User Guideline for *Correcting Small Sample Bias in Linear Models with Many Covariates*

Miren Azkarate-Askasua and Miguel Zerecero

This document is a guideline to use our bootstrap correction for second order moments and explains the main features of the functions involved in the bootstrap correction. We provide an example of a labor market with low mobility to test the bootstrap correction of the second order moments detailed in *Correcting Small Sample Bias in Linear Models with Many Covariates*.

1 Example

The folder *data* contains the simulated labor market with low mobility and roughly 2.8 million person-year observations. The code *simulate_example.R* replicates the simulation. The logarithm of wages are simulated as:

$$w_{it} = \theta_i + \psi_{J(i,t)} + \delta_{O(i,t)} + q_{it}\gamma + \varepsilon_{it}, \tag{1}$$

where the function $J(i,t)$ gives the identity of the unique firm that employs worker i at time t, θ_i is a worker fixed effect, $\psi_{J(i,t)}$ is the firm $J(i,t)$ fixed effect, $\delta_{O(i,t)}$ is an occupation fixed effect and q_{it} are time varying observables (age and age squared), and ε_{it} is the error term.

2 Function Overview

The code running the example correction is *main_example.m* in the main folder where we initialize the parallelization and define the variables to call the function performing the bootstrap correction ***boot_correction*** that is in the source folder *src*. Here we explain the main features of the function performing the bootstrap that are own written or borrowed as explained below:

Listing 1: Function for bootstrap correction

```
function [plugin,delta,corrected,decomp_pi,decomp_b,dimensions,NT,n_problems_lev]=
    boot_correction(y,mat_id_fe,mat_controls,n_lev,n_boot,period,group,type_boot,
    type_hc,mytol,resid_controls,filename)
```

boot_correction is the main function collecting user-defined variables and calling the rest of the functions. The mandatory inputs are:

(a) *y*: the outcome of dimension N* x 1 (where N* is the number of person-year observations).

(b) *mat_id_fe*: a matrix with fixed effect identifiers. It requires at least two leading identifiers (workers and firms in the example) but can have additional fixed effects that will be ignored when creating the connected set but normalized to avoid collinearities.

The non-mandatory inputs are:

(a) *mat_controls*: matrix of additional controls that are not fixed effects. Dimensions: N* x K. The default is empty. If the user provides the matrix, the variances and covariances are estimated for all K variables together.

(b) *n_lev*: a natural number stating the simulations for the leverage estimation. The default value when choosing *type_hc* equal to 'hc_2' is 300.

(c) *n_boot*: a natural number stating the simulations for the bootstrap corrections. The default value is 300.

(d) *period*: a vector with different period identifiers to compute the second order moments per period. Dimension: N* x 1. For example N* can be partitioned in N_1^* and N_2^* observations to compute second order moments for the first N_1^* and the last N_2^* observations. The default is empty.

(e) *group*: Group identifiers when assuming that errors are serially correlated. Dimension: N* x 1. It requires that *type_boot* is 'wild_block'. The group can be the worker-firm match, the region or any other grouping. The default when *type_boot* is 'wild_block' is to generate match identifiers to assume serially correlated errors at the match level.

(f) *type_boot*: can take values of 'diag' when assuming a diagonal covariance matrix or 'wild_block' when assuming serial correlation of the error terms. The default value is 'diag'.

(g) *type_hc*: Can take values of 'hom', 'hc_0', 'hc_1' or 'hc_2' depending on the estimator of the diagonal covariance matrix. It requires that *type_boot* is equal to 'diag'. When assuming *type_boot* equal to 'wild_block' the function does not use the variable *type_hc*. The default value is 'hc_2'.

(h) *mytol*: tolerance parameter for the preconditioned conjugate gradient method *pcg*. The default is $1e^{-6}$.

(i) *resid_controls*: Indicator to residualize all other variables except the two leading fixed effects. The default is to residualize additional fixed effects and controls beyond the two leading fixed effects by setting resid_control=1.

(j) *filename*: a string with the beginning of the filenme for the exported CSV tables. If the user provides a filename of 'example', the exported tables will be: example_plugin_estimates, example_corrected_estimates, example_var_decomp_plugin, example_var_decomp_corrected. The default is empty. The user can store the tables in a different folder than the working directory by defining the relative path using the *filename*. For example *filename*='./results/example' would store in the subfolder *results* from the current working directory.

The output is:

1. *Tables to folder*: the function stores 4 csv tables with plugin estimates of the second order moments, corrected estimates and their respective variance decompositions. If *filename* is not provided, the tables are stored in the working directory.

2. *plugin*: plug-in estimates of second order moments. If the user provided the input *period* those are estimated per specified period.

3. *delta*: is the bootstrap estimate of the bias of second order moments.

4. *corrected*: are the second order estimates that are computed as *plugin* - *delta*. If the user provided the input *period* those are estimated per specified period.

5. *decomp_pi*: variance decomposition using the plugin estimates of second order moments. If the user provided a *period* vector, the decomposition is performed per period.

6. *decomp_b*: variance decomposition using the bootstrap corrected estimates of second order moments. If the user provided a *period* vector, the decomposition is performed per period.

7. *dimensions*: dimensions taking into account normalizations of fixed effects.

8. *NT*: number of person-year observations in the final sample where the second order moments are computed. NT can be below N* if the user did not provide a connected set or a leave-one-out connected set. If the user provided a *period* vector, *NT* corresponds to the total final sample size across all periods.

9. *n_problems_lev*: Number of problematic leverage estimates identified in the diagnostic and directly computed.

In the following we summarize the main steps of the *boot_correction* function. First, we compute the connected set generated by the two leading fixed effects with the function *connected*.[1]

Listing 2: Find the connected set

```
    %% Compute the connected set and relabel
%Find the connected set. Then define dimensions and relevant matrices.
sel = connected(firmid,lagfirmid,sel);

%%% Filter after finding connected set
y=y(sel); firmid=firmid(sel); id=id(sel);
all_covariates = all_covariates(sel,:);
lagfirmid=lagfirmid(sel);
```

If the chosen bootstrap type is 'diag' and the covariance matrix estimator is 'hc_2' the code computes the leave-one-out connected set in *pruning* and eliminates worker identifiers that only appear once.[2] Then we normalize the fixed effects to avoid collinearities and compute the dimensions.

[1]This function was written based on code from Card, Heining, and Kline (2013).

[2]The *pruning* function and *build_adj* are based on an older version of the function *pruning_unbal _v3* and *build_adj* from Kline, Saggio, and Sølvsten (2020).

Listing 3: Pruning

```
    if (strcmp(type_boot,'diag') && (strcmp(type_hc,'hc_2')))
        disp('----------------- Pruning ----------------- ')
        tic
        [y,id,firmid,all_covariates,period,id_old,firmid_old] = pruning(y,id,firmid,
            all_covariates,n_FE,period,id_old,firmid_old);
        toc

        disp(['Size of pruned data: ',num2str(size(y,1))])



        % Remove ids that appear only once. They dont generate connections
        % and have leverage equal to 1
        % Count obs per id
        [~,~,iunique] = unique(id);
        n = accumarray(iunique(:),1);
        % Remove observations with only one observation
        sel = (n>1); % filter out unique
        sel = sel(iunique); % Repeat to filter original vector
```

If the user only provided the two leading fixed effects or is willing to residualize additional fixed effects and controls by setting *resid_control* equal to 1, the normal equations will be estimated using a fast preconditioner for Laplacian matrices estimated in the function *cmg_sdd* borrowed from Kline et al. (2020). Their function is based on codes provided by Ioannis Koutis related to the paper Koutis, Miller, and Tolliver (2011). If there are additional fixed effects or controls and the user set resid_control equal to 0, the preconditioners for the *pcg* Matlab function are calculated using a Cholesky decomposition for general matrices.[3]

Listing 4: Residualize

```
    %% Residualize extra Fixed Effects and Controls (if resid_controls==1)
    if lap==1 && resid_controls==1
        disp('Warning: no controls nor extra fixed effects to residualize...')
        disp('Proceeding with two leading fixed effects')
    elseif lap==0 && resid_controls==1
        disp('Residualizing controls and/or extra fixed effects')
        n_2FE = size([D,F*S],2); %dimension of two leading fixed effects
        xx=X'*X;
        xy=X'*y;
        % Preconditioner for general matrices
        L =lchol_iter(xx);
```

[3]We borrowed the function iteratively computing Cholesky decompositions, *lchol_iter*, from Kline et al. (2020).

```
    if size(L,1) > 0 % if one of the -ichol()- evaluations succeeded, then use preconditioner
        % Estimation with all covariates
             b           = pcg(xx,xy,mytol,300,L,L');
    else
             b           = pcg(xx,xy,mytol,300);
    end
     % Redefine left-hand variable. Variance decompositions will be done
     %on this variable
     y = y - X(:,n_2FE+1:end)*b(n_2FE+1:end);
```

If the user chose to estimate the covariance matrix assuming heteroskedasticity, type_hc='hc_2', we estimate the leverage of each observation h_{ii} in the function *lev_diagonal_par*. We make sure that the estimated leverages lie between zero and one by adapting the normalization of Kline et al. (2020) as explained in the Online Appendix where we propose a different non-linearity bias correction. We additionally have a diagnostic that ensures that the estimated values of $1/(1-h_{ii})$ after the correction are consistent with values of leverages that are between $1/N$ and 1 as stated in Proposition OA-2.

Listing 5: Leverage

```
    disp('---------- Leverage estimation ---------- ')

    % Output is the estimated leverage (hii) and the estimated 1 -
    % leverage (mii)
    tic
    disp('Start to estimate the leverages...')
    [hii, mii, correction_lev] = lev_diagonal_par(n_lev, NT, X, xx, L, mytol,lap);
    disp('Done!')
    toc

    %% Diagnostic of leverage estimation
    fprintf('\n')
    disp('---------- Diagnostic ---------- ')
    tic
    [hc_2,n_problems_lev] = diagnostic_par(correction_lev, NT, X, xx, L, mii, mytol,lap);
    toc

            hc = hc_2; % without residual to the square
            est_var_r = r.^2.*hc; % this is hc_2 in the paper
```

In the last part of the code we estimate the bias of the second order moments by bootstrapping.

The dependent variable in the bootstrap will change depending on the chosen covariance matrix estimator. If the covariance matrix is assumed to be diagonal, the dependent variable is: $y_{b,i} = \sqrt{\widehat{\psi}_i}\, r_i$, where $\widehat{\psi}_i$ is the estimated variance for observation i and r_i is the Rademacher entry of observation i. If the covariance matrix is block diagonal, the dependent variable is slightly different as r_i is substituted by $r_{G(i)}$, that is the same Rademacher entry for observations in group $G(i)$ and the estimated variance only corrects for the degrees of freedom as explained in the main text. This means that when the covariance matrix is block-diagonal, for all observations within a block, we simulate only one Rademacher entry and use this entry $r_{G(i)}$ instead of r_i.

Listing 6: Bootstrap

```
[delta, corrected] = boot_reg_periods_par(n_boot, est_var_r,type_boot,group,
    design_mat, dimensions, period, sel_mom, n_moments, X, xx, L, plugin, mytol,
    lap);
```

The outcome of the bootstrap are estimates of the bias of second order moments δ^* which are used to compute the corrected second order moments.

3 User Guideline

The mandatory and optional inputs of the function correcting second order moments *boot_correction* are explained in Section 2.

A user assuming heteroscedastic errors and estimating the covariance matrix with HC_2 can use the minimum amount of inputs by defining an outcome variable y and a matrix of identifiers with at least two columns corresponding to the worker and firm identifiers. The function will estimate the leverage of each observation and the bootstrap correction with 200 simulations on the leave-one-out connected set. If the matrix of identifiers *mat_id_fe* has more than two columns as in the example, the default will be to residualize the additional fixed effects and compute the second order moments of the two leading fixed effects, the worker and firm effects.

Listing 7: Heteroscedasticity. Mandatory variables

```
% User input
mat_id_fe = [id firmid occ_id];

[plugin,delta,corrected,decomp_pi,decomp_b,dimensions,NT,n_problems_lev] =
boot_correction(y,mat_id_fe);
```

Below there is an example of the output when the covariance matrix is estimated using HC_2 and the additional fixed effects and controls are residualized. After residualizing, the matrix $X'X$ accepts a Laplacian representation and the leverage and bootstrap estimations take roughly 150 seconds for the example.[4]

[4]The matrix $X'X$ accepts a Laplacian representation if one set of dummies, for example the firm dummies, are changed from 1 to -1. More formally, let $X = [D\,F]$, where D and F are the person and firm dummy matrices, respectively. Their corresponding estimated

```
Starting parallel pool (parpool) using the 'local' profile ...
Connected to the parallel pool (number of workers: 6).
------------------------------------
EXAMPLE RESIDUALIZING
------------------------------------
Size initial data: 2811538
Relabeling ids...
---------- Connected set ----------
# of firms: 49646
# connected sets:2550
Largest connected set contains 47051 firms

Relabeling ids again...
Size connected set data: 2766909
---------- Pruning ----------
Elapsed time is 5.394211 seconds.
Size of pruned data: 2630701
Size after removing one-times: 2630701
Residualizing controls and/or extra fixed effects
USER WARNING: function -ichol()- with diagcomp = 0.1 failed!
USER WARNING: function -ichol()- with diagcomp = 0.16751 failed!
USER WARNING: function -ichol()- with diagcomp = 0.25126 failed!
USER WARNING: function -ichol()- with diagcomp = 0.37689 failed!
pcg converged at iteration 63 to a solution with relative residual 3.3e-07.
Done!
---------- Initial AKM ----------
Building preconditioner for Laplacian Matrix...
Done!
pcg converged at iteration 9 to a solution with relative residual 5.3e-07.
Plugin estimates:
var_worker  var_firm cov_worker_firm
2.4339       1.9927     0.1048

---------- Leverage estimation ----------
Start to estimate the leverages...
Done!
Elapsed time is 148.031077 seconds.
```

vector parameters are $\widehat{\theta}$ for the person dummies and $\widehat{\psi}$ for the firm dummies. Now, define $\tilde{F} \equiv -F$, and $\tilde{X} \equiv [D\, \tilde{F}]$. The estimated parameters of the regression using $\tilde{X}$ instead of X would be $\widehat{\theta}$ and $-\widehat{\psi}$. Thus, we would have that $F\widehat{\psi} = \tilde{F}(-\widehat{\psi})$. Then, none of the objects of interest will change.

```
---------- Diagnostic ----------
Problems in leverage estimation: 0
Elapsed time is 0.020842 seconds.

---------- Bootstrap ----------
Elapsed time is 150.960278 seconds.
Corrected estimates:
var_worker  var_firm cov_worker_firm
1.9816       1.7468     0.33469



---------- Variance decomposition ----------
Variance decomposition of plugin:
type_decomp    period   var_y      var_worker   var_firm  2*cov_worker_firm  var_resid
{'levels' }      1       5.3989      2.4339       1.9927       0.2096      0.76268
{'percent'}      1            1     0.45081       0.3691      0.038823      0.14127

Variance decomposition of corrected:
type_decomp    period   var_y      var_worker   var_firm  2*cov_worker_firm  var_resid
{'levels' }      1       5.3989      1.9816       1.7468      0.66939        1.001
{'percent'}      1            1     0.36704      0.32355      0.12399      0.18542

FINISHED!
```

The user can also define more controls by defining *mat_controls*. If the user wants to estimate corrected second order moments for the additional controls *resid_controls* needs to be equal to 0 as shown below. If the user is only interested in estimating the variance and covariance of the two leading fixed effects, it will be enough to include *mat_controls* as the default is to residualize additional fixed effects and controls.

Listing 8: Heteroscedasticity. Controls without residualizing

```
% User input and paramters
resid_controls=0;
mat_controls = [age age_sq];

[plugin,delta,corrected,decomp_pi,decomp_b,dimensions,NT,n_problems_lev] =
boot_correction(y,mat_id_fe,mat_controls,[],[],[],[],[],[],[],resid_controls,[]);
```

Without residualizing, the matrix X'X does not accept a Laplacian representation and the code is more than ten times slower in estimating the bootstrap. This is because the preconditioner used in the MATLAB *pcg* function is not as efficient as the one which was specially designed for Laplacian matrices.

To conclude the examples assuming heteroscedasticity of the error terms, one can also estimate the second order moments for all the covariates for different periods and change the number of simulations in the estimation of the leverage and the bias. The example below stores the 4 output csv in a different folder to the working directory.

Listing 9: Heteroscedasticity. Controls, different periods and other parameters

```
% User input and parameters
period=data.period;
n_lev=100;
n_boot=100;
resid_controls=0;
filename='.\results\example';

[plugin,delta,corrected,decomp_pi,decomp_b,dimensions,NT,n_problems_lev] =
boot_correction(y,mat_id_fe,mat_controls,n_lev,n_boot,period,[],[],[],[],
    resid_controls,filename)
```

Finally, if the user assumes serial correlation that leads to the covariance matrix to be block diagonal, it is necessary to specify that the type of boostrap is the Wild block bootstrap. If the user does not provide a vector of group identifiers, the function assumes serial correlation at the match level.

Listing 10: Serial correlation. Mandatory inputs

```
% User input and parameters
group=data.group;
type_boot='wild_block';

[plugin,delta,corrected,decomp_pi,decomp_b,dimensions,NT,n_problems_lev] =
boot_correction(y,mat_id_fe,[],[],[],[],group,type_boot,[],[],[]);
```

References

Card, D., J. Heining, and P. Kline (2013): "Workplace heterogeneity and the rise of West German wage inequality," *The Quarterly journal of economics*, 128, 967–1015.

Kline, P., R. Saggio, and M. Sølvsten (2020): "Leave-out estimation of variance components," *Econometrica*, 88, 1859–1898.

Koutis, I., G. L. Miller, and D. Tolliver (2011): "Combinatorial preconditioners and multilevel solvers for problems in computer vision and image processing," *Computer Vision and Image Understanding*, 115, 1638–1646.