Fast Estimation of Linear and Poisson Models with High-Dimensional Fixed Effects in Python: The `FastHDFE` package

Mario Larch[1,2*], Mirco Schönfeld[1] and Serge Shikher[3]

[1*]University of Bayreuth, Bayreuth, Germany.
[2]CEPII, CESifo, GEP, WIFO, and ifo.
[3]U.S. International Trade Commission, Washington, DC, USA.

*Corresponding author(s). E-mail(s): mario.larch@uni-bayreuth.de;
Contributing authors: mirco.schoenfeld@uni-bayreuth.de;
serge.shikher@usitc.gov;

Abstract

We present two easy-to-install Python commands, `reghdfe` and `ppmlhdfe`, for estimating linear and multiplicative (Poisson pseudo-maximum-likelihood, PPML) models with high-dimensional fixed effects. They rest on the method of alternating projections and, for PPML, iteratively reweighted least squares, and provide homoskedastic, heteroskedasticity-robust, and multi-way cluster-robust standard errors, detection and removal of separated observations via the iterative rectifier, and singleton handling, with the demeaning step implemented in compiled C code. The commands closely replicate the Stata packages of the same name, reproducing their point estimates to at least six decimal places and the standard errors to at least five decimal places. On the ITPD-E gravity dataset, with roughly 83 million observations and more than four million fixed effects, they run about six to forty-four times faster than Stata while integrating naturally into the Python ecosystem.

Keywords: high-dimensional fixed effects, Poisson pseudo-maximum-likelihood, gravity models, method of alternating projections, multi-way clustering, Python, econometric software

JEL Classification: C13 , C23 , C55 , C87 , F14

1 Motivation and Background

Datasets with a large number of observations are becoming increasingly common in many fields. Data at the household level, firm level, or country level cover many households, firms, and countries, respectively, and often over a comparably long period of time. To control for unobserved heterogeneity, many estimating models include fixed effects and often in various dimensions at the same time—this is then called a high-dimensional fixed effects specification.

To take a concrete example, the third release of ITPD-E (https://www. usitc.gov/data/gravity/itpde.htm) covers 264 countries and 170 industries, and up to 37 years of data, pooling of all data and using importer-industry-year, exporter-industry-year, and importer-exporter-industry fixed effects and assuming a balanced dataset (which it is not) will lead to up to $264 \times 170 \times 37 + 264 \times 170 \times 37 + 264 \times 264 \times 170 = 15,169,440$ fixed effects.

With common estimation approaches available in software packages such as Stata, R, or Python, it is not possible, due to time and memory constraints, to estimate models with such high-dimensional fixed effects. This is true even in the case of a linear specification estimated using Ordinary Least Squares (OLS). It is even more challenging to handle the high-dimensional fixed effects in nonlinear settings. However, common specifications motivated by theory are often nonlinear, such as the trade gravity equation, which is nowadays commonly estimated in its multiplicative form using Poisson Pseudo Maximum Likelihood (PPML), see Santos Silva and Tenreyro (2006). The same is true for the wage equation, to name a second example.

While there are now several commands in different software packages that can handle high-dimensional fixed effects, easy-to-install, easy-to-use, fast, and efficient commands to estimate linear and multiplicative models using PPML in Python are, to the best of our knowledge, still missing or only partly able to take into account the

best-practice recommendations to estimate three-way gravity models. We provide such commands.

We build on existing programs for OLS and PPML estimation with high-dimensional fixed effects and create easy-to-use and easy-to-install packages for Python. For the linear context, there are some good packages that deal with the huge number of fixed effects, such as `reghdfe` in Stata from Correia (2016). For the nonlinear context, approaches to dealing with high-dimensional fixed effects have only recently been developed. The seminal article from Santos Silva and Tenreyro (2006) advocated estimating the gravity equation in its multiplicative form. Santos Silva and Tenreyro (2006) showed that the first-order conditions are identical to the first-order conditions of the Poisson estimator. Hence, they suggest estimating the gravity model via Poisson Pseudo Maximum Likelihood (PPML) using a log-link function. PPML with high-dimensional fixed effects is even more challenging to estimate. Larch et al. (2019) suggest `ppml_panel_sg`, a fast Stata PPML panel structural gravity estimation. They use the latest recommendations from Yotov et al. (2016) and implement a PPML estimation procedure that can be used to obtain estimates for a large number of exporter-time, importer-time, and exporter-importer ("pair") fixed effects. They also describe how to obtain multi-way clustered PPML standard errors in the presence of these fixed effects. Correia et al. (2020) develop `ppmlhdfe` for Stata, which is faster than `ppml_panel_sg` and can be used with any fixed effects model. In addition to these developments in the trade literature that focus on PPML, there are several contributions dealing with high-dimensional fixed effects in generalized linear models (including the Poisson model, but also the Exponential model, and the Gamma model, to name just a few), for example Stammann (2017).

Motivated by these developments, providing Python packages that can handle high-dimensional fixed effects has the following advantages:

- Python offers a versatile and flexible programming language with a focus on readability, promoting increasing popularity in research.
- Python has many packages and can be integrated into a complete project and extended in many directions.
- Python has become one of the programming languages that researchers without a scientific computing background can also easily learn and use.
- Python allows for compiling part of the code, which is closer to hardware or which can use hardware optimization.

Against this background, the contribution of this paper is threefold. First, we provide two easy-to-install and easy-to-use Python commands, `reghdfe` and `ppmlhdfe`, that bring the functionality of the popular Stata packages of the same name to the Python ecosystem. Both commands handle an arbitrary number of high-dimensional fixed effects and offer homoskedastic, heteroskedasticity-robust, and multi-way cluster-robust standard errors, the detection and removal of separated observations through the iterative rectifier of Correia et al. (2026), and singleton handling. Second, we combine the best available algorithms—the method of alternating projections for partialling out the fixed effects and, for PPML, an iteratively reweighted least-squares scheme—with a compiled C backend for the computationally intensive demeaning step, so that the commands remain fast and memory-efficient even on datasets with tens of millions of observations and millions of fixed effects. Third, we validate our implementation against Stata's `reghdfe` and `ppmlhdfe`: on the large ITPD-E gravity dataset our commands reproduce the Stata point estimates to at least six decimal places and the standard errors to at least five decimal places, while running between about six and forty-four times faster (Tables 4 and 3). Taken together, these features make high-dimensional fixed effects estimation of both linear and multiplicative models routinely feasible within a single, open-source Python workflow.

The remainder of the paper is organized as follows. Section 2 documents the two commands and their common interface. Section 3 describes the underlying estimation methods, and Section 4 reviews existing high-dimensional fixed effects estimators. Section 5 presents two applications—one based on the small GME example dataset and one on the large ITPD-E dataset—that illustrate the commands and benchmark them against Stata. Section 6 concludes.

2 Documentation

We provide a Python package containing two sub-modules called `reghdfe` and `ppmlhdfe`, that can be used to estimate linear and multiplicative models with high-dimensional fixed effects. There is a common interface for both sub-modules. The package is distributed as a standard Python wheel and can be installed with `pip`; the source code will be made openly available in a public repository (see Section 7). The sub-modules allow choosing the structure of fixed effects as well as the type of standard errors. The packages can be installed on all available platforms, making it easy to get the estimations running.

The `reghdfe`-command contains an estimator for linear models, which is optimized for speed and will produce the common output typically expected after a linear regression. Similarly, the `ppmlhdfe`-command contains an estimator for multiplicative models with high-dimensional fixed effects. As we envision the commands to be useful for different fields, the commands allow the choice of the structure of fixed effects as well as the type of standard errors.

We had two main focuses in mind when developing the packages: i) the ease of use, including easy installation and taking care of package dependencies; ii) efficient implementation that makes the estimation fast and feasible even with very large datasets.

2.1 Common Calling Conventions

Both commands share a common interface, and the conventions described in this subsection apply to `reghdfe` and `ppmlhdfe` alike. They are stated here once and are not repeated in the description of each command.

Positional versus keyword arguments.

In Python, none of the parameters of the two commands are declared keyword-only, so every argument may in principle be supplied either *positionally* (in the exact order given in the "Syntax" line of the respective command) or as a *keyword argument* of the form `name=value`. Because both commands accept a long list of options, relying on positional order beyond the first few arguments is error-prone. We therefore recommend, and use throughout the examples below, the following convention:

- pass the dependent variable `y`, the regressors `x`, and—if desired—the fixed effects `fixedeffects` and the data frame `data` *positionally*, in this order;
- pass every remaining option as a *keyword argument* (e.g. `setype="cluster"`, `clustvars=[...]`, `FEgen=True`).

Keyword arguments may be given in any order, and any option that is not supplied takes its default value (the defaults are listed with each option in Appendix A). The dependent variable `y` and the data frame `data` are the only arguments that must always be provided.

Models without regressors.

A model *without any regressors* can be estimated by passing an empty list for `x`, i.e. `x=[]` (or simply `[]` in the second positional slot). The command then estimates a pure fixed-effects model.

Models without fixed effects.

A model *without high-dimensional fixed effects* is obtained by omitting the `fixedeffects` argument (equivalently, by passing `fixedeffects=None`). This mirrors leaving out Stata's `absorb()` option: the command then absorbs a single constant, reports an intercept (`_cons`), and its output is directly comparable to a plain regression.

Optional fixed-effect output.

The observation-wise sum of the fixed effects (`fe_rowsum`) and the dictionary of the individual estimated fixed effects (`fixed_effects`) are only computed and attached to the result object when `FEgen=True` is passed. With the default `FEgen=False`, these two attributes are *not* present on the returned object; all other attributes are always available. This is noted again next to the two attributes in the description of each command.

2.2 Requirements

Our package is based on state-of-the-art packages providing functionality for linear algebra and scientific computing. These requirements are very common for packages in the field of data science and economics. Precisely, the commands require NumPy, SciPy, and Pandas. The output of some commands further requires tabulate which is used to format console output properly. The installation of requirements is handled via pip, a common package management tool in the Python ecosystem.

2.3 Installing

The package can be installed via pip:

```
pip install fasthdfe
```

2.4 The `reghdfe` and `ppmlhdfe` Commands

The package exposes two commands with a common interface. `reghdfe` estimates linear (OLS) models with high-dimensional fixed effects, and `ppmlhdfe` estimates multiplicative models by Poisson pseudo-maximum-likelihood (PPML). Both return a result object that holds the point estimates, standard errors, variance–covariance matrix, t- and p-values, fitted values, residuals, and a range of model diagnostics, and—when requested—the estimated fixed effects. Their calling syntax is

```
reghdfe(y, x, fixedeffects=, data=, setype=, clustvars=, FEgen=, ...)
ppmlhdfe(y, x, fixedeffects=, data=, setype=, clustvars=, off=, eta_0=,
         FEgen=, start=, separation=, keepsingletons=,standardize_data=,...)
```

The first four arguments are common to both commands: a single-element list `y` with the dependent variable, a list `x` of regressors, a list `fixedeffects` of the absorbed dimensions (interactions are formed with "#", e.g. `"exp#year"`), and the `data` frame. Following the conventions of Section 2.1, these are typically passed positionally and all remaining options by keyword; a model without regressors is obtained with `x=[]` and a model without fixed effects by omitting `fixedeffects`. The "="-sign behind the remaining command line options indicates that these are keyword arguments instead of positional arguments.

The most frequently used options are the standard-error type `setype` ("none", "homoskedastic", "heteroskedastic", or "cluster"), the associated cluster variables `clustvars` (up to four dimensions), and `FEgen`, which additionally returns the estimated fixed effects. For `ppmlhdfe`, further options control the treatment of separation (`separation`, defaulting to the iterative rectifier), the handling of singletons (`keepsingletons`), the standardization of regressors (`standardize_data`), the choice of starting values (`start`, `eta_0`), and an optional offset (`off`). Convergence tolerances

and iteration limits follow Stata's defaults. A complete description of every argument, its admissible values and default, together with the full list of returned attributes, is given in Appendix A.

3 Estimation Methods

We build on the previous work of others in different software packages, as well as on our own work implementing high-dimensional fixed effects estimators in Python for linear models and nonlinear models. For linear models, note that with one type of fixed effects, we can use a within transformation to avoid the computation of the fixed effects (see for example Baltagi 2013). Guimarães and Portugal (2010) and Gaure (2013) use the method of alternating projections, tracing back to von Neumann (1951) and Halperin (1962) with the Frisch-Waugh-Lovell theorem to ease computation, which is a generalization of the within transformation to multiple sets of fixed effects. Application of this method also avoids estimation of the fixed effects and yields simple expressions for the point estimates as well as standard errors.

For nonlinear models, such as the Poisson Pseudo Maximum Likelihood, the nonlinear relationship between the linear predictor and conditional mean has to be tackled. Whereas in linear models the parameters of interest can be estimated separately from the fixed effects, the same does not hold for PPML at first glance. The reason is that fixed effects contribute to the linear predictor. Hence, they have to be updated in each iteration of the optimization routine. Guimarães and Portugal (2010) use a Gauss-Seidel algorithm combined with a numerical optimization routine to solve for the fixed effects. Larch et al. (2019) estimate a gravity model with high-dimensional fixed effects based on a modified Gauss-Seidel algorithm. We use the approach of Stammann (2017) and Correia et al. (2020), utilizing the fact that parameter updates can be computed by weighted least squares if the score function is expanded by a first-order Taylor approximation. This is known as the Newton-Raphson algorithm, and the algorithm

coincides with Fisher Scoring/Iteratively Reweighted Least Squares (IRLS) algorithm for PPML with log-link. This can be seen as a direct extension of Gaure (2013) to generalized linear models and allows us to apply the methods developed in the first step also for the PPML estimation. As a bonus, the standard errors are also computed as a by-product of the final iteration.

Additionally, we allow for capturing different assumptions about error correlation (or "clustering") patterns that are reasonable for the data and model at hand. In particular, Egger and Tarlea (2015) argue that standard errors for a panel-data gravity model should allow for simultaneous correlations across all three main dimensions of the panel—exporter, importer, and time. This can be done by "multi-way" clustering (see Cameron et al. 2011). Note that clustering simultaneously on exporter, importer, and year allows for correlation in the error term within all six possible cluster dimensions {exporter, importer, year, exporter-year, importer-year, exporter-importer}. It therefore nests the typical practice of assuming errors are solely clustered across time within each country-pair. Multi-way clustering on exporter, importer, and time is therefore expected to lead to more conservative inferences (Egger and Tarlea 2015). Our Python programs allow the user to choose the type of standard errors, including homoskedastic standard errors, Huber-White heteroskedasticity-robust standard errors, and a flexible way to cluster standard errors, all using components from the final iteration.

The critical component of both programs is the method of the alternating projections algorithm.

There is one additional challenge with PPML using high-dimensional fixed effects. Santos Silva and Tenreyro (2010) and Correia et al. (2020) show that under certain data constellations, estimates from Poisson regressions may not exist. For example, if two or more regressors are perfectly collinear over the subsample where the dependent variable is non-zero, one has to check whether estimates exist to avoid spurious or no estimates. Santos Silva and Tenreyro (2010) provide a simple example of a model

with non-collinear regressors that does not have a solution. The situation is more difficult to detect, as the perfect collinearity is over the subset of non-zero values for the dependent variable only. With multiple high-dimensional fixed effects, collinearity checks have to be performed across all the different fixed effects. One has to check whether each individual regressor is collinear over the positive values with the complete set of fixed effects, as well as whether any subset of fixed effects and non-fixed effect regressors is collinear over the positive values of trade. These checks can be performed by (i) applying the within-transformation technique and (ii) recognizing that fixed effects themselves only present an issue under certain circumstances, namely when the within-transformed version of each non-fixed effect regressor is uniformly zero.

Beyond purely fixed-effect-driven separation, Correia et al. (2026) show that separation can also arise from interactions between the regressors and the fixed effects, and they propose the *iterative rectifier* (IR, also called ReLU) as the only procedure that systematically detects separation of both kinds. We implement the IR algorithm in `ppmlhdfe` and expose it via the `separation` option, whose valid arguments mirror those of Stata's `ppmlhdfe`: `"fe"`, `"ir"` (the default), `"none"`, or a user-supplied list of methods. The IR algorithm iteratively solves a weighted least-squares problem to find a non-negative certificate $\mathbf{u}$ in the column space of $[\mathbf{X}, \mathbf{D}]$ satisfying $u_i = 0$ for $y_i > 0$ and $u_i \geq 0$ for $y_i = 0$. Observations with $u_i >$ `sep_tol` are flagged as separated and dropped from the estimation, and the singleton / fe-separation loop is then re-run on the reduced sample, since removing separated observations can create new singletons. Using the IR check by default brings our `ppmlhdfe` into line with the default behavior of Stata's `ppmlhdfe`.

Further, we account for singletons. Singletons are observations that can be fully explained by the included fixed effects (see for example Correia 2015, for a discussion). These observations do not contribute to the structural parameters, as for any parameter value, the fixed effects can be chosen such that the residual is zero. Additionally,

singletons lead to an underestimation of the standard errors. Using an iterative scheme over all sets of fixed effects that checks for variation in the dependent variable, we exclude singletons from estimation by default. The behavior is controlled by the `keepsingletons` option: if set to `True`, only groups where y is identically zero are dropped (the minimum required for Poisson identification), and singletons as well as groups with constant positive y are kept. Keeping singletons can be useful for replicating analyses that report the full sample but, as noted above, should be used with care when clustered standard errors are requested.

Combining all these features in the Python packages and using the power of Python, we build on and improve the statistical techniques and programs available. An efficient multi-core implementation, together with a combination of the best available algorithms, led to packages that can handle large datasets with many fixed effects.

The estimators build directly on methods developed in the literature. To keep the main text concise, the detailed derivations—for the linear within-estimator, the IRLS scheme for PPML, and the multi-way cluster-robust standard errors—are collected in Appendix B.

4 Existing High-Dimensional Fixed Effects Estimators

To the best of our knowledge, we now discuss several other packages that allow estimation of linear models and Poisson regressions with multiple levels of fixed effects.

For **Stata**, the package `reghdfe` from Correia (2016) (https://scorreia.com/software/reghdfe/) is able to estimate linear models with multiple levels of fixed effects. It builds upon the generalized within-estimator of Guimarães and Portugal (2010) and Gaure (2013) but improves in terms of convergence properties.

The package `ppmlhdfe` from Correia et al. (2020) (https://scorreia.com/software/ppmlhdfe/) also does within-transformation and uses a

Newton-Raphson estimation. However, it solves the model without completely within-transforming the data from scratch in each iteration. This enables Correia et al. (2020) to realize speed gains.

The packages from Stata, specifically the `ppmlhdfe`-package, will serve as our benchmark.

For **R**, the package `lfe` from Gaure (2013) (https://cran.r-project.org/web/ packages/lfe/ index.html) uses the method of alternating projections to estimate linear models with multiple group fixed effects.

The package `alpaca` from Stammann (2017) (https://github.com/amrei-stammann/alpaca) combines a within-transformation step with a Newton-Raphson estimation algorithm, which is equivalent to the IRLS method for the PPML estimator with a log-link function.

The package `fixest` from Bergé (2018) (https://github.com/lrberge/fixest) provides a family of functions to perform estimations with multiple fixed-effects. It involves a within-transformation step with a Newton-Raphson estimation algorithm, but uses a nonlinear Gauss-Seidel method to update the fixed effects as described in Figueiredo et al. (2015).

There are some recent developments for handling high-dimensional fixed effects in **Python**. Specifically, the Python `pyhdfe`-package provided by Jeff Gortmaker and Anya Tarascina is available at https://pyhdfe.readthedocs.io /en/stable/index.html#. This package provides algorithms for absorbing high-dimensional fixed effects. It also allows dropping singletons, calculating degrees of freedom with clustered standard errors, and has options to employ several accelerations for fixed-point iterations. It also provides various types of algorithms to absorb the fixed effects:

1. within transformation (which only works for a single fixed effects dimension);
2. method of alternating projections. This is the method implemented in `reghdfe` and described in Guimarães and Portugal (2010); Correia (2016), for example;

3. LSMR method of Fong and Saunders (2011), modified for simultaneous iteration over multiple matrix columns;
4. the method from Somaini and Wolak (2016), which is non-iterative but only works for two dimensions;
5. regressions on dummy variables. This is only for comparison and will not work with high-dimensional fixed effects.

However, this command only works for linear models. As a programmer's command, it is not useful for actually performing estimations.

Furthermore, there is the `pyfixest`-package (https://github.com/py-econometrics/pyfixest), which can be viewed as a Python implementation of the R-`fixest`-package. While this package is a great public good for estimating gravity equations, there are at least two shortcomings that we see: i) the package seems not to handle perfectly collinear regressors well, ii) the package does not support more than two-way clustering.

For linear models, there is also the Python-package `FixedEffectModel` (https://pypi.org/project/FixedEffectModel/ and the `regpyhdfe`-package (https:// regpyhdfe.readthedocs.io/en/ latest/index.html), the latter seeming work in progress.

The package `fastreg` from Douglas Hanley (https://github.com/iamlemec/fastreg?tab=readme-ov-file#documentation) creates well-structured data matrices from actual data using Pandas DataFrames, achieving gains in computation time this way. We considered `fastreg` as a possible foundation for our Python implementation but decided against it for four reasons. First, `fastreg`'s dedicated high-dimensional fixed-effect interface (the `hdfe` argument) accepts only a single (possibly interacted) categorical term, whereas the typical gravity specification we target requires three separate sets of fixed effects—exporter-time, importer-time, and exporter-importer. Remaining fixed effects can be declared as `absorb`, but doing so sacrifices the ability to recover their standard errors and, for gravity-sized datasets, has to assemble very

large sparse dummy matrices. Second, `fastreg` does not include a procedure for detecting separation in Poisson models, which is a central concern for gravity applications with many zero trade flows and is addressed explicitly by Correia et al. (2026). Third, `fastreg`'s generalized linear model routines rely on generic gradient-based optimization on the log-likelihood (via JAX), whereas the IRLS-plus-alternating-projections approach we adopt is both faster and more numerically robust for high-dimensional fixed-effect Poisson problems. Fourth, `fastreg`'s support for multi-way cluster-robust inference is more limited than what is required for gravity applications following Egger and Tarlea (2015). Our implementation instead builds directly on the Stammann and Correia-Guimarães-Zylkin line of methods and includes a compiled C-backend for the alternating-projections step, which addresses the computational motivation for `fastreg` without sacrificing the features above.

These are all valuable contributions, and we picked and built on the best available implementations for the `reghdfe` and `ppmlhdfe` commands of our `fasthdfe` package.

5 Application of Commands

In this section, we provide a demonstration of the programs using two examples: i) the dataset from the GME package, and ii) the ITPD-E. While the first dataset is smaller and therefore lends itself to experimentation with the command and its different options, the ITPD-E is a large, sectoral dataset that will be useful to demonstrate that the programs can also deal with a huge amount of fixed effects.

5.1 Dataset from the GME Package

The dataset from the GME package is available at https://www.usitc.gov/data/ gravity_example_trade_and_grav_data_small.csv. This dataset contains data for 62 exporters, 62 importers and 27 years.

We demonstrate the codes by estimating a log-linearized gravity model using `reghdfe` and a gravity model in its multiplicative form using `ppmlhdfe`. We estimate the model containing different types of fixed effects and using different types of standard errors to show how to use the options and the feasibility to estimate this model also with importer-year, exporter-year, and importer-exporter fixed effects.

The dataset from the GME package is a panel dataset containing values for trade flows (*trade_value*), and a set of control variables: the log of the geographical distance between two countries (*log_distance*), a dummy variable for whether two countries have a regional trade agreement (*agree_pta*), a dummy variable for whether two countries have a common language (*common_language*), and a dummy variable indicating whether two countries share a common border (*contiguity*).

5.1.1 Linear Model

For the linear model, we specify the following estimating equation:

$$\begin{aligned}\ln(trade\_value)_{ij,t} = {} & \beta_1 log\_distance_{ij} + \beta_2 agree\_pta_{ij,t} + \beta_3 common\_language_{ij} \\ & + \beta_4 contiguity_{ij,t} + \text{fixed effects} + \epsilon_{ij,t}, \end{aligned} \tag{1}$$

where $\epsilon_{ij,t}$ denotes a remainder error for exporter i, importer j, and year t. To control for multilateral resistances (see Anderson and van Wincoop 2003), we include exporter-time and importer-time fixed effects. To control for unobserved time-constant bilateral effects, we include bilateral fixed effects. Note that in this case, all variables without time-variation will be perfectly controlled for by the fixed effects.

We compare the results of our Python programs with the results obtained with Stata using the appropriate similar commands.

We estimate this specification with homoskedastic, heteroskedasticity-robust, and multi-way cluster-robust standard errors. As an example, the cluster-robust variant is obtained with

```
from fasthdfe import reghdfe
reghdfe(['lntrade_value'],
        ['log_distance','agree_pta', 'common_language','contiguity'],
        fixedeffects = ['importer#year','exporter#year','exporter#importer'],
        data = df,
        setype='cluster',
        clustvars=['importer','exporter','year'])
```

Because `log_distance` and `common_language` are time-invariant, they are perfectly absorbed by the bilateral fixed effects and are dropped. Table 1 reports the estimates. The point estimates are identical across the three columns; only the standard errors change with the assumed error structure. The results coincide with those of Stata's `reghdfe` to at least six decimal places; the full command-line output of both implementations is reproduced in Appendix C.

5.1.2 Multiplicative Model Estimated Using PPML

For the multiplicative model, we specify the following estimating equation:

$$\begin{aligned} trade\_value_{ij,t} = \exp\big(&\beta_1 log\_distance_{ij} + \beta_2 agree\_pta_{ij,t} + \beta_3 common\_language_{ij} \\ &+\beta_4 contiguity_{ij,t} + \text{fixed effects}\big) + \epsilon_{ij,t}, \end{aligned} \tag{2}$$

where $\epsilon_{ij,t}$ denotes a remainder error for exporter i, importer j, and year t. In order to control for multilateral resistances (see Anderson and van Wincoop 2003), we include

Table 1 GME data: linear (OLS) gravity estimates.

	(1) Homoskedastic	(2) Robust	(3) Cluster
`agree_pta`	0.0411	0.0411	0.0411
	(0.0213)	(0.0167)	(0.0458)
`contiguity`	0.9579	0.9579	0.9579
	(0.5328)	(0.3297)	(0.3615)
Observations	80,350	80,350	80,350
R^2	0.932	0.932	0.932

Notes: Dependent variable ln(*trade_value*). All columns absorb importer–year, exporter–year, and exporter–importer fixed effects; `log_distance` and `common_language` are time-invariant and absorbed. Standard errors in parentheses; column (3) clusters on importer, exporter, and year (27 clusters). Estimates match Stata's `reghdfe`; the full output is in Appendix C.

exporter-time and importer-time fixed effects. To control for unobserved time-constant bilateral effects, we include bilateral fixed effects. Note that in this case, all variables without time-variation will be perfectly controlled for by the fixed effects.

We again compare the results of our Python programs with the results obtained with Stata using the appropriate similar commands.

We estimate the multiplicative specification with heteroskedasticity-robust and with multi-way cluster-robust standard errors. The cluster-robust variant is obtained with

```
from fasthdfe import ppmlhdfe
ppmlhdfe(['trade_value'],
         ['log_distance','agree_pta','common_language','contiguity'],
         fixedeffects = ['importer#year','exporter#year','exporter#importer'],
         data = df,
         setype='cluster',
         clustvars=['importer','exporter','year'])
```

Table 2 reports the estimates. As in the linear case, `log_distance` and `common_language` are absorbed, the point estimates are common across columns, and the results reproduce those of Stata's `ppmlhdfe`; the full output is given in Appendix C.

Table 2 GME data: PPML gravity estimates.

	(1) Robust	(2) Cluster
`agree_pta`	−0.0321	−0.0321
	(0.0178)	(0.0468)
`contiguity`	0.6820	0.6820
	(0.1469)	(0.0477)
Observations	82,729	82,729
Pseudo-R^2	0.991	0.991

Notes: Dependent variable *trade_value*, estimated by PPML. All columns absorb importer–year, exporter–year, and exporter–importer fixed effects; `log_distance` and `common_language` are absorbed. Standard errors in parentheses; column (2) clusters on importer, exporter, and year (27 clusters). Estimates match Stata's `ppmlhdfe`; the full output is in Appendix C.

5.2 ITPD-E

Our second application uses the ITPD-E Release 3, which is available for download at https://www.usitc.gov/data/gravity/itpde.htm. ITPD-E Release 3 contains data for 264 countries and 170 industries. It covers the years 1986-2022 for the agricultural industries 1-26, the years 1988-2022 for forestry (industry 27), fishing (industry 28) and the mining and energy (industries 29-35) and manufacturing and industries (industries 36-153), and the years 2000-2022 for the services industries (industries 154-170). ITPD-E Release 3 is a large database that has 83,503,326 observations.

We complement the trade and domestic sales data from ITPD-E Release 3 with USITC's Dynamic Gravity Dataset Release 2.1 (https://www.usitc.gov/

data/gravity/dgd.htm), from which we use the variable "contiguity" (other non-time varying effects are already controlled by the fixed effects) and Mario Larch's Regional Trade Agreements Database from Egger and Larch (2008), from which we use the variable "rta", which is a dummy variable taking value 1 when trade between two countries takes place under a regional trade agreement, and is zero else. After merging these control variables and cleaning the dataset a bit (dropping BLX, EUN, FRE, SVU, and ETF, filling in missing values for "contiguity" for SRB for 2005 taking values from 2006, and filling the last three years missing in the dataset for "contiguity" using the last available year, and keeping only observations where we have information of the control variables), we end up with 82,904,447 observations, containing 31,847,454 non-zero trade flow observations.

To demonstrate that our estimators can handle such large datasets, we pool over industries, leading to computationally very challenging specifications including exporter–industry–year, importer–industry–year, and exporter–importer–industry fixed effects. We estimate both a linear (log-linearized) and a multiplicative (PPML) gravity model on this dataset, and compare both the estimates and the computation times of our Python commands with those of the corresponding Stata commands.

5.2.1 Linear Model

For the linear model, we specify the following estimating equation:

$$\ln(trade\_value)_{ijs,t} = \beta_1 rta_{ij,t} + \beta_2 contiguity_{ij,t} + \text{fixed effects} + \epsilon_{ijs,t}, \qquad (3)$$

where $\epsilon_{ijs,t}$ denotes a remainder error for exporter i, importer j, industry s, and year t. In order to control for multilateral resistances (see Anderson and van Wincoop 2003), we include exporter-industry-time and importer-industry-time fixed effects. In order to control for unobserved time-constant bilateral effects, we include bilateral-industry

fixed effects. Note that in this case, all variables without time-variation will be perfectly controlled for by the fixed effects.

Standard errors are clustered on exporter, importer, industry, and year, as suggested for gravity equation estimation by Egger and Tarlea (2015). The resulting Python and Stata estimates are reported in Table 3 (Panel A): the two implementations produce virtually identical point estimates and standard errors, agreeing to at least six decimal places. The full command-line output of both implementations is reproduced in Appendix C.

5.2.2 Multiplicative Model Estimated Using PPML

For the multiplicative model, we specify the following estimating equation:

$$trade\_value_{ijs,t} = \exp\left(\beta_1 rta_{ij,t} + \beta_2 contiguity_{ij,t} + \text{fixed effects}\right) + \epsilon_{ijs,t}, \quad (4)$$

where $\epsilon_{ijs,t}$ denotes a remainder error for exporter i, importer j, industry s, and year t. In order to control for multilateral resistances (see Anderson and van Wincoop 2003), we include exporter-industry-time and importer-industry-time fixed effects. In order to control for unobserved time-constant bilateral effects, we include bilateral-industry fixed effects. Note that in this case, all variables without time-variation will be perfectly controlled for by the fixed effects.

As for the linear model, standard errors are clustered on exporter, importer, industry, and year, and we additionally use the "`separation(fe)`" option. The Python and Stata estimates are reported in Table 3 (Panel B): the point estimates agree to at least six decimal places and the standard errors to at least five decimal places. The full command-line output is reproduced in Appendix C.

Table 3 Agreement of point estimates and standard errors (Python vs. Stata), ITPD-E.

	Coefficient		Std. error	
Variable	Python	Stata	Python	Stata
Panel A: Linear model (OLS)				
`rta`	0.0125774	0.0125774	0.0278550	0.0278550
`contiguity`	0.3449093	0.3449093	0.1519794	0.1519794
Panel B: PPML				
`rta`	0.1005611	0.1005609	0.0362445	0.0362446
`contiguity`	−0.3233573	−0.3233568	0.0698125	0.0698111

Notes: Both specifications absorb exporter–industry–year, importer–industry–year, and exporter–importer–industry fixed effects, with standard errors clustered on exporter, importer, industry, and year. The two implementations use the identical estimation sample. Point estimates agree to at least six decimal places; standard errors agree to at least six decimal places for the linear model and at least five for PPML, the small PPML discrepancy reflecting minor differences in the finite-sample variance calculation rather than in the sample or the point estimates.

5.2.3 Computation Times

The main motivation for our Python implementation is speed on large datasets. We therefore compare the wall-clock runtime of our Python commands with the corresponding Stata commands on two machines: a workstation with an AMD Ryzen Threadripper PRO 9985WX 64-core processor and 512 GB of RAM, and the USITC modeling server model-05 (16 Xeon Gold 6342 cores, 384 GB of RAM, Windows Server 2022).

For the linear model, Stata took 510.22 seconds (8 min 30 s) on the workstation and 849.33 seconds (14 min 9 s) on the USITC server, whereas our Python command took only 68.26 seconds (1 min 8 s) and 135.95 seconds (2 min 16 s), respectively. The gains are even larger for the computationally more demanding PPML model: Stata took 63,285.77 seconds (about 17.6 hours) on the workstation and 89,861.15 seconds (about 25 hours) on the USITC server, against only 1,442.87 seconds (about 24 min) and 7,334.66 seconds (2 h 2 min) for Python.

Table 4 collects these times, and Figure 1 displays the resulting speed-ups. Across both specifications and both machines, our Python implementation is between about six and forty-four times faster than Stata, with the largest gains for the computationally most demanding PPML specification. The USITC runs were carried out with an earlier version of the code, so the workstation figures—in particular the much larger PPML speed-up—better reflect the current performance of the implementation.

Table 4 Runtime comparison on the ITPD-E dataset (Stata vs. Python).

Model	Hardware	Stata (s)	Python (s)	Speed-up
Linear (OLS)	Threadripper PRO 9985WX (64c, 512 GB)	510.22	68.26	7.5×
Linear (OLS)	USITC model-05 (16c Xeon 6342, 384 GB)	849.33	135.95	6.2×
PPML	Threadripper PRO 9985WX (64c, 512 GB)	63,285.77	1,442.87	43.9×
PPML	USITC model-05 (16c Xeon 6342, 384 GB)	89,861.15	7,334.66	12.3×

Notes: Both specifications absorb exporter–industry–year, importer–industry–year, and exporter–importer–industry fixed effects, and use standard errors clustered on exporter, importer, industry, and year. The PPML specification additionally uses the `separation(fe)` option. Times are wall-clock seconds; the speed-up is the Stata time divided by the Python time.

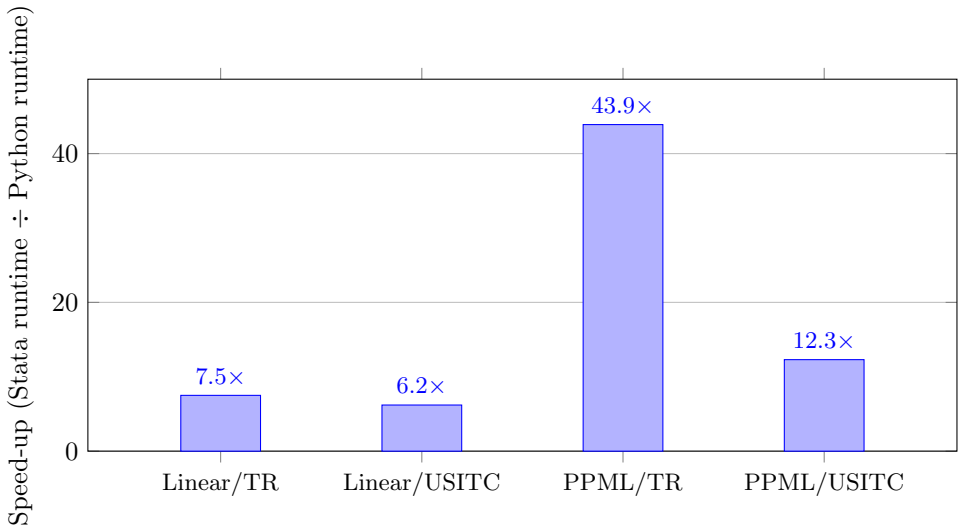

Fig. 1 Speed-up of our Python implementation over Stata on the ITPD-E dataset, i.e. the Stata runtime divided by the Python runtime. "TR" denotes the AMD Threadripper PRO 9985WX workstation and "USITC" the USITC modeling server; "Linear" and "PPML" the linear and Poisson specifications. The underlying wall-clock times are reported in Table 4.

6 Conclusion

We have presented `reghdfe` and `ppmlhdfe`, two Python commands for estimating linear and multiplicative models with high-dimensional fixed effects. The commands mirror the interface and output of the widely used Stata packages of the same name and combine the method of alternating projections for partialling out the fixed effects, an iteratively reweighted least-squares scheme for PPML, multi-way cluster-robust inference, detection of separated observations through the iterative rectifier, and singleton handling, with the computationally intensive demeaning step implemented in compiled C code.

Two applications illustrate the commands. On the small GME example dataset they reproduce a range of specifications and options; on the large ITPD-E gravity dataset, with roughly 83 million observations and more than four million absorbed fixed effects, they reproduce the Stata point estimates to at least six decimal places and the standard errors to at least five decimal places while running between about six and forty-four times faster (Tables 4 and 3). By integrating naturally into the Python scientific-computing ecosystem and being installable through `pip`, the commands lower the barrier to best-practice estimation of gravity-type models and, more generally, of models with high-dimensional fixed effects. Future work will extend the framework to additional generalized-linear-model families, add further variance estimators, and continue to improve computational performance.

7 Code and Data Availability

The `reghdfe` and `ppmlhdfe` commands are distributed as part of an open-source Python package `fasthdfe`. The package is available on the Python Package Index (PyPI), so that it can be installed with a single `pip install fasthdfe`.

The datasets used in the applications are publicly available. The small example dataset is distributed with the GME package (USITC gravity data portal); the ITPD-E

Release 3 database (https://www.usitc.gov/data/gravity/itpde.htm) and the Dynamic Gravity Dataset Release 2.1 (https://www.usitc.gov/data/gravity/dgd.htm) are provided by the USITC; and Mario Larch's Regional Trade Agreements Database is described in Egger and Larch (2008).

Declarations

Funding. This work was supported by the United States International Trade Commission (USITC) under contract 34300025P0032.

Competing interests. The authors declare that they have no competing interests.

Code and data availability. See Section 7.

Disclaimer. The views expressed are strictly those of the authors and do not represent the opinions of the USITC or any of its Commissioners.

Appendix A Detailed Command Reference

This appendix documents every argument of `reghdfe` and `ppmlhdfe`, their admissible values and defaults, and the full set of attributes returned in the result object.

A.1 `reghdfe`

`reghdfe` is a command available via our package to estimate linear models with high-dimensional fixed effects that has the following structure:

1. Inputs:
   (a) dependent variable,
   (b) a set of regressors,
   (c) a flexible way to specify the type of fixed effects one wants to include,
   (d) options to choose convergence criteria,
   (e) options to choose from various types of standard errors (homoskedastic standard errors, heteroskedasticity-robust standard errors, clustered standard errors),
   (f) option to enable/disable conjugate gradient acceleration,
   (g) option to save fixed effects.
2. Outputs:
   (a) coefficient estimates,
   (b) standard errors,
   (c) variance-covariance matrix,
   (d) t-values,
   (e) number of observations,
   (f) model diagnostics (e.g., Akaike Information, Bayesian Information),
   (g) fitted values, which can be merged with the input data,
   (h) residuals, which can be merged with the input data,
   (i) optional: fixed effects, which can be merged with the input data.

3. Syntax:

```
reghdfe(y,
        x,
        fixedeffects              = None,
        data                      = None,
        setype                    = "none",
        clustvars                 = None,
        FEgen                     = False,
        verb                      = 0,
        eps_MAP                   = 1e-9,
        minsubiter                = 0,
        maxsubiter                = 1000,
        accel                     = True,
        nt                        = 0,
        print_stata_style_summary = True)
```

The general rules on which arguments can be passed positionally and which should be passed by keyword, as well as how to estimate models without regressors (`x=[]`) or without fixed effects (omitting `fixedeffects`), are described in Section 2.1 and apply here. In short, `y`, `x`, `fixedeffects`, and `data` are typically passed positionally in this order, and all remaining options by keyword.

(a) Parameters:

- `y`: single entry list containing the name of the dependent variable as a string. It is the name of the column in `data`. Must always be provided.
- `x`: string list of the names of the explanatory variables, which are columns in `data`. Pass an empty list, `[]`, to estimate a model without any

regressors (a pure fixed-effects or constant-only model). No default; this argument must be given (possibly as `[]`).

- `fixedeffects`: string list of the names of the fixed effects. Variables can be combined by "#", e.g. "state#year" creates state-year fixed effects. Default is `None`, which mirrors omitting Stata's `absorb()`: the model is then estimated with a single absorbed constant (an intercept `_cons`) and no high-dimensional fixed effects.
- `data`: pandas data frame containing all relevant variables. Must always be provided.
- `setype`: string, type of standard errors. Allowed values are "none" (no standard errors are computed), "homoskedastic" ("homo" / "iid" are synonyms), "heteroskedastic" ("r", "robust", "hetero" and "HC1" are synonyms), and "cluster" ("multiway", "clu", "cl", "c" are synonyms). Default is `''none''`.
- `clustvars`: string list of cluster variables. Maximum length 4, ignored if `setype` is not "cluster" (and required if it is). Default is `None`.
- `FEgen`: bool, if `True`, the observation-wise sum of the fixed effects (`fe_rowsum`) and the dictionary of individual fixed-effect estimates (`fixed_effects`) are computed and attached to the result object. With the default `False` these two attributes are not present on the returned object (see Section 2.1). Default is `False`.

(b) Additional Parameters:

- `accel`: bool, if `True`, MAP employs conjugate gradient acceleration technique following Hernández-Ramos et al. (2011). Default is `True`.
- `nt`: integer, sets the number of threads to use. Default is `0`, which uses "system settings". Has no effect if the backend is Python.
- `verb`: int $\in \{0, 1, 2\}$, level of "in-between" output displayed (0 is none). Default is `0`.

- `eps_MAP`: float $\in (0, \infty)$, convergence criterion of the alternating projections algorithm given. Default is 10^{-9}.
- `minsubiter`: minimum number of iterations within the demeaning process. Default is 0.
- `maxsubiter`: maximum number of iterations within the demeaning process. Default is 1000.
- `print_stata_style_summary`: bool, regression output table is printed. Default is `True`.

(c) Returns a class object with the following attributes:

- `params`: point estimates.
- `bse`: standard errors if calculated, "NULL" otherwise.
- `covmat`: variance-covariance matrix if calculated, "NULL" otherwise.
- `covmat_nofix`: variance-covariance matrix prior to the "eigenvalue fix". Only present if `eigenvalue_fix` is `True`.
- `eigenvalue_fix`: `True` only if the covariance matrix was not positive semi-definite and the eigenvalue fix had to be applied (negative eigenvalues replaced by zero); otherwise the attribute is not set.
- `cov_type`: string, type of standard errors.
- `clustvars`: list, string list with cluster dimensions if `cov_type` is "cluster", "NULL" otherwise.
- `tvalues`: t-values for $H_0 = 0$.
- `pvalues`: p-values for a two-sided t-test with $H_0 = 0$.
- `nobs`: int, number of observations used for estimation. Some observations may have been excluded and do not count in here.
- `df_model`: number of parameters in X plus the theoretical upper bound of fixed effects (collinear fixed effects count as estimated parameters

here; this may lead to a slight over-estimation of small-sample corrected standard errors).

- `df_resid`: `nobs` − `df_model`.
- `aic`: Akaike Information Criterion.
- `bic`: Bayesian Information Criterion.
- `llf`: value of the log-likelihood.
- `deviance`: sum of the squared deviance residuals, equal to the sum of squared residuals.
- `pearson_chi2`: Pearson's χ^2-value, equal to the sum of squared residuals.
- `family_name`: "Gaussian distribution".
- `family_link`: "Identity function", $(g(\cdot))$.
- `scale`: ML estimate of σ^2. To obtain the OLS estimates, multiply by nobs/(nobs-1).
- `method`: "OLS".
- `fit_history`: dict with key 'iteration', equal to 1, as analytical OLS solution is computed.
- `fittedvalues`: array of the fitted values.
- `fe_rowsum`: array of the observation-wise sum of the fixed effects $\boldsymbol{\eta} - \mathbf{x}'\boldsymbol{\beta}$. Only present if `FEgen=True` (see Section 2.1); otherwise the attribute is not set.
- `fixed_effects`: dictionary of all estimates of all fixed effects specified (the explicit constant `_cons`, the per-observation centered fixed effects, and the group-level values). Only present if `FEgen=True`; otherwise the attribute is not set.
- `resid`: array of residuals.
- `msg`: list, information/warning messages during the estimation process.
- `formula_R`: string, formula as you would type it in R.

- `formula_Stata`: string, formula as you would type it in Stata.
- `model`: GLM class, for compatibility with the gme-package.

(d) Example:

```
from fasthdfe import reghdfe
result = reghdfe(['lntrade_value'],
     ['LN_DIST','agree_pta','contiguity','common_language'],
     fixedeffects = ['exp_ind#industry_id#year',
                     'imp_ind#industry_id#year',
                     'exp_ind#imp_ind#industry_id'],
     data = df,
     setype='cluster',
     clustvars=['exp_ind','imp_ind','year','industry_id'])
```

A.2 ppmlhdfe

`ppmlhdfe` is a user-friendly Python command to estimate multiplicative models with high-dimensional fixed effects with PPML that has the following structure:

1. Inputs:
   (a) dependent variable,
   (b) a set of regressors,
   (c) a flexible way to specify the type of fixed effects one wants to include,
   (d) options to choose convergence criteria,
   (e) options to choose from various types of standard errors (homoskedastic standard errors, heteroskedasticity-robust standard errors, clustered standard errors),
   (f) options for initial values of the linear predictor (simple, OLS-based, or user-supplied starting values for selected coefficients),

(g) options to detect and drop separated observations (fixed-effect-style separation, iterative rectifier, or none, mirroring Stata's `separation()` option),

(h) option to keep or drop singleton observations,

(i) an offset option for constrained coefficients,

(j) option to enable/disable conjugate gradient acceleration,

(k) option to standardize regressors before estimation for improved numerical stability (mirroring Stata's `standardize_data(1)` default),

(l) option to save fixed effects.

2. Outputs:

(a) coefficient estimates,

(b) standard errors,

(c) variance-covariance matrix,

(d) t-values,

(e) number of observations,

(f) diagnostics on the number of observations kept and dropped (total, invalid values, singletons/fe-separation, and separation detected by the iterative rectifier),

(g) model diagnostics (e.g., Akaike Information Criterion, Bayesian Information Criterion, deviance and pseudo-likelihood),

(h) fitted values, which can be merged with the input data,

(i) residuals, which can be merged with the input data,

(j) optional: fixed effects, which can be merged with the input data.

3. Syntax:

```
ppmlhdfe(y,
         x,
         fixedeffects             = None,
         data                     = None,
         setype                   = "none",
         clustvars                = None,
         off                      = None,
         eta_0                    = None,
         FEgen                    = False,
         start                    = "simple",
         olsguess                 = False,
         separation               = "ir",
         sep_tol                  = 1e-5,
         sep_maxiter              = 100,
         sep_K_penalty            = 1e6,
         keepsingletons           = False,
         nocheck                  = False,
         verb                     = 0,
         eps_beta                 = 1e-8,
         miniter                  = 1,
         maxiter                  = 1000,
         eps_MAP                  = 1e-9,
         minsubiter               = 0,
         maxsubiter               = 16000,
         start_inner_tol          = 1e-4,
```

```
standardize_data          = True,
accel                     = True,
nt                        = 0,
print_stata_style_summary = True)
```

The general rules on which arguments can be passed positionally and which should be passed by keyword, as well as how to estimate models without regressors (`x=[]`) or without fixed effects (omitting `fixedeffects`), are described in Section 2.1 and apply here. In short, `y`, `x`, can be passed positionally in this order, and all remaining options—of which `ppmlhdfe` has many—by keyword.

(a) Parameters:

- `y`: single entry list containing the name of the dependent variable as a string. It is the name of the column in `data` containing the explanatory variable. Must always be provided.
- `x`: string list of the names of the explanatory variables, i.e. names of the columns in `data` containing explanatory variables. Pass an empty list, `[]`, to estimate a model without any regressors (a pure fixed-effects or constant-only model). No default; this argument must be given (possibly as `[]`).
- `fixedeffects`: string list of the names of the fixed effects. Variables can be combined by "#", e.g. "state#year" creates state-year fixed effects. Default is `None`, which mirrors omitting Stata's `absorb()`: the model is then estimated with a single absorbed constant (an intercept `_cons`) and no high-dimensional fixed effects.
- `data`: pandas data frame containing all relevant variables. Must always be provided.

- `setype`: string, type of standard errors. Allowed values are "none" (no standard errors are computed), "homoskedastic" ("homo" / "iid" are synonyms), "heteroskedastic" ("r", "robust", "hetero" and "HC1" are synonyms), and "cluster" ("multiway", "clu", "cl", "c" are synonyms). Default is `''none''`.
- `clustvars`: string list of cluster variables. Maximum length 4, ignored if `setype` is not "cluster" (and required if it is). Default is `None`.
- `FEgen`: bool, if `True`, the observation-wise sum of the fixed effects (`fe_rowsum`) and the dictionary of individual fixed-effect estimates (`fixed_effects`) are computed and can be accessed after estimation (when the returned object is stored in `result`, then via `result.fe_rowsum` and `result.fixed_effects`, respectively). With the default `False` these two attributes are not present on the returned object (see Section 2.1). Default is `False`.
- `eta_0`: single-entry list containing the *name of a column in* `data` that holds a pre-computed linear predictor η, with one value per observation—for example `eta_0=["eta_start"]` when `data` contains a column `eta_start`. The supplied η must already include the fixed-effect contribution ($\mathbf{X}\boldsymbol{\beta}$ plus the fixed effects); a possible offset is supplied separately through `off` and added by the command, so it must not be part of `eta_0`. If set, `eta_0` overrides `start` (and forces `olsguess` to `False`). Default is `None`.
- `start`: strategy for the initial values of η. Accepts one of the following:
  - `"simple"` (default): $\eta = \ln\big((y + \overline{y})/2\big)$, the Stata `ppmlhdfe`-style default.
  - `"ols"`: use OLS on the fixed-effect-demeaned data, then take $\eta =$ ln(fitted values).

  - `dict`: user-supplied starting values for selected regression coefficients, e.g. {`"log_distance":-1.0, "contiguity": 0.4`}. Unspecified variables start at zero.

  In every case the fixed effects themselves receive no user-supplied starting values: with `start` the starting fixed-effect level is computed internally by projecting the default starting predictor onto the fixed-effect space, so only the non-fixed-effect coefficients $\boldsymbol{\beta}$ can be initialised (through a `dict`).

- `olsguess`: bool, deprecated. Kept for backward compatibility. If `True`, equivalent to `start="ols"`. Default is `False`.
- `off`: single entry list containing the string of the offset variable.
- `separation`: string, list of strings, bool, or `None`, default `"ir"`. Controls detection and removal of separated observations, mirroring the `separation()` option of Stata's `ppmlhdfe`. Allowed values:
  - `"fe"`: drop observations whose fixed-effect group has y constant across all observations (this is already performed unconditionally by the singleton/constant-y loop; listing it is informational).
  - `"ir"` / `"relu"`: iterative rectifier; detects separation arising from both fixed effects *and* regressors (Correia et al. 2026).
  - `"none"` / `"off"` / `False` / `None`: disable additional separation checks (the implicit "fe"-style check still runs).
  - `"default"` / `"auto"` / `True`: equivalent to `["ir"]` (the default, matching Stata).
  - list: apply all listed methods, e.g. `["fe","ir"]`.
- `sep_tol`: float, threshold above which an observation is flagged as a separation certificate by the iterative rectifier. Default is 10^{-5}.
- `sep_maxiter`: int, maximum number of outer iterations of the iterative rectifier. Default is 100.

- `sep_K_penalty`: float, penalty weight on $y > 0$ observations in the IR weighted regression. Too large can cause numerical issues in the inner weighted MAP; too small allows leakage onto $y > 0$ observations. Default is 10^6.
- `keepsingletons`: bool, default `False` (matching Stata's default). If `False`, drop any fixed-effect group with no within-group variation in y (singletons, constant-y groups, and classical fe-style separation). If `True`, only drop groups where y is identically zero—the minimum required for Poisson identification—and keep singletons and constant-positive-y groups. Useful for replicating analyses that report the full sample, but singletons carry no identifying information and can distort cluster-robust standard errors.
- `standardize_data`: bool, default `True` (matching Stata's `standardize_data(1)`). If `True`, each regressor is divided by its sample standard deviation before estimation, and the resulting coefficients and variance-covariance matrix are rescaled to original units at the end. This substantially improves numerical stability when regressors have very different magnitudes (a common case in trade gravity, where dummies sit next to log distances or large monetary values), and prevents the cluster-robust variance computation from becoming near-singular when a regressor's identifying variation is concentrated in a small subset of observations after fixed-effect absorption.

(b) Additional Parameters:

- `accel`: bool, if `True`, MAP employs conjugate gradient acceleration technique following Hernández-Ramos et al. (2011). Default is `True`.
- `nt`: integer, sets the number of threads to use. Default is `0`, which uses "system settings".

- `nocheck`: bool, if `True`, no consistency check is skipped. Default is `False`.
- `verb`: int $\in \{0, 1, 2\}$, level of output displayed (0 is none). Default is 0.
- `eps_beta`: float $\in (0, \infty)$, convergence criterion of the IRLS loop, $||\boldsymbol{\beta}^{(r+1)} - \boldsymbol{\beta}^{(r)}||_2 <$ `eps_beta`. Convergence is declared only once this criterion is satisfied in two consecutive iterations, mirroring Stata's convention. Default is 10^{-8}, matching Stata `ppmlhdfe`'s `tolerance(1e-8)`.
- `eps_MAP`: float $\in (0, \infty)$, target convergence criterion of the alternating projections algorithm. The *effective* inner tolerance applied at convergence is $\max(\min(\texttt{eps\_MAP}, 0.1 \cdot \texttt{eps\_beta}), 10^{-12})$, following Stata `reghdfe`'s rule that the inner solve must be at least ten times tighter than the outer IRLS criterion but never tighter than machine precision. Default is 10^{-9}, matching Stata `reghdfe`'s `itol(1e-9)`.
- `start_inner_tol`: float $\in (0, \infty)$, starting tolerance for the inner partialling-out loop on the first IRLS iteration. The inner tolerance is loosened to this value early in IRLS (when the outer $\boldsymbol{\beta}$-change is large) and tightens proportionally to the outer convergence measure across iterations, with the floor described under `eps_MAP`. This adaptive schedule avoids spending inner iterations on precision that the outer loop cannot yet exploit, while guaranteeing the required tightness at convergence. Default is 10^{-4}, matching Stata `reghdfe`'s `start_inner_tol(1e-4)`.
- `miniter`: minimum number of iterations of the IRLS loop. Default is 1.
- `maxiter`: maximum number of iterations of the IRLS loop. Default is 1000, matching Stata `ppmlhdfe`'s `maxiter(1000)`.
- `minsubiter`: minimum number of iterations within the demeaning process. Default is 0.
- `maxsubiter`: maximum number of iterations within the demeaning process. Default is 16000, matching Stata `reghdfe`'s `iterations(16000)`.

- `print_stata_style_summary`: bool, regression output table and Stata-style drop/separation notices are printed. Default is `True`.

(c) Returns a class object with the following attributes:

- `params`: point estimates.
- `bse`: standard errors if calculated, "NULL" otherwise.
- `covmat`: variance-covariance matrix if calculated, "NULL" otherwise.
- `covmat_nofix`: variance-covariance matrix prior to the "eigenvalue fix". Only present if `eigenvalue_fix` is `True`.
- `eigenvalue_fix`: `True` only if the covariance matrix was not positive semi-definite and the eigenvalue fix had to be applied (negative eigenvalues replaced by zero); otherwise the attribute is not set.
- `cov_type`: string, type of standard errors.
- `clustvars`: list, string list with cluster dimensions if `cov_type` is "cluster", "NULL" otherwise.
- `tvalues`: t-values for $H_0 = 0$.
- `pvalues`: p-values for a two-sided t-test with $H_0 = 0$.
- `nobs`: int, number of observations used for estimation. Some observations may have been excluded and do not count in here.
- `df_model`: number of parameters in X plus the theoretical upper bound of fixed effects (collinear fixed effects count as estimated parameters here; this may lead to a slight over-estimation of small-sample corrected standard errors).
- `df_resid`: `nobs` − `df_model`.
- `aic`: Akaike Information Criterion.
- `bic`: Bayesian Information Criterion.
- `llf`: value of the log-likelihood.

- `deviance`: sum of the squared deviance residuals. For $y = 0$, the deviance is approximated using L'Hôpital's rule.
- `pearson_chi2`: Pearson's χ^2-value.
- `family_name`: "Poisson distribution".
- `family_link`: "Natural logarithm function", $(g(\cdot))$.
- `scale`: 1, per definition of the Poisson distribution.
- `method`: "IRLS", Iteratively re-weighted least squares.
- `fit_history`: dict with key 'iteration', which is the number of iterations until convergence.
- `fittedvalues`: array of the fitted values $\hat{y} = \mu = g^{-1}(\eta) = \exp(\eta)$.
- `fe_rowsum`: array of the observation-wise sum of the fixed effects $\boldsymbol{\eta} - \mathbf{x}'\boldsymbol{\beta}$. Only present if `FEgen=True` (see Section 2.1); otherwise the attribute is not set.
- `fixed_effects`: dictionary of all estimates of all fixed effects specified (the explicit constant `_cons`, the per-observation centered fixed effects, and the group-level values). Only present if `FEgen=True`; otherwise the attribute is not set.
- `resid_additive`: array of additive residuals $(y - \hat{y})$.
- `resid_multiplicative`: array of multiplicative residuals $(y/\hat{y})$.
- `msg`: list, information/warning messages during the estimation process.
- `formula_R`: string, formula as you would type it in R.
- `formula_Stata`: string, formula as you would type it in Stata.
- `model`: GLM class, for compatibility with the gme-package.
- `separation_methods`: list of strings, the separation methods that were applied (e.g. `[''ir'']`).
- `num_separated`: int, number of observations dropped by the iterative rectifier. Equal to `N_dropped_ir_sep`.

- `keepsingletons`: bool, the value of the `keepsingletons` argument passed in.
- `N_full_initial`: int, number of observations in the input data frame before any drops.
- `N_dropped_invalid`: int, number of observations dropped due to missing, infinite, or negative values.
- `N_dropped_fe_sep`: int, number of observations dropped by the singleton / constant-y / fe-style separation loop.
- `N_dropped_ir_sep`: int, number of observations dropped by the iterative rectifier.

(d) Example:

```
from fasthdfe import ppmlhdfe
result = ppmlhdfe(['trade'],
      ['LN_DIST','agree_pta','contiguity','common_language'],
      fixedeffects = ['exp_ind#industry_id#year',
                      'imp_ind#industry_id#year',
                      'exp_ind#imp_ind#industry_id'],
      data = df,
      eps_beta=1e-5,
      setype='cluster',
      clustvars=['exp_ind', 'imp_ind','year',
      'industry_id'])
```

Appendix B Estimation Details

This appendix collects the derivations underlying the estimators summarized in Section 3: the linear within-estimator, the IRLS scheme for PPML, and the calculation of the standard errors.

B.1 Linear Model

Consider the following linear model:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \sum_{q=1}^{Q} \mathbf{D}_q \boldsymbol{\delta}_q + \boldsymbol{\varepsilon}, \tag{B1}$$

where $\mathbf{y}$ denotes the vector of size $N \times 1$ containing the dependent variable, with N denoting the number of observations, $\mathbf{X}$ denotes the $N \times K$ matrix of explanatory variables, $\boldsymbol{\beta}$ is the $K \times 1$ parameter vector of interest and $\mathbf{D}_q$ denotes the q-th set out of Q sets of fixed effects, with corresponding parameter vector $\boldsymbol{\delta}_q$. $\boldsymbol{\varepsilon}$ denotes the error term. For ease of notation, we will assume for the remainder of the section that $Q = 1$, i. e., that there is only a single set of fixed effects. However, the methods can be applied to infinite sets of fixed effects as described in Gaure (2013). Minimizing the sum of squared residuals leads to the following first order conditions, i.e., the normal equations from the partitioned regression (see Greene 2020, page 75):

$$\mathbf{X}'\mathbf{X}\hat{\boldsymbol{\beta}} + \mathbf{X}'\mathbf{D}\hat{\boldsymbol{\delta}} = \mathbf{X}'\mathbf{y}, \tag{B2}$$

$$\mathbf{D}'\mathbf{X}\hat{\boldsymbol{\beta}} + \mathbf{D}'\mathbf{D}\hat{\boldsymbol{\delta}} = \mathbf{D}'\mathbf{y}. \tag{B3}$$

Solving (B3) for $\hat{\boldsymbol{\delta}}$ and inserting in (B2) yields:

$$\mathbf{X}'\mathbf{X}\hat{\boldsymbol{\beta}} + \mathbf{X}'\mathbf{D}\left[(\mathbf{D}'\mathbf{D})^{-1}\mathbf{D}'\mathbf{y} - (\mathbf{D}'\mathbf{D})^{-1}\mathbf{D}'\mathbf{X}\hat{\boldsymbol{\beta}}\right] = \mathbf{X}'\mathbf{y}$$

$$\Leftrightarrow \mathbf{X}'\mathbf{X}\hat{\boldsymbol{\beta}} - \mathbf{X}'\mathbf{D}(\mathbf{D}'\mathbf{D})^{-1}\mathbf{D}'\mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}'\mathbf{y} - \mathbf{X}'\mathbf{D}(\mathbf{D}'\mathbf{D})^{-1}\mathbf{D}'\mathbf{y}$$

$$\Leftrightarrow \hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbb{M}\mathbf{X})^{-1}\mathbf{X}'\mathbb{M}\mathbf{y}, \tag{B4}$$

where $\mathbb{M} := \mathbf{I} - \mathbf{D}(\mathbf{D}'\mathbf{D})^{-1}\mathbf{D}'$ is the so-called annihilator matrix, $\mathbf{I}$ denotes an identity matrix of the appropriate size, and the transpose of any matrix $\mathbf{M}$ is denoted by $\mathbf{M}'$. Hats denote estimates.

The actual calculation of $\mathbb{M}$ is computationally expensive and sometimes infeasible as it is of size $N \times N$. However, the structure of $\mathbb{M}$ is known as each row of $\mathbf{D}$ consists of a single 1 and 0 otherwise. Using the fact that $\mathbb{M}$ is idempotent and symmetric, we can rewrite (B4) as:

$$\hat{\boldsymbol{\beta}} = \big((\mathbb{M}\mathbf{X})'(\mathbb{M}\mathbf{X})\big)^{-1}(\mathbb{M}\mathbf{X})'(\mathbb{M}\mathbf{y}),$$

$$\Leftrightarrow \hat{\boldsymbol{\beta}} = (\tilde{\mathbf{X}}'\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}'\tilde{\mathbf{y}}, \tag{B5}$$

where $\tilde{\mathbf{X}} := \mathbb{M}\mathbf{X}$ is the within-transformation applied to the explanatory variables and $\tilde{\mathbf{y}}$ is defined analogously, i.e., $\tilde{\mathbf{y}} := \mathbb{M}\mathbf{y}$. Equation (B5) restates the well-known OLS formula, but with transformed variables. For more than one set of fixed effects $\mathbb{M}$ loses its sparse structure, but using the Method of Alternating Projections (MAP) we still can calculate the correct $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{y}}$ by applying the group-wise demeaning in an alternating fashion, as described in Gaure (2013). The sub-matrix of the Hessian associated with the parameter vector $\hat{\boldsymbol{\beta}}$ can be derived using the block inverse formula and is given by:

$$\mathcal{H}_\beta = (\mathbf{X}'\mathbb{M}\mathbf{X})^{-1} = \big((\mathbb{M}\mathbf{X})'(\mathbb{M}\mathbf{X})\big)^{-1} = (\tilde{\mathbf{X}}'\tilde{\mathbf{X}})^{-1}. \tag{B6}$$

B.2 PPML

Next, we consider the following model:

$$\mathbf{y} = \exp\left(\mathbf{X}\boldsymbol{\beta} + \sum_{q=1}^{Q} \mathbf{D}_q \boldsymbol{\delta}_q\right) + \boldsymbol{\varepsilon} = \exp(\mathbf{Z}\boldsymbol{\gamma}) + \boldsymbol{\varepsilon} = \exp(\boldsymbol{\eta}) + \boldsymbol{\varepsilon}, \tag{B7}$$

with components defined as in the previous subsection.

Note that, while most PPML applications in the literature use an additive error term, the multiplicative error term treatment is perfectly valid too. Santos Silva and Tenreyro (2006) show on p. 644 that $y_i = \exp(\mathbf{x}_i\boldsymbol{\beta}) + \varepsilon_i$ can be expressed as $y_i = \exp(\mathbf{x}_i\boldsymbol{\beta})\epsilon_i$, with $\epsilon_i = 1 + \varepsilon_i / \exp(\mathbf{x}_i\boldsymbol{\beta})$ and $E[\epsilon_i|x] = 1$. Further, in footnote 7, they state "Whether the error enters additively or multiplicatively is irrelevant for our purposes." (p. 643). Already Wooldridge (1992) noted this: "Another argument for defining economic quantities in terms of $E(y|\mathbf{x})$ is that it circumvents the issue of whether the "disturbances" in a model are additive or multiplicative. When $y \geq 0$ and interest centers on $\mu(\mathbf{x}) \equiv E(y|\mathbf{x})$, the disturbances can be multiplicative or additive: The models

$$y = \mu(\mathbf{x}) + \varepsilon, \qquad E(\varepsilon|\mathbf{x}) = 0, \tag{B8}$$

$$y = \mu(\mathbf{x})\epsilon, \qquad \epsilon \geq 0 \qquad E(\epsilon|\mathbf{x}) = 1, \tag{B9}$$

are both correct and observationally equivalent without further restrictions on ε and ϵ; just define $\varepsilon \equiv y - \mu(\mathbf{x})$ and $\epsilon \equiv y/\mu(\mathbf{x})$ (assuming that $P[\mu(\mathbf{x}) > 0] = 1$). [...] (B8) and (B9) are simply different ways of stating that $E(y|\mathbf{x}) = \mu(\mathbf{x})$." (p. 938). Consistent estimates with PPML, as a Pseudo-Maximum-Likelihood estimator, only require the correct specification of the conditional expectation function, which is assumed to be the same in the additive and multiplicative case. See Mullahy (1997) and Windmeijer and

Santos Silva (1997) for additional discussion of the use of additive vs. multiplicative error formulations.

To estimate the model, one can either maximize the log-likelihood function or solve the moment conditions $E[\mathbf{Z}'\boldsymbol{\varepsilon}] = \mathbf{0}$. Both methods are equivalent, as the first order conditions with respect to $\boldsymbol{\gamma}$ corresponding to the maximization of the log-likelihood function coincide with the moment conditions, given by:

$$\frac{1}{N}\mathbf{Z}'\big(\mathbf{y} - \exp(\mathbf{Z}\boldsymbol{\gamma})\big) = \frac{1}{N}\mathbf{Z}'\big(\mathbf{y} - \exp(\boldsymbol{\eta})\big) = \mathbf{0}, \tag{B10}$$

where $\boldsymbol{\eta} = \mathbf{Z}\boldsymbol{\gamma}$. The parameter updates can be calculated using the Iteratively Re-weighted Least Squares (IRLS) algorithm, which is the solution of a weighted linear regression with a working dependent variable given by $s_i = \hat{\eta}_i + y_i/\exp(\hat{\eta}_i) - 1$ (see, Correia et al. 2026, for example) and weights given by $\exp(\hat{\eta}_i)$. Thus we can apply the methods described in the previous section in order to update the parameter estimates without inversion of $\big(\mathbf{Z}'\text{diag}\big(\exp(\hat{\boldsymbol{\eta}})\big)\mathbf{Z}\big)$. The last ingredient for the IRLS-algorithm is the update of the linear prediction $\hat{\boldsymbol{\eta}}$. Here we can utilize the Frisch-Waugh-Lovell theorem based on Frisch and Waugh (1933) and Lovell (1963), which states that the residuals of our projected system given by $\tilde{\mathbf{s}} - \tilde{\mathbf{X}}\hat{\boldsymbol{\beta}}$ and the residuals of the full system given by $\mathbf{s} - \hat{\boldsymbol{\eta}}$ are identical, so that the updated linear predictor is given by $\hat{\boldsymbol{\eta}} = \mathbf{s} - \tilde{\mathbf{s}} + \tilde{\mathbf{X}}\hat{\boldsymbol{\beta}}$. The IRLS-algorithm can then be described as:

Algorithm 1 IRLS-Algorithm used for `ppmlhdfe`

0. form an initial estimate of $\hat{\boldsymbol{\eta}}$, set convergence criterion ϵ

while $(||\hat{\boldsymbol{\beta}}^{(r+1)} - \hat{\boldsymbol{\beta}}^{(r)}||_2 > \epsilon)$

1. calculate $\hat{\boldsymbol{\mu}}^{(r)} = \exp(\hat{\boldsymbol{\eta}}^{(r)})$
2. calculate $\mathbf{s}^{(r)} = \hat{\boldsymbol{\eta}}^{(r)} + \mathbf{y} \oslash \hat{\boldsymbol{\mu}}^{(r)} - 1$
3. form the estimates of $\tilde{\mathbf{X}}^{(r)}$ and $\tilde{\mathbf{s}}^{(r)}$ using MAP
4. calculate $\hat{\boldsymbol{\beta}}^{(r+1)} = \left(\tilde{\mathbf{X}}^{(r)\prime}\mathrm{diag}\left(\hat{\boldsymbol{\mu}}^{(r)}\right)\tilde{\mathbf{X}}^{(r)}\right)^{-1}\tilde{\mathbf{X}}^{(r)\prime}\mathrm{diag}\left(\hat{\boldsymbol{\mu}}^{(r)}\right)\tilde{\mathbf{s}}^{(r)}$
5. calculate $\hat{\boldsymbol{\eta}}^{(r+1)} = \mathbf{s}^{(r)} - \tilde{\mathbf{s}}^{(r)} + \tilde{\mathbf{X}}^{(r)}\hat{\boldsymbol{\beta}}^{(r+1)}$
6. compute the candidate deviance $D^{(r+1)} = 2\sum_i \left[y_i \log\left(y_i/\hat{\mu}_i^{(r+1)}\right) - \left(y_i - \hat{\mu}_i^{(r+1)}\right)\right]$, with the convention $0 \cdot \log(0/\mu) = 0$
7. if $D^{(r+1)} > D^{(r)}$: step-halve toward the previous iterate by setting $\hat{\boldsymbol{\beta}}^{(r+1)} \leftarrow \frac{1}{2}\left(\hat{\boldsymbol{\beta}}^{(r)} + \hat{\boldsymbol{\beta}}^{(r+1)}\right)$ and $\hat{\boldsymbol{\eta}}^{(r+1)} \leftarrow \frac{1}{2}\left(\hat{\boldsymbol{\eta}}^{(r)} + \hat{\boldsymbol{\eta}}^{(r+1)}\right)$; recompute $\hat{\boldsymbol{\mu}}^{(r+1)}$ and $D^{(r+1)}$, and repeat until $D^{(r+1)} \leq D^{(r)}$ (or a safeguard limit on the number of halvings is reached)
8. go to step 1

end

The superscript (r) is used to denote estimates in the r-th iteration and $\oslash$ is used to denote element-wise division.

Steps 6 and 7 implement step-halving on the deviance: if a candidate IRLS update would increase the deviance, the step is halved toward the previous iterate until the deviance is non-increasing. The same safeguard is implemented internally in Stata's `ppmlhdfe` (Correia et al. 2020). The IRLS update is exact only to first order: when the quadratic approximation underlying the working response $\mathbf{s}^{(r)}$ breaks down, the unconstrained step can push $\hat{\boldsymbol{\eta}}$ into a region of the parameter space where the deviance is much larger than at the previous iterate—typically because the implicit gauge of

the absorbed fixed effects has drifted in a weakly identified cell, even while $\hat{\boldsymbol{\beta}}$ remains well-behaved. The deviance check in step 6 detects this; the halving in step 7 rolls the step partially back, jointly in $(\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\eta}})$ so that the halved iterate remains a valid (smaller-magnitude) IRLS step. In our implementation up to thirty halvings are attempted before the step is accepted with a warning. The outer convergence criterion in the `while`-condition must additionally be satisfied in two consecutive iterations before the loop terminates, mirroring Stata's two-hit rule.

The sub-matrix of the Hessian associated with the parameter vector $\hat{\boldsymbol{\beta}}$ is given by:

$$\mathcal{H}_\beta = (\mathbf{X}'\text{diag}(\hat{\boldsymbol{\mu}})\mathbb{M}\mathbf{X})^{-1} = \big((\mathbb{M}\mathbf{X})'\text{diag}(\hat{\boldsymbol{\mu}})(\mathbb{M}\mathbf{X})\big)^{-1} = (\tilde{\mathbf{X}}'\text{diag}(\hat{\boldsymbol{\mu}})\tilde{\mathbf{X}})^{-1}. \quad \text{(B11)}$$

B.3 Standard Errors

So far, we mainly discussed the computation of the point estimates. Estimating standard errors for PPML estimates with high-dimensional fixed effects is also challenging. One could construct from the iterative procedure the complete Hessian matrix and invert it to obtain standard errors. However, with many fixed effects, this may be impracticable. Recent advances in the related literature offer better alternatives. Figueiredo et al. (2015) show for the case of a PPML model with two-way HDFEs that the variance-covariance matrix of a Poisson regression is proportional to that of an appropriately weighted linear regression and therefore the Frish-Waugh-Lovell theorem can be applied. This strategy can be extended to the case of multiple high-dimensional fixed effects.

Besides the technical challenges to obtain standard errors, we allow for error correlation (or "clustering") patterns that are reasonable for the data and model at hand. In particular, Egger and Tarlea (2015) argue that standard errors for a panel-data gravity model should allow for simultaneous correlations across all three main dimensions of the panel—exporter, importer, and time. This can be done by "multiway" clustering (see Cameron et al. 2011). Note that clustering simultaneously on i, j, and

t allows for correlation in the error term within all six possible cluster dimensions $\{i, j, t, it, jt, ij\}$. It therefore nests the typical practice of assuming errors are solely clustered across time within each country-pair ij. Multiway clustering on exporter, importer, and time is therefore expected to lead to more conservative inferences (Egger and Tarlea 2015).

Under the assumption of identical and independently distributed (i.i.d.) errors it follows from the Hessian matrices in the previous sections that the variance-covariance estimate is given by:

$$V[\hat{\boldsymbol{\beta}}] = (\tilde{\mathbf{X}}'\mathbf{W}\tilde{\mathbf{X}})^{-1}, \tag{B12}$$

where $\tilde{\mathbf{X}}$ and $\mathbf{W}$ are based on the respective estimates in the last iteration. In the linear case the weights are given by $\mathbf{W} = \mathbf{I}$, i. e., the identity matrix. For PPML weights are given by $\mathbf{W} = \text{diag}(\hat{\boldsymbol{\mu}})$. By utilizing the block inverse formula the inversion of the whole system can be avoided, which is useful for two reasons. First, the size of $\mathbf{X}$ is usually quiet manageable and second, the inverse of $\mathbf{Z}'\mathbf{Z}$ might not exist due to collinearity of the fixed effects. However, it is neither necessary to calculate a point-estimate nor a standard error for any of the fixed effects.

Allowing for heteroskedasticity, the Eicker-White standard errors (Eicker 1963; White 1980) can be estimated as follows:

$$V_{rob}[\hat{\boldsymbol{\beta}}] = \frac{N}{N-K}(\tilde{\mathbf{X}}'\mathbf{W}\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}'\text{diag}(\hat{\mathbf{u}} \odot \hat{\mathbf{u}})\tilde{\mathbf{X}}(\tilde{\mathbf{X}}'\mathbf{W}\tilde{\mathbf{X}})^{-1}, \tag{B13}$$

where $\odot$ is used to denote element-wise multiplication, $\mathbf{W}$ is the identity matrix in the linear case and $\text{diag}(\hat{\boldsymbol{\mu}})$ for PPML. Further, the residual terms are given by $\hat{\mathbf{u}} = \hat{\boldsymbol{\varepsilon}}$ for both models. The $N/(N-K)$ is a small-sample degrees-of-freedom correction following Stata's implementation.

Relaxing the error term assumptions to allow for error correlation within defined groups (clusters), one can adjust the formula for the heteroskedasticity-robust standard errors (which are a special case of clustered standard errors, as each observation is a cluster on its own). Thus, the cluster-robust standard errors can be calculated by:

$$V_{clu}[\hat{\boldsymbol{\beta}}] = \frac{N-1}{N-K-N_{fe}} \underbrace{(\tilde{\mathbf{X}}'\mathbf{W}\tilde{\mathbf{X}})^{-1}}_{\hat{\mathbf{V}}} \underbrace{\Big(\sum_{g=1}^{G} \frac{\min_G}{(\min_G - 1)} \tilde{\mathbf{X}}_g' \hat{\mathbf{u}}_g \hat{\mathbf{u}}_g' \tilde{\mathbf{X}}_g\Big)}_{\hat{\mathbf{M}}} \underbrace{(\tilde{\mathbf{X}}'\mathbf{W}\tilde{\mathbf{X}})^{-1}}_{\hat{\mathbf{V}}}, \tag{B14}$$

with the definitions from the Eicker-White standard errors, but with adjusted "meat" $\hat{\mathbf{M}}$ of the sandwich estimate. $\min_G / (\min_G - 1)$ is a small-sample degree-of-freedom correction based on the minimum number of clusters among the individual clustering variables ($\min_G$) following Stata's implementation and applied to each component of the "meat". The term $(N-1)/(N-K-N_{fe})$ is an overall small-sample degrees-of-freedom correction, again following Stata's implementation, where N_{fe} denotes the total number of non-redundant fixed effects. To be computationally feasible for a large amount of data, we use the sum notation here, as the full matrix representation requires calculation of two $N \times N$ matrices, which may lead to memory issues.

Following Cameron et al. (2011) to calculate the multiway-clustered standard errors with D dimensions of clustering, the estimate for $\hat{\mathbf{M}}$ has to be replaced by:

$$\hat{\mathbf{M}} = \sum_{||\mathbf{r}||=k, \mathbf{r}\in\mathbb{R}} (-1)^{k+1} \hat{\mathbf{M}}_{\mathbf{r}}, \tag{B15}$$

where $\mathbf{r}$ is a D-vector, with dth coordinate equal to r_d, and we define the set $R \equiv \{\mathbf{r} : r_d \in \{0,1\}, d = 1,2,...,D, \mathbf{r} \neq \mathbf{0}\}$ where the elements of R index whether two observations are joint members of at least one cluster. l and m denote specific observations. $I_{\mathbf{r}}(l,m)$ takes the value one if observations l and m are both members of

all clusters for which $r_d = 1$. $||\mathbf{r}||$ denotes the ℓ_1-norm of the vector $\mathbf{r}$. $\hat{\mathbf{M}}_{\mathbf{r}}$ is defined as follows:

$$\hat{\mathbf{M}}_{\mathbf{r}} = \sum_{i=1}^{N} \sum_{j=1}^{N} \tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_j' \hat{\mathbf{u}}_i \hat{\mathbf{u}}_j I_{\mathbf{r}}(i,j).$$

Appendix C Detailed Estimation Output

For completeness, this appendix reproduces, for each specification discussed in Section 5, the exact commands and the full command-line output of both our Python implementation and Stata's `reghdfe`/`ppmlhdfe`.

C.1 GME Data: Linear Model

The first specification assumes homoskedastic standard errors:

Python command:

```
reghdfe(['lntrade_value'],
    ['log_distance','agree_pta', 'common_language','contiguity'],
    fixedeffects = ['importer#year',
                    'exporter#year',
                    'exporter#importer'],
    data = self._df, setype='homoskedastic',
    print_stata_style_summary=True, verb=0)
```

Output:

```
==============================================================================
HDFE Linear regression                            Number of obs   =   80,350
Absorbing 3 HDFE groups                           F(   2,  73711) =     3.54
Homoskedastic standard errors                         Prob > F    =   0.0291
                                                    R-squared     =   0.9316
                                                  Adj R-squared   =   0.9255
                                                    Within R-sq.  =   0.0001
                                                       Root MSE   =   0.8657

                                    Std. Err.
             Coefficient    Std. Err.           t    P>|z|    [95% conf. interval]
----------  -------------  -----------  -----------  -------  ----------------------
 agree_pta       0.0410733    0.0213120         1.93    0.054  -0.0006981   0.0828448
contiguity       0.9578766    0.5328071         1.80    0.072  -0.0864232   2.0021764
     _cons      18.4002353    0.0206440       891.31    0.000  18.3597730  18.4406975

                    Categories    -  Redundant    =  Num. Coefs
-----------------  ------------  --------------  ---------------  --
importer#year              1383               0             1383
exporter#year              1637              27             1610
exporter#importer          3706              62             3644  ?
? = number of redundant parameters may be higher
```

Stata command:

```
reghdfe ln_tradevalue log_distance agree_pta common_language
  contiguity, absorb(exp_ind#year imp_ind#year exp_ind#imp_ind)
```

Output:

```
HDFE Linear regression                            Number of obs   =     80,350
Absorbing 3 HDFE groups                           F(   2,  73711) =       3.54
                                                  Prob > F        =     0.0291
                                                  R-squared       =     0.9316
                                                  Adj R-squared   =     0.9255
                                                  Within R-sq.    =     0.0001
                                                  Root MSE        =     0.8657

------------------------------------------------------------------------------
 ln_tradevalue | Coefficient  Std. err.      t    P>|t|     [95% conf. interval]
---------------+--------------------------------------------------------------
  log_distance |          0  (omitted)
     agree_pta |   .0410733    .021312     1.93   0.054    -.0006981    .0828448
common_language |          0  (omitted)
    contiguity |   .9578766   .5328071     1.80   0.072    -.0864232    2.002176
         _cons |   18.40024    .020644   891.31   0.000     18.35977     18.4407
------------------------------------------------------------------------------

Absorbed degrees of freedom:
-----------------------------------------------------------+
      Absorbed FE | Categories  - Redundant  = Num. Coefs |
------------------+----------------------------------------|
     imp_ind#year |      1383           0        1383     |
     exp_ind#year |      1637          27        1610     |
  exp_ind#imp_ind |      3706          62        3644    ?|
-----------------------------------------------------------+
? = number of redundant parameters may be higher
```

The second specification assumes robust standard errors:

Python command (assuming the same `import`-statement as before):

```
reghdfe(['lntrade_value'],
    ['log_distance','agree_pta', 'common_language','contiguity'],
    fixedeffects = ['importer#year', 'exporter#year',
                'exporter#importer'],
    data = self._df, setype='r',
    print_stata_style_summary=True, verb=0)
```

Output:

```
============================================================================
HDFE Linear regression                            Number of obs   =   80,350
Absorbing 3 HDFE groups                           F(   2,  73711) =     7.52
Robust standard errors                                Prob > F    =   0.0005
                                                    R-squared     =   0.9316
                                                  Adj R-squared   =   0.9255
                                                    Within R-sq.  =   0.0001
                                                       Root MSE   =   0.8657

                                    Robust
             Coefficient    Std. Err.         t    P>|z|    [95% conf. interval]
----------  -------------  -----------  --------  -------  ----------------------
 agree_pta      0.0410733    0.0166891      2.46    0.014   0.0083628   0.0737839
contiguity      0.9578766    0.3297494      2.90    0.004   0.3115691   1.6041841
     _cons     18.4002353    0.0134503   1368.01    0.000  18.3738726  18.4265979

                    Categories    -  Redundant    =  Num. Coefs
-----------------  ------------  --------------  ---------------  --
importer#year              1383               0             1383
exporter#year              1637              27             1610
exporter#importer          3706              62             3644  ?
? = number of redundant parameters may be higher
```

Stata command:

```
reghdfe ln_tradevalue log_distance agree_pta common_language contiguity,
  absorb(exp_ind#year imp_ind#year exp_ind#imp_ind) vce(robust)
```

Output:

```
HDFE Linear regression                            Number of obs   =     80,350
Absorbing 3 HDFE groups                           F(   2,  73711) =       7.52
                                                  Prob > F        =     0.0005
                                                  R-squared       =     0.9316
                                                  Adj R-squared   =     0.9255
                                                  Within R-sq.    =     0.0001
                                                  Root MSE        =     0.8657

------------------------------------------------------------------------------
             |               Robust
ln_tradevalue | Coefficient  std. err.      t    P>|t|     [95% conf. interval]
--------------+---------------------------------------------------------------
log_distance |          0  (omitted)
   agree_pta |   .0410733   .0166891     2.46   0.014     .0083628    .0737839
common_language |          0  (omitted)
 contiguity |   .9578766   .3297494     2.90   0.004     .3115691    1.604184
       _cons |   18.40024   .0134503  1368.01   0.000     18.37387     18.4266
------------------------------------------------------------------------------

Absorbed degrees of freedom:
-----------------------------------------------------------+
     Absorbed FE | Categories  - Redundant  = Num. Coefs |
-----------------+-----------------------------------------|
   imp_ind#year |      1383           0        1383     |
   exp_ind#year |      1637          27        1610     |
exp_ind#imp_ind |      3706          62        3644    ?|
-----------------------------------------------------------+
? = number of redundant parameters may be higher
```

The third specification clusters standard errors by exporter, importer, and year, as suggested for gravity equation estimation by Egger and Tarlea (2015):

Python command (assuming the same `import`-statement as before):

```
reghdfe(['lntrade_value'],
    ['log_distance','agree_pta','common_language','contiguity'],
    fixedeffects = ['importer#year',
                    'exporter#year',
                    'exporter#importer'],
    data = self._df, setype='cluster',
    print_stata_style_summary=True, verb=0,
    clustvars=['importer', 'exporter', 'year'])
```

Output:

```
==============================================================================
HDFE Linear regression                            Number of obs   =   80,350
Absorbing 3 HDFE groups                           F(   2,     26) =     3.58
Statistics robust to heteroskedasticity               Prob > F    =   0.0423
                                                    R-squared     =   0.9316
Number of clusters (importer) =       61          Adj R-squared   =   0.9254
Number of clusters (exporter) =       62            Within R-sq.  =   0.0001
Number of clusters (    year) =       27              Root MSE    =   0.8662
                 (Std. err. adjusted for 27 clusters in importer exporter year)

                                    Robust
             Coefficient    Std. Err.          t    P>|z|     [95% conf. interval]
----------  -------------  -----------  --------  -------  ----------------------
 agree_pta       0.0410733    0.0457739      0.90    0.378  -0.0530163   0.1351630
contiguity       0.9578766    0.3614618      2.65    0.014   0.2148811   1.7008720
     _cons      18.4002353    0.0241584    761.65    0.000  18.3505771  18.4498935

                    Categories    -  Redundant    =  Num. Coefs
-----------------  ------------  --------------  ---------------  --
importer#year              1383            1383                0  *
exporter#year              1637            1637                0  *
exporter#importer          3706            3706                0  *
* = FE nested within cluster; treated as redundant for DoF computation
```

Stata command:

```
reghdfe ln_tradevalue log_distance agree_pta common_language contiguity,
  absorb(exp_ind#year imp_ind#year exp_ind#imp_ind) vce(cluster exp_ind
  imp_ind year)
```

Output:

```
HDFE Linear regression                            Number of obs   =     80,350
Absorbing 3 HDFE groups                           F(   2,     26) =       3.58
Statistics robust to heteroskedasticity           Prob > F        =     0.0423
                                                  R-squared       =     0.9316
Number of clusters (exp_ind) =         62         Adj R-squared   =     0.9254
Number of clusters (imp_ind) =         61         Within R-sq.    =     0.0001
Number of clusters (year)    =         27         Root MSE        =     0.8662

                    (Std. err. adjusted for 27 clusters in exp_ind imp_ind year)
------------------------------------------------------------------------------
               |               Robust
  ln_tradevalue | Coefficient  std. err.      t    P>|t|     [95% conf. interval]
---------------+--------------------------------------------------------------
   log_distance |          0  (omitted)
      agree_pta |   .0410733   .0457745     0.90   0.378    -.0530175    .1351642
common_language |          0  (omitted)
     contiguity |   .9578766   .3614663     2.65   0.014     .2148719    1.700881
          _cons |   18.40024   .0241587   761.64   0.000     18.35058    18.44989
------------------------------------------------------------------------------

Absorbed degrees of freedom:
-----------------------------------------------------------+
       Absorbed FE | Categories  - Redundant  = Num. Coefs |
-------------------+---------------------------------------|
      imp_ind#year |      1383        1383           0    *|
      exp_ind#year |      1637        1637           0    *|
   exp_ind#imp_ind |      3706        3706           0    *|
-----------------------------------------------------------+
* = FE nested within cluster; treated as redundant for DoF computation
```

C.2 GME Data: PPML

The first specification assumes robust standard errors:[1]

Python command:

```
ppmlhdfe(['trade_value'],
    ['log_distance','agree_pta','common_language','contiguity'],
    fixedeffects = ['importer#year', 'exporter#year', 'exporter#importer'],
    data = self._df, setype='r',
    print_stata_style_summary=True, verb=0)
```

Output:

```
==============================================================================
HDFE PPML regression                                  No. of obs    =   82,729
Absorbing 3 HDFE groups                               Residual df   =   76,086
                                                      Wald chi2( 2) =    24.06
Deviance       =  7.11108e+12                         Prob > chi2   =   0.0000
Log pseudolikelihood = -3.55554e+12                   Pseudo R2     =   0.9908
------------------------------------------------------------------------------
                             Robust
             Coefficient    Std. Err.       z    P>|z|     [95% conf. interval]
----------  -------------  -----------  ------  -------  ---------------------
 agree_pta     -0.0321458    0.0177813   -1.81    0.071   -0.0669965   0.0027050
contiguity      0.6819601    0.1469345    4.64    0.000    0.3939738   0.9699463
     _cons     23.4737940    0.0369114  635.95    0.000   23.4014490  23.5461391

                    Categories    -  Redundant    =  Num. Coefs
-----------------  ------------  --------------  ---------------  --
importer#year              1383               0             1383
exporter#year              1637              27             1610
exporter#importer          3710              62             3648  ?
? = number of redundant parameters may be higher
```

[1] As we are estimating pseudo-maximum likelihood and know that our standard errors will not be homoskedastic, we do not provide results for homoskedastic errors in this case.

Stata command:

```
ppmlhdfe trade_value log_distance agree_pta common_language contiguity,
  absorb(exp_ind#year imp_ind#year exp_ind#imp_ind) vce(robust)
```

Output:

```
HDFE PPML regression                                  No. of obs      =     82,729
Absorbing 3 HDFE groups                               Residual df     =     76,086
                                                      Wald chi2(2)    =      24.06
Deviance             =  7.11106e+12                   Prob > chi2     =     0.0000
Log pseudolikelihood = -3.55553e+12                   Pseudo R2       =     0.9908
------------------------------------------------------------------------------
             |               Robust
 trade_value | Coefficient  std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
log_distance |          0  (omitted)
   agree_pta |  -.0321458   .0177812    -1.81   0.071    -.0669963    .0027048
common_language |          0  (omitted)
 contiguity |   .6819601   .1469336     4.64   0.000     .3939756    .9699446
      _cons |    23.4738   .0369112   635.95   0.000     23.40145    23.54614
------------------------------------------------------------------------------

Absorbed degrees of freedom:
-----------------------------------------------------------+
  Absorbed FE | Categories  - Redundant  = Num. Coefs |
--------------+--------------------------------------------|
 imp_ind#year |      1383           0        1383     |
 exp_ind#year |      1637          27        1610     |
exp_ind#imp_ind |      3710          62        3648    ?|
-----------------------------------------------------------+
? = number of redundant parameters may be higher
```

The second specification clusters standard errors by exporter, importer, and year, as suggested for gravity equation estimation by Egger and Tarlea (2015):

Python command (assuming the same `import`-statement as before):

```
ppmlhdfe(['trade_value'],
    ['log_distance','agree_pta','common_language','contiguity'],
    fixedeffects = ['importer#year', 'exporter#year','exporter#importer'],
    data = self._df, setype='cluster',
    print_stata_style_summary=True, verb=0,
    clustvars=['importer', 'exporter', 'year'])
```

Output:

```
==============================================================================
HDFE PPML regression                                  No. of obs    =   82,729
Absorbing 3 HDFE groups                               Residual df   =       26
                                                     Wald chi2( 2)  =    301.20
Deviance       =  7.11108e+12                         Prob > chi2   =   0.0000
Log pseudolikelihood = -3.55554e+12                   Pseudo R2     =   0.9908
Number of clusters (importer)=       61
Number of clusters (exporter)=       62
Number of clusters (    year)=       27
                  (Std. err. adjusted for 27 clusters in importer exporter year)
------------------------------------------------------------------------------
                             Robust
             Coefficient    Std. Err.        z    P>|z|    [95% conf. interval]
----------  -------------  -----------  -------  -------  ---------------------
 agree_pta     -0.0321458    0.0467764    -0.69    0.492  -0.1238258   0.0595343
contiguity      0.6819601    0.0476739    14.30    0.000   0.5885210   0.7753992
     _cons     23.4737940    0.0194645  1205.98    0.000  23.4356442  23.5119438

                    Categories    -  Redundant    =  Num. Coefs
-----------------  ------------  --------------  ---------------  --
importer#year              1383            1383                0  *
exporter#year              1637            1637                0  *
exporter#importer          3710            3710                0  *
? = number of redundant parameters may be higher
* = FE nested within cluster; treated as redundant for DoF computation
```

Stata command:

```
ppmlhdfe trade_value log_distance agree_pta common_language
  contiguity, absorb(exp_ind#year imp_ind#year exp_ind#imp_ind)
  vce(cluster exp_ind imp_ind year)
```

Output:

```
HDFE PPML regression                              No. of obs      =     82,729
Absorbing 3 HDFE groups                           Residual df     =         26
Statistics robust to heteroskedasticity           Wald chi2(2)    =     301.21
Deviance             =  7.11106e+12               Prob > chi2     =     0.0000
Log pseudolikelihood = -3.55553e+12               Pseudo R2       =     0.9908

Number of clusters (imp_ind)=         61
Number of clusters (exp_ind)=         62
Number of clusters (year)   =         27
                     (Std. err. adjusted for 27 clusters in imp_ind exp_ind year)
------------------------------------------------------------------------------
               |               Robust
    trade_value | Coefficient  std. err.      z    P>|z|     [95% conf. interval]
----------------+-------------------------------------------------------------
   log_distance |          0  (omitted)
      agree_pta |  -.0321458   .0467761    -0.69   0.492    -.1238253    .0595337
common_language |          0  (omitted)
     contiguity |   .6819601   .0476734    14.30   0.000      .588522    .7753982
          _cons |    23.4738   .0194644  1205.99   0.000     23.43565    23.51194
------------------------------------------------------------------------------

Absorbed degrees of freedom:
-----------------------------------------------------------+
       Absorbed FE | Categories  - Redundant  = Num. Coefs |
-------------------+---------------------------------------|
      imp_ind#year |      1383        1383           0    *|
      exp_ind#year |      1637        1637           0    *|
   exp_ind#imp_ind |      3710        3710           0    *|
-----------------------------------------------------------+
* = FE nested within cluster; treated as redundant for DoF computation
```

C.3 ITPD-E: Linear Model

Python command:

```
reghdfe(['ln_trade'],
    ['rta','contiguity'],
    fixedefffects = ['exp_ind#industry_id#year',
                     'imp_ind#industry_id#year',
                     'exp_ind#imp_ind#industry_id'],
    data = self._df, setype='cluster',
    clustvars=['exp_ind','imp_ind', 'industry_id','year'])
```

Output:

```
================================================================================
HDFE Linear regression                            Number of obs   =   31,044,619
Absorbing 3 HDFE groups                               F(   2,     36) =     2.66
Statistics robust to heteroskedasticity                  Prob > F    =   0.0833
                                                       R-squared     =   0.8191
Number of clusters ( exp_ind) =      249              Adj R-squared   =   0.7926
Number of clusters ( imp_ind) =      250                Within R-sq.  =   0.0000
Number of clusters (industry_id) =      170              Root MSE    =   1.6771
Number of clusters (    year) =       37
        (Std. err. adjusted for 37 clusters in exp_ind imp_ind industry_id year)

                                    Robust
             Coefficient    Std. Err.         t    P>|z|    [95% conf. interval]
----------  -------------  -----------  --------  -------  ----------------------
       rta      0.0125774    0.0278550      0.45    0.654  -0.0439152   0.0690700
contiguity      0.3449093    0.1519794      2.27    0.029   0.0366809   0.6531377
     _cons     -2.8409732    0.0130594   -217.54    0.000  -2.8674590  -2.8144875

                              Categories    -  Redundant    =  Num. Coefs
---------------------------  ------------  --------------  ---------------  --
exp_ind#industry_id#year           799948          799948                0  *
imp_ind#industry_id#year          1020577         1020577                0  *
exp_ind#imp_ind#industry_id       2149222         2149222                0  *
* = FE nested within cluster; treated as redundant for DoF computation


runtime Δt=68.259192943573
```

Stata command:

```
reghdfe ln_trade rta contiguity, absorb(exp_ind#industry_id#year
    imp_ind#industry_id#year exp_ind#imp_ind#industry_id)
    vce(cluster exp_ind imp_ind industry_id year)
```

Output:

```
(dropped 802835 singleton observations)
(MWFE estimator converged in 54 iterations)

HDFE Linear regression                            Number of obs   = 31,044,619
Absorbing 3 HDFE groups                           F(   2,     36) =       2.66
Statistics robust to heteroskedasticity           Prob > F        =     0.0833
                                                  R-squared       =     0.8191
Number of clusters (exp_ind) =        249         Adj R-squared   =     0.7926
Number of clusters (imp_ind) =        250         Within R-sq.    =     0.0000
Number of clusters (industry_id) =        170     Root MSE        =     1.6771
Number of clusters (year)    =         37

      (Std. err. adjusted for 37 clusters in exp_ind imp_ind industry_id year)
------------------------------------------------------------------------------
             |               Robust
    ln_trade | Coefficient  std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
         rta |   .0125774    .027855     0.45   0.654    -.0439152      .06907
  contiguity |   .3449093   .1519794     2.27   0.029     .0366809     .6531377
       _cons |  -2.840973   .0130594  -217.54   0.000    -2.867459   -2.814487
------------------------------------------------------------------------------

Absorbed degrees of freedom:
-----------------------------------------------------------------+
                  Absorbed FE | Categories  - Redundant  = Num. Coefs |
------------------------------+----------------------------------------|
       exp_ind#industry_id#year |    799948      799948           0    *|
       imp_ind#industry_id#year |   1020577     1020577           0    *|
    exp_ind#imp_ind#industry_id |   2149222     2149222           0    *|
-----------------------------------------------------------------+
* = FE nested within cluster; treated as redundant for DoF computation
r; t=510.22 21:41:33
```

C.4 ITPD-E: PPML

Python command:

```
ppmlhdfe(['trade'],['rta','contiguity'],
    fixedeffects = ['exp_ind#industry_id#year',
                    'imp_ind#industry_id#year',
                    'exp_ind#imp_ind#industry_id'],
    data = self._df,  setype='cluster',
    clustvars=['exp_ind','imp_ind', 'industry_id','year'],
    separation=['fe'])
```

Output:

```
==============================================================================
HDFE PPML regression                              No. of obs    =   82,851,201
Absorbing 3 HDFE groups                           Residual df   =       36
                                                  Wald chi2( 2) =    22.17
Deviance       =  1.07699e+08                     Prob > chi2   =   0.0000
Log pseudolikelihood = -7.31427e+07               Pseudo R2     =   0.9949
Number of clusters ( exp_ind)=      251
Number of clusters ( imp_ind)=      251
Number of clusters (industry_id)=      170
Number of clusters (    year)=       37
        (Std. err. adjusted for 37 clusters in exp_ind imp_ind industry_id year)
------------------------------------------------------------------------------
                             Robust
             Coefficient    Std. Err.        z    P>|z|    [95% conf. interval]
----------  -------------  -----------  -------  -------  ----------------------
       rta      0.1005611    0.0362445     2.77    0.006   0.0295231   0.1715990
contiguity     -0.3233573    0.0698125    -4.63    0.000  -0.4601873  -0.1865274
     _cons     10.6765045    0.0043751  2440.30    0.000  10.6679295  10.6850795

                             Categories    -  Redundant    =  Num. Coefs
--------------------------  ------------  --------------  ---------------  --
exp_ind#industry_id#year           954051          954051                0  *
imp_ind#industry_id#year          1111815         1111815                0  *
exp_ind#imp_ind#industry_id       2761466         2761466                0  *
? = number of redundant parameters may be higher
* = FE nested within cluster; treated as redundant for DoF computation
runtime Δt=1442.8680713176727
```

Stata command:

```
ppmlhdfe trade rta contiguity, absorb(exp_ind#industry_id#year
  imp_ind#industry_id#year exp_ind#imp_ind#industry_id)
  vce(cluster exp_ind imp_ind industry_id year) separation(fe)
```

Output:

```
HDFE PPML regression                                  No. of obs      =   82851201
Absorbing 3 HDFE groups                               Residual df     =         36
Statistics robust to heteroskedasticity               Wald chi2(2)    =      22.17
Deviance             =  107699037.4                   Prob > chi2     =     0.0000
Log pseudolikelihood = -73142533.96                   Pseudo R2       =     0.9949

Number of clusters (exp_ind)=        251
Number of clusters (imp_ind)=        251
Number of clusters (industry_id)=       170
Number of clusters (year)   =         37
      (Std. err. adjusted for 37 clusters in exp_ind imp_ind industry_id year)
------------------------------------------------------------------------------
             |               Robust
       trade | Coefficient  std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
         rta |   .1005609   .0362446     2.77   0.006     .0295227     .171599
  contiguity |  -.3233568   .0698111    -4.63   0.000     -.460184   -.1865296
       _cons |   10.67651    .004375  2440.32   0.000     10.66793    10.68508
------------------------------------------------------------------------------

Absorbed degrees of freedom:
-----------------------------------------------------------------+
                     Absorbed FE | Categories  - Redundant  = Num. Coefs |
---------------------------------+-------------------------------------|
        exp_ind#industry_id#year |    954051      954051           0    *|
        imp_ind#industry_id#year |   1111815     1111815           0    *|
     exp_ind#imp_ind#industry_id |   2761466     2761466           0    *|
-----------------------------------------------------------------+
* = FE nested within cluster; treated as redundant for DoF computation
r; t=63285.77 15:16:19
```

References

Anderson JE, van Wincoop E (2003) Gravity with gravitas: A solution to the border puzzle. American Economic Review 93(1):170–192

Baltagi BH (2013) Econometric Analysis of Panel Data, 5th edn. John Wiley & Sons Inc., URL https://www.ebook.de/de/product/21225775/badi_h_baltagi_econometric_analysis_of_panel_data.html

Bergé L (2018) Efficient estimation of maximum likelihood models with multiple fixed-effects: The r package fenmlm. CREA Discussion Paper Series available for download at https://EconPapersrepecorg/RePEc:luc:wpaper:18-13 URL https://EconPapers.repec.org/RePEc:luc:wpaper:18-13

Cameron A, Gelbach J, Miller D (2011) Robust inference with multiway clustering. Journal of Business & Economic Statistics 29(2):238–249

Correia S (2015) Singletons, cluster-robust standard errors and fixed effects: A bad mix. unpublished manuscript available at http://scorreiacom/research/singletonspdf

Correia S (2016) A feasible estimator for linear models with multi-way fixed effects. unpublished manuscript available at http://scorreiacom/research/hdfepdf

Correia S, Guimarães P, Zylkin T (2020) Fast poisson estimation with high-dimensional fixed effects. The Stata Journal 20(1):95–115. https://doi.org/10.1177/1536867x20909691

Correia S, Guimarães P, Zylkin T (2026) Verifying the existence of maximum likelihood estimates for generalized linear models. unpublished manuscript available at https://arxivorg/pdf/190301633pdf

Egger P, Larch M (2008) Interdependent preferential trade agreement memberships: An empirical analysis. Journal of International Economics 76(2):384–399

Egger P, Tarlea F (2015) Multi-way clustering estimation of standard errors in gravity models. Economics Letters 134:144–147

Eicker F (1963) Asymptotic normality and consistency of the least squares estimators for families of linear regressions. The Annals of Mathematical Statistics 34(2):447–456. URL http://www.jstor.org/stable/2238390

Figueiredo O, Guimarães P, Woodward D (2015) Industry localization, distance decay, and knowledge spillovers: Following the patent paper trail. Journal of Urban Economics 89:21–31. https://doi.org/10.1016/j.jue.2015.06.003

Fong DCL, Saunders M (2011) LSMR: An iterative algorithm for sparse least-squares problems. SIAM Journal on Scientific Computing 33(5):2950–2971. https://doi.org/10.1137/10079687x

Frisch R, Waugh FV (1933) Partial time regressions as compared with individual trends. Econometrica 1(4):387. https://doi.org/10.2307/1907330

Gaure S (2013) OLS with multiple high dimensional category variables. Computational Statistics & Data Analysis 66:8–18. https://doi.org/10.1016/j.csda.2013.03.024

Greene WH (2020) Econometric Analysis, Global Edition, 8th edn. Pearson, URL https://www.ebook.de/de/product/32624527/william_h_greene_econometric_analysis_global_edition.html

Guimarães P, Portugal P (2010) A simple feasible procedure to fit models with high-dimensional fixed effects. Stata Journal 10(4):628–649(22). URL http://www.stata-journal.com/article.html?article=st0212

Halperin I (1962) The product of projection operators. Acta Scientiarum Mathematicarum (Szeged) 1-2(23):96–99

Hernández-Ramos LM, Escalante R, Raydan M (2011) Unconstrained optimization techniques for the acceleration of alternating projection methods. Numerical Functional Analysis and Optimization 32(10):1041–1066. https://doi.org/10.1080/01630563.2011.591954

Larch M, Wanner J, Yotov YV, et al (2019) Currency unions and trade: A PPML re-assessment with high-dimensional fixed effects. Oxford Bulletin of Economics and Statistics 81(3):487–510. https://doi.org/10.1111/obes.12283

Lovell MC (1963) Seasonal adjustment of economic time series and multiple regression analysis. Journal of the American Statistical Association 58(304):993–1010. https://doi.org/10.1080/01621459.1963.10480682

Mullahy J (1997) Instrumental-variable estimation of count data models: Applications to models of cigarette smoking behavior. Review of Economics and Statistics 79(4):586–593. https://doi.org/10.1162/003465397557169

Santos Silva J, Tenreyro S (2006) The log of gravity. Review of Economics and Statistics 88(4):641–658

Santos Silva J, Tenreyro S (2010) On the existence of the maximum likelihood estimates in poisson regression. Economics Letters 107(2):310–312

Somaini P, Wolak FA (2016) An algorithm to estimate the two-way fixed effects model. Journal of Econometric Methods 5(1). https://doi.org/10.1515/jem-2014-0008

Stammann A (2017) Fast and feasible estimation of generalized linear models with high-dimensional k-way fixed effects. unpublished manuscript available at

https://arxivorg/abs/170701815 arXiv:1707.01815 [stat.AP]

von Neumann J (1951) Functional Operators (AM-22), Volume 2. Princeton University Press, URL https://www.ebook.de/de/product/3678862/john_von_neumann_functional_operators_am_22_volume_2.html

White H (1980) A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity. Econometrica 48(4):817–838. URL http://www.jstor.org/stable/1912934

Windmeijer F, Santos Silva J (1997) Endogeneity in count data models: An application to demand for health care. Journal of Applied Econometrics 12(3):281–294. https://doi.org/10.1002/(sici)1099-1255(199705)12:3⟨281::aid-jae436⟩3.0.co;2-1

Wooldridge J (1992) Some alternatives to the box-cox regression model. International Economic Review 33(4):935–955. https://doi.org/10.2307/2527151

Yotov YV, Piermartini R, Monteiro J, et al (2016) An Advanced Guide to Trade Policy Analysis: The Structural Gravity Model. United Nations and World Trade Organization, Geneva, Switzerland, available for download at https://unctad.org/publication/advanced-guide-trade-policy-analysis-structural-gravity-model -volume-2