe for LIML. Since the statistic is between the first and the second critical value, the size should be between 10% and 15%.

We also have critical values that are based on the worst bias relative to the OLS bias. For example, if the model is estimated by the Fuller method and we are willing to accept a relative bias of at most 5%, we need the statistic to exceed 15.60. Since the statistic of `mod2` is only greater than 6.62 (the last critical value), the relative bias may be as large as 30%. Note that the critical values based on the relative bias are only available for TSLS when the number of over-identifying restrictions are greater or equal to 2. For the following model, all critical values are available. In this case, the instruments are very strong. But are they valid?

```
g <- reformulate(c("educ", Xname), "lwage")
h <- reformulate(c(c("nearc4","nearc2","IQ","KWW"), Xname))
mod5 <- momentModel(g, h, data=card.data, vcov="iid")
CDtest(mod5)
```

```
## Cragg and Donald Test for Weak Instruments
## ******************************************
## Number of included Endogenous: 1
## Number of excluded Exogenous: 4
## The test is not robust to heteroskedasticity
## Statistics: 228.2
##
## Stock and Yogo (2005) critical values
## *************************************
## Target relative bias for TSLS:
## bias=0.05   bias=0.1   bias=0.2   bias=0.3
##     16.85      10.27       6.71       5.34
##
## Target size for TSLS:
##  size=0.1  size=0.15   size=0.2  size=0.25
##     24.58      13.96      10.26       8.31
##
## Target relative bias for Fuller-K:
## bias=0.05   bias=0.1   bias=0.2   bias=0.3
##     10.09       8.10       5.36       4.46
##
## Target size for LIML:
##  size=0.1  size=0.15   size=0.2  size=0.25
##      5.44       3.87       3.30       2.98
```

## 3.2 Testing for weak instrument: Sanderson and Windmeijer (2016)

This test was derived for models with at least 2 endogenous variables ($k_2 > 2$ in our model). Let $X_{2,j}$ be the j$^{\text{th}}$ included endogenous variable and and $X_{2,-j}$ be the $k_2 - 1$ remaining included endogenous variables, then the procedure is :

- Estimate the model
$$X_{2,j} = X_{2,-j}\delta_1 + X_1\delta_2 + v$$
by TSLS using the instruments $Z$ and save the residuals $\hat{v}$.

- Estimate the model
$$\hat{v} = X_1\kappa_1 + Z_2\kappa_2 + \xi$$
by OLS
- Compute the F-test for $H_0 : \kappa_2 = 0$. Let $\tilde{F}$ be the value of the statistics.
- Compute the Sanderson and Windmeijer (2016) statistics $F_{j|-j} = \tilde{F}[l_2/(l_2 - k_2 + 1)]$.

To illustrate the procedure, we consider the following model based on the simulated dataset `simData`:

$$y = \beta_0 + \beta_1 y_1 + \beta_2 y_2 + \beta_3 y_3 + \beta_4 x_1 + \beta_5 x_2 + u\,,$$

where `y1`, `y2` and `y3` are assumed to be endogenous. We want to estimate the model using the 5 excluded exogenous variables `z1` to `z5`. To use our notation, we have $X_1 = \{$`x1`, `x2`$\}$, $X_2 = \{$`y1`, `y2`, `y3`$\}$ and $Z_2 = \{$`z1`, `z2`, `z3`, `z4`, `z5`$\}$. Following the above procedure the statistic using `j=1` is:

```
data(simData)
## Step 1
m <- tsls(y1~y2+y3+x1+x2, ~z1+z2+z3+z4+z5+x1+x2, data=simData)
e <- residuals(m)
## Step 2
fit <- lm(e~z1+z2+z3+z4+z5+x1+x2, simData)
fitr <- lm(e~x1+x2, simData)
F <- anova(fit, fitr)$F[2]
## Step 4
(sw1 <- F*5/(5-2))
```

```
## [1] 0.7500098
```

The function `SWtest` computes this test and returns the

```
smod <- momentModel(y~y1+y2+y3+x1+x2, ~z1+z2+z3+z4+z5+x1+x2, data=simData)
SWtest(smod,1,FALSE)
```

```
## [1] 0.7500098
```

Following Sanderson and Windmeijer (2016), for models with $k_2$ endogenous variables and $l_2$ excluded exogenous, we compare the statistic with the Stock and Yogo (2005) critical values for models with $l_2 - 1$ endogeous variables and $k_2 - l_2 + 1$ excluded exogenous. This allows us to test the intruments for models with 3 endogenous variables without generating new tables. In the following, we can see that the critical values are obtained by reducing the number of endogenous variables by 1 and the number of excluded exogenous variables by 2. Clearly, the instruments are weak in this simulated model.

```
SWtest(smod)
```

```
## Sanderson and Windmeijer Test for Weak Instruments
## ***************************************************
## Number of included Endogenous: 3(-1 for the critical values)
## Number of excluded Exogenous: 5(-2 for the critical values)
## The test is not robust to heteroskedasticity
## Statistics: 0.75
##
## Stock and Yogo (2005) critical values
## ***********************************
## Critical value adjusted to Sanderson and Windmeijer (2016) specification
##
## Target size for TSLS:
##  size=0.1  size=0.15   size=0.2  size=0.25
```

```
##     13.43       8.18       6.40       5.45
##
## Target relative bias for Fuller-K:
## bias=0.05   bias=0.1   bias=0.2   bias=0.3
##     11.62       9.21       6.57       5.70
##
## Target size for LIML:
##  size=0.1  size=0.15   size=0.2  size=0.25
##      5.44       3.81       3.32       3.09
```

These critical values are obtained by running the function `SYTables` with the argument `SWcrit` set to `TRUE`. Note that the authors show also that the same critical values can be used if we multiply the Cragg and Donald statistic by $k_2/(k_2 - l_2 + 1)$. It is therefore possible to test for weak instruments in a model with 3 endogenous variables using the `CDtest` function, if we set the argument `SWcrit` to `TRUE`.

### 3.3 Testing for weak instrument: Montiel Olea and Pflueger (2013)

In most applied economic studies, it is unrealistic to assume that the errors are conditionally homoskedastic. When the errors are conditionally heteroskedastic, it is recommended by Andrews, Stock, and Sun (2019) to use the effective F-test of Montiel Olea and Pflueger (2013) (MOP). Assuming that $k_2 = 1$, which is the only option for this test, the procedure is:

- Replace y by $M_1 y$, $X_2$ by $M_1 X_2$ and $Z_2$ by $M_1 Z_2$ and normalize the matrix $Z_2$ so that $Z_2'Z_2/n = I$ (we replace $Z$ by $Q$ from its QR decomposition times $\sqrt{n}$) . Then, the model becomes

$$y = X_2\beta_2 + u$$

  and

$$X_2 = Z_2\Pi_2 + e\,.$$

- Obtain the robust covariance matrix estimate of $\hat{\Pi}_2$, $\hat{W}_2$.
- The test is $F_{eff} = (X_2'Z_2Z_2'X_2)/[n\text{tr}(\hat{W}_2)]$, where $\text{tr}(A)$ is the trace of $A$. Since $\hat{\Pi}_2 = (Z_2'Z_2)^{-1}Z_2'X_2 = Z_2'X_2/n$, we can write the test as $F_{eff} = n\hat{\Pi}_2'\hat{\Pi}_2/\text{tr}(\hat{W}_2)$.

This is computed by the function `MOPtest`. For now, no critical values are reported. Will be added soon.

```
MOPtest(mod2)
```

```
## Montiel Olea and Pflueger Test for Weak Instruments
## ***************************************************
## Simplified Test for TSLS
## Type of LS covariance matrix: iid
## Number of included Endogenous: 1
## Effective degrees of freedom: 2
## x: 10
## Statistics: 7.935
## Critical Value (size=0.05): 19.29
## P-Value: 0.7284
```

We can see that it is close to the non-robust F test:

```
CDtest(mod2, FALSE)
```

```
## [1] 7.893096
```

This is because the model `mod2` is defined with `vcov="iid"`. Therefore, $\hat{W}_2$ is a non-robust covariance matrix. If we want an HCCM estimator, we need to define the model with `vcov="MDS"`. It is also possible to compute the test using HAC estimator if needed. We use the `update` method to change `vcov` and rerun the test. We see that the robust test is a little higher than the non-robust.

```
mod2 <- update(mod2, vcov="MDS")
MOPtest(mod2)
```

```
## Montiel Olea and Pflueger Test for Weak Instruments
## ***************************************************
## Simplified Test for TSLS
## Type of LS covariance matrix: HCCM
## Number of included Endogenous: 1
## Effective degrees of freedom: 1.934279
## x: 10
## Statistics: 8.176
## Critical Value (size=0.05): 19.45
## P-Value: 0.7033
```

The above procedure is the simplified version of the test. We start exploring the generalized test. First, we need an estimate of the matrix $W$. Given the structure of $Z$, the robust covariance matrix of the OLS estimators is the covariance matrix of the moment conditions, because when $Z'Z = I$, the OLS estimator of $y = Z\beta + u$ is $\hat{\beta} = Z'y = \beta + Z'u$. Therefore, the variance of $\hat{\beta}$ is the variance of the moment condition function $Z'u$. The reduced form for our model is:

$$\begin{aligned} y &= X_2\beta_2 + u = Z_2[\Pi_2\beta_2] + v \\ X_2 &= Z_2\Pi_2 + e \end{aligned}$$

For example, we can compute $W_2$ above as follows for `mod2`.

- We extract $Z_2$, $X_1$, $X_2$ and $y$,

```
## get Z
spec <- modelDims(mod2)
Z2 <- model.matrix(mod2, "excludedExo")
X1 <- model.matrix(mod2, "includedExo")
X2 <- model.matrix(mod2, "includedEndo")
y <- modelResponse(mod2)
```

- We project $X_1$ off $Z_2$, $X_2$ and $y$ and normalize $Z$ using its QR decomposition times $\sqrt{n}$:

```
fitX1  <- lm.fit(X1, Z2)
Z2 <- fitX1$residuals
X2 <- qr.resid(fitX1$qr, X2)
y <- qr.resid(fitX1$qr, y)
Z2 <- qr.Q(qr(Z2))*sqrt(nrow(Z2))
```

To compute $\hat{W}_2$, we can use the tools already included in the package. We just need to create a `linearModel` object with no endogenous variables. For $\hat{W}_2$, we regress $X_2$ on $Z_2$ and use $Z_2$ as instruments. We can set the argument `vcov` to `"MDS"` to obtain a robust to heteroskedasticity $\hat{W}_2$ (or to `"CL"` for clustered or `"HAC"` for serially correlated errors).

```
colnames(Z2) = paste("Z", 1:ncol(Z2), sep="")
dat <- as.data.frame(cbind(X2,Z2))
g <- reformulate(colnames(Z2), colnames(X2), FALSE)
h <- reformulate(colnames(Z2), NULL, FALSE)
m2 <- momentModel(g,h,data=dat,vcov="MDS")
```

We need to compute the OLS estimator and then use the `vcov` method for `linearModel` objects to estimate the asymptotic variance of $Z_2'e/\sqrt{n}$:

```
b <- crossprod(Z2,X2)/nrow(Z2)
w2 <- vcov(m2, b)
```

This is the only part of $W$ we need to estimate for the simplified version of the test. For the general test, we also need to estimate $W_1$, which is the asymptotic variance of $Z_2'v/\sqrt{n}$, and $W_{12}$ which is the asymptotic covariance between $Z_2'e/\sqrt{n}$ and $Z_2'v/\sqrt{n}$. This can be done by writing the model as a system of equations with the same regressors and instruments. The above `g` is the second equation, so we need to add the first in a list and put `h` in a list:

```
dat <- as.data.frame(cbind(y=y,X2,Z2))
g <- list(reformulate(colnames(Z2), "y", FALSE),  g)
h <- list(h)
m <- sysMomentModel(g,h,data=dat,vcov="MDS")
b <- list(c(crossprod(Z2,y)/nrow(Z2)),
          c(crossprod(Z2,X2)/nrow(Z2)))
w <- vcov(m, b)
w
```

```
##              Eqn1.Z1      Eqn1.Z2      Eqn2.Z1      Eqn2.Z2
## Eqn1.Z1  0.148439520  0.003538106  0.241860165 -0.002252146
## Eqn1.Z2  0.003538106  0.159419517 -0.002252146  0.278892794
## Eqn2.Z1  0.241860165 -0.002252146  3.504443285  0.010990879
## Eqn2.Z2 -0.002252146  0.278892794  0.010990879  3.762294024
```

Note that we need to adjust the sample size. The way the model `m` is defined, the sample is multiplied by 2. Since we divide by twice the sample size to compute the estimator, we need to multiply the estimated W by 2. We can see that $\hat{W}_2$ is the $2 \times 2$ bottom left block of $\hat{W}$:

```
w2
```

```
##            Z1         Z2
## Z1 3.50444328 0.01099088
## Z2 0.01099088 3.76229402
```

This is what the function `getMOPW` computes. For now, it is not exported, so we need to run it using `momentfit:::`. The function returns the different $\hat{W}$'s separately for convenience. Here we see $\hat{W}_2$:

```
res <- momentfit:::getMOPw(mod2)
res$w2
```

```
##            Eqn2.Z1    Eqn2.Z2
## Eqn2.Z1 3.50444328 0.01099088
## Eqn2.Z2 0.01099088 3.76229402
```

The function also returns the elements `w1`, `w12` and `omega`. The latter is $\hat{\Omega} = [\hat{v}, \hat{e}]'[\hat{v}, \hat{e}]/n$. The matrices $\hat{W}_1$ and $\hat{W}_{12}$ are needed to compute the effective degrees of freedom:

$$K_{eff} = \frac{[\text{tr}(\hat{W}_2)]^2(1+2x)}{\text{tr}(\hat{W}_2'\hat{W}_2)\max \text{eval}(\hat{W}_2)}$$

where $x = B_e(\hat{W}, \hat{\Omega})/\tau$, where $\tau$ is the worst bias relative to the benchmark and $B_e(\hat{W}, \hat{\omega})$ is some function. In the simplified version, the parameter $x$ is equal to $1/\tau$, so only $\hat{W}_2$ is needed. For the generalized test, $x$ depends on $B_e(\hat{W}, \hat{\omega})$ for $e =$ LIML or TSLS, which needs to be computed using a one dimentional optimizer. The function `getMOPx` returns the value of `x`. The main arguments are `w`, which is the output from `getMOPw`, `tau` that we explained above and the type of estimation we want to test the instruments for. As usual, LIML is less biased so it requires a smaller $F_{eff}$ to get the same relative bias as TSLS.

```
momentfit:::getMOPx(w=res, tau=0.10, type="LIML")
```

```
## [1] 5.178162
```

By default, the `MOPtest` function computes the simplified test, which is the one obtained above. For the generalized test, we set the argument `simplified` to `FALSE`.

```
MOPtest(mod2, simplified=FALSE, estMethod="LIML")
```

```
## Warning in MOPtest(mod2, simplified = FALSE, estMethod = "LIML"): The generalized test is for models
##  simplified is changed to TRUE
```

```
## Warning in MOPtest(mod2, simplified = FALSE, estMethod = "LIML"): The simplified test is not defined
## estMethod changed to TSLS
```

```
## Montiel Olea and Pflueger Test for Weak Instruments
## ***************************************************
## Simplified Test for TSLS
## Type of LS covariance matrix: HCCM
## Number of included Endogenous: 1
## Effective degrees of freedom: 1.934279
## x: 10
## Statistics: 8.176
## Critical Value (size=0.05): 19.45
## P-Value: 0.7033
```

The test is the same as above, but the critical value is smaller, which is expected since the simplify test tends to have higher critical value, especially when the number of excluded instruments is small. We can compare the test with the one for TSLS:

```
MOPtest(mod2, simplified=FALSE, estMethod="TSLS")
```

```
## Warning in MOPtest(mod2, simplified = FALSE, estMethod = "TSLS"): The generalized test is for models
##  simplified is changed to TRUE
```

```
## Montiel Olea and Pflueger Test for Weak Instruments
## ***************************************************
## Simplified Test for TSLS
## Type of LS covariance matrix: HCCM
## Number of included Endogenous: 1
## Effective degrees of freedom: 1.934279
## x: 10
## Statistics: 8.176
## Critical Value (size=0.05): 19.45
## P-Value: 0.7033
```

It seems that we are rejecting the weak instrument hypothesis for TSLS and not for LIML. This is surprising. We'll need to investigate.

As mentioned by the authors, the efficient F is the robust F when the model is just identified. The model `mod4` created above is just-identified, but we need to change the argument `vcov` to `"MDS"`:

```
mod4 <- update(mod4, vcov="MDS")
MOPtest(mod4, estMethod="TSLS", print=FALSE)["Feff"]
```

```
##     Feff
## 14.21423
```

We can compare with the first stage F computed by `momentStrength`. As we can see, it is the same as long as we choose the HC0 type.

```
momentStrength(update(mod4, vcovOptions=list(type="HC0")))$strength
```

```
##         Stats df1  df2           pv
## educ 14.21423   1 2994 0.0001662837
```

### 3.4 Testing for weak instruments: Lewis and Mertens (2022)

The authors generalize the MOP test for models with multiple endogenous variables. We first need to compute the matrix $\Phi = (I_{k_2} \otimes \text{vec}(I_{l_2}))'[W_2 \otimes I_{l_2}](I_{k_2} \otimes \text{vec}(I_{l_2}))$. This is clearly not something we want to write explicitely. It is unlikely to encounter very large $l_2$ and $k_2$, but we want to avoid creating large matrices just in case. If we write $W_2$ as a $k_2 \times k_2$ block matrix, with $A_{ij}$ being the $l_2 \times l_2$ matrix in the ith row and jth column, then $\Phi_{ij} = \text{tr}(A_{ij})$. The following shows that the function `psiMat`, which uses this trace argument, produces the right matrix:

```
w2 <- matrix(1:144, 12, 12)
w2[lower.tri(w2)] <- t(w2)[lower.tri(w2)]
k2 <- 3
l2 <- 4
phiMatT <- function(w2, k2, l2)
{
    A <- kronecker(diag(k2), c(diag(l2)))
    B <- kronecker(w2, diag(l2))
    t(A)%*%B%*%A
}

phiMat <- function(w2, k2, l2)
{
    psi <- matrix(NA, k2,k2)
    for (i in 1:k2)
    {
        c0 <- 1+(i-1)*l2
        c1 <- ncol(w2)
        di <- diag(w2[,c0:c1])
        tmp <- colSums(matrix(di, nrow=l2))
        if (i == 1)
        {
            diag(psi) <- tmp
        } else if (i==k2) {
            psi[1,k2] <- psi[k2,1] <- tmp
        } else {
            j <- seq_len(length(tmp))
            j <- cbind(j,j)
            psi[-(1:(i-1)),][j] <- psi[,-(1:(i-1))][j] <- tmp
        }
    }
    psi
}
phiMatT(w2,k2,l2)
```

```
##      [,1] [,2] [,3]
## [1,]   82  274  466
## [2,]  274  290  482
## [3,]  466  482  498
```

```
phiMat(w2,k2,12)
```

```
##      [,1] [,2] [,3]
## [1,]   82  274  466
## [2,]  274  290  482
## [3,]  466  482  498
```

The statistic proposed by the authors is

$$g_{min} = \min \text{eval}[n\hat{\Phi}^{-1/2}\hat{\Pi}_2'\hat{\Pi}_2\hat{\Phi}^{-1/2}]$$

It is clear that this is the MOP effective F test when $k_2 = 1$, because $k_2 = 1$ implies that $\Phi = \text{tr}(W_2)$.

```
LewMertest <- function(object, tau=0.10, size=0.05, print=TRUE,
                       estMethod = c("TSLS", "LIML"), simplified = TRUE,
                       digits = max(3L, getOption("digits") - 3L), ...)
{
    estMethod <- match.arg(estMethod)
    if (!inherits(object, "linearModel"))
        stop("object must be of class linearModel")
    spec <- modelDims(object)
    if (sum(spec$isEndo)<1)
    {
        warning("The model does not contain endogenous variables")
        return(NA)
    }
    Z2 <- model.matrix(object, "excludedExo")
    X1 <- model.matrix(object, "includedExo")
    X2 <- model.matrix(object, "includedEndo")
    y <- modelResponse(object)
    if (!is.null(X1))
    {
        fitX1  <- lm.fit(X1, Z2)
        Z2 <- as.matrix(fitX1$residuals)
        X2 <- qr.resid(fitX1$qr, X2)
        y <- qr.resid(fitX1$qr, y)
    }
    Z2 <- qr.Q(qr(Z2))*sqrt(nrow(Z2))
    colnames(Z2) <- paste("Z", 1:ncol(Z2), sep="")
    colnames(X2) <- paste("endo",1:ncol(X2), sep="")
    b1 <- c(crossprod(Z2,y)/spec$n)
    Pi2 <- crossprod(Z2,X2)/spec$n
    g <- c(reformulate(colnames(Z2), "y", FALSE),
              lapply(colnames(X2), function(ei)
                  reformulate(colnames(Z2), ei, FALSE)))
    h <- reformulate(colnames(Z2), NULL, FALSE)
    dat <- as.data.frame(cbind(y=y,X2,Z2))
    m <- sysMomentModel(g=g, list(h), data = dat, vcov=object@vcov,
                        vcovOptions = object@vcovOptions)
    v <- cbind(y-Z2%*%b1, X2-Z2%*%Pi2)
    omega <- crossprod(v)/nrow(v)
    w <- vcov(m, c(list(b1),lapply(1:ncol(Pi2), function(i) Pi2[,i])))
    w2 <- w[(ncol(Z2)+1):ncol(w),(ncol(Z2)+1):ncol(w)]
    phi <- phiMat(w2, ncol(X2), ncol(Z2))
```

```
    gmin <- eigen(solve(phi, nrow(Z2)*crossprod(Pi2)))$val[1]
    gmin
}
LewMertest(mod2)
```

```
## [1] 8.176379
```

```
LewMertest(mod3)
```

```
## [1] 18.47247
```

### 3.5 Data Generating Process (for later use)

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

$$y_1 = \beta y_2 + u$$
$$y_2 = \pi' Z + e\,,$$

where $Z \in \mathbb{R}^k$, $\text{Var}(u) = \text{Var}(e) = 1$, $\text{Cor}(e, u) = \rho$, $\pi_i = \eta$ 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 $\eta$.

```
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


Andrews, I., J. H. Stock, and L. Sun. 2019. "Weak Instruments in Intrumental Variables Regression: Theory and Practice." *Annual Review of Economics* 11: 727–53.

Cragg, J. G., and S. G. Donald. 1993. "Testing Identifiability and Specification in Instrumental Variable Models." *Econometric Theory* 9: 222–40.

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.

Lewis, D. J., and K. Mertens. 2022. "A Robust Test for Weak Instruments with Multiplle Endogenous Regressors." *FRB of Dallas Working Paper No. 2208*. https://ssrn.com/abstract=4144103.

Montiel Olea, J., and C. Pflueger. 2013. "A Robust Test for Weak Instruments." *Journal of Business and Economic Statistics* 31: 358–69.

Nagar, A. L. 1959. "Testing Identifiability and Specification in Instrumental Variable Models." *Econometrica* 27: 575–95.

Sanderson, E., and F. Windmeijer. 2016. "A Weak Instrument F-Test in Linear Iv Models with Multiple Endogenous Variables." *Journal of Econometrics* 190: 212–21.

Stock, J. H., and M. Yogo. 2005. *Testing for Weak Instruments in Linear Iv Regression. In Identification and Inference for Econometric Models: Essays in Honor of Thomas Rothenberg*. Edited by Donald W. K. Andrews and James H. Stock. Cambridge: Cambridge University Press.