2020

Fast Poisson Estimation with High-Dimensional Fixed Effects

Sergio Correia

Paulo Guimarães

Thomas Zylkin
University of Richmond, tzylkin@richmond.edu

Recommended Citation

Correia, Sergio, Paulo Guimarães, and Tom Zylkin. "Fast Poisson estimation with high-dimensional fixed effects." *The Stata Journal* 20, no. 1 (2020): 95-115. https://doi-org.newman.richmond.edu/10.1177/1536867X20909691.

PPMLHDFE: Fast Poisson Estimation with High-Dimensional Fixed Effects

Sergio Correia[1], Paulo Guimarães[2], and Tom Zylkin[3]

[1]Federal Reserve Board, sergio.a.correia@frb.gov
[2]Banco de Portugal, pfguimaraes@bportugal.pt
[3]University of Richmond, tzylkin@richmond.edu

August 5, 2019

Abstract

In this paper we present `ppmlhdfe`, a new Stata command for estimation of (pseudo) Poisson regression models with multiple high-dimensional fixed effects (HDFE). Estimation is implemented using a modified version of the iteratively reweighted least-squares (IRLS) algorithm that allows for fast estimation in the presence of HDFE. Because the code is built around the `reghdfe` package, it has similar syntax, supports many of the same functionalities, and benefits from `reghdfe`'s fast convergence properties for computing high-dimensional least squares problems. Performance is further enhanced by some new techniques we introduce for accelerating HDFE-IRLS estimation specifically. `ppmlhdfe` also implements a novel and more robust approach to check for the existence of (pseudo) maximum likelihood estimates.

Keywords: `ppmlhdfe`, `reghdfe`, Poisson regression, high-dimensional fixed-effects

1 Introduction

Poisson regression is now well established as the standard approach to model count data. However, it is also gaining popularity as a viable alternative for estimation of multiplicative models where the dependent variable is nonnegative. Commonly, these models are estimated by linear regression applied to a log-transformed dependent variable. But, as with ordinary least squares (OLS), the only assumption required for consistency of the Poisson regression estimator is the correct specification of the conditional mean of the dependent variable (Gourieroux et al., 1984). In this setting, Poisson regression becomes Poisson pseudo maximum likelihood (PPML) regression. Gourieroux et al.'s results greatly extend the realm of application of Poisson regression because there is no

need to specify a distributional assumption for the dependent variable and, therefore, application is no longer restricted to count data. This means that PPML can be applied to any dependent variable with nonnegative values without the need to explicitly specify a distribution for the dependent variable. Moreover, unlike the log-linear model, PPML regression provides a natural way to deal with zero values on the dependent variable. Yet another advantage of PPML regression versus log-linear regression is that in the presence of heteroskedasticity, the parameters of log-linearized models estimated by OLS are inconsistent (Santos Silva and Tenreyro, 2006). In this context, the use of robust standard errors to mitigate concerns about heteroskedaticity will lead to incorrect inference because OLS estimators are not consistent in the first place.

The potential of PPML regression was recognized early in the spatial sciences by Davies and Guy (1987), who recommended using pseudo–likelihood methods instead of the more popular Poisson regression for the modeling of spatial flows. However, it was not until Santos Silva and Tenreyro (2006) that PPML really took off, particularly in the international trade literature. In that paper, the authors made an excellent case for the PPML model and posited it as the ideal estimator for gravity equations. At around the same time, Blackburn (2007) questioned the use of the traditional OLS approach for estimation of the Mincerian wage regression and proposed the use of pseudo-maximum likelihood estimators such as PPML regression. His basic point was essentially the same—labor economists routinely estimate wage regressions on micro datasets using log-linear regression, disregarding the fact that heteroskedasticity may undermine the validity of the results. A similar critique has also taken hold in the health economics literature, where the usage of log-linear regression to model health-care expenditures and utilization has been questioned (for example, Manning and Mullahy, 2001). Here, the more obvious reason for the adoption of PPML is the inadequacy of the log-transformation to deal with the large number of zeros typical in these areas.

In sum, in the presence of nonnegative data with possibly many zeros, if one wants to make minimal assumptions about the distribution of the data, then PPML seems like the safest bet. This situation is very likely to occur across many areas of research, particularly when working with highly granular data (for example, when modeling firm R&D expenditures, patent citation counts, daily product store sales, number of doctor visits, firm credit volumes, number of auction bidders, and number of commuters across regions).

Nevertheless, in applied work many researchers still resort to log-linear regressions in contexts where PPML would be better justified. One possible explanation is the ease with which researchers can estimate linear regressions that control for multiple fixed effects. The increasing availability of larger panel-type datasets, coupled with advances in estimation techniques for linear regression models with high-dimensional fixed effects (HDFE) has allowed researchers to control for multiple sources of heterogeneity. Stata users are familiar with the user-written package `reghdfe`, programmed by one of the authors, which has become Stata's standard tool for estimating linear models with multiple HDFE.

In this paper we show that PPML with HDFE can be implemented with almost

the same ease as linear regression with HDFE. To this end, we present `ppmlhdfe`, a new Stata command for fast estimation of Poisson regression models with HDFE. The `ppmlhdfe` command is to Poisson regression what `reghdfe` represents for linear regression in the Stata world—a fast and reliable command with support for multiple fixed effects. Moreover, `ppmlhdfe` takes great care to verify the existence of a maximum likelihood solution, adapting the innovations and suggested approaches described in Correia et al. (2019). It also introduces some novel acceleration techniques relative to existing algorithms for HDFE nonlinear estimation that eliminate some unnecessary steps and lead to faster computation of the parameters of interest.

2 Stata Commands for Estimation of Models with HDFE

The Stata community has been particularly active in developing and implementing methods to handle regression models that include more than one HDFE. The first such command, `a2reg`, was coded by Amine Ouazad and was made available in 2008. The program was basically a port of the FORTRAN code written by Robert Creecy for estimation of a linear regression model with two HDFE. The approach is detailed in Abowd et al. (2002) and involves solving the least-squares system of normal equations directly by application of the iterative conjugate gradient algorithm. The program provided the exact solution for the coefficients of the regression, but it lacked basic functionalities such as the calculation of the associated standard errors and data checks for multicollinearity. At around the same time, (Cornelissen, 2008) introduced the command `felsdvreg` that, like `a2reg`, was meant for estimation of a linear regression with two HDFE. Cornelissen used a clever decomposition of the design matrix to simplify estimation. His command was able to produce estimates of the standard errors, but his approach was only successful for particular data configurations and likely to break for larger datasets. A couple of years later, Guimarães and Portugal (2010) discussed an alternative algorithm for estimation of models with HDFE. Used in conjunction with the Frisch-Waugh-Lovell theorem (FWL), the algorithm could estimate these models using a minimum amount of memory and made easy the calculation of regular or one-way clustered standard errors. Following the publication of Guimarães and Portugal (2010), Johannes Schmieder made available the `gpreg` command while Guimarães produced the `reg2hdfe` command. Both commands used the general algorithm proposed in Guimarães and Portugal (2010) along with the FWL transformation. While `gpreg` was generally faster, `reg2hdfe` was able to handle larger data sets.

Later, Sergio Correia developed what is currently the state-of-the-art estimation command for linear regression models with HDFE. This command, `reghdfe`, offered several major improvements over existing commands. First, the convergence algorithm at the core of `reg2hdfe` was improved and written in Mata, making it faster, and with better convergence properties (Correia, 2016). Second, it supported multiple HDFE and their interactions, allowing for the full usage of factorial variable notation to control for the fixed effects. Other relevant improvements consisted of support for instrumental-

variables and different variance specifications, including multi-way clustering, support for weights, and the ability to use all post-estimation tools typical of official Stata commands such as `predict` and `margins`.[1] By all accounts `reghdfe` represents the current state-of-the-art command for estimation of linear regression models with HDFE, and the package has been very well accepted by the academic community.[2]

The fact that `reghdfe` offers a very fast and reliable way to estimate linear regression models with HDFE has opened up the way for estimation of other nonlinear regression models with HDFE. This is because many nonlinear models can be estimated by recursive application of linear regression. An obvious example is the nonlinear models that can be estimated by the nonlinear least-squares algorithm.[3] Another example is the iteratively reweighted least squares (IRLS) algorithm that was developed for estimation of generalized linear models (GLMs). This was the approach implemented by Guimarães in the user-written Stata command `poi2hdfe` developed for estimation of PPML regression with two HDFEs. The `poi2hdfe` command is basically a "wrapper" around `reghdfe` that implements estimation of a Poisson regression model with two HDFE. Because structural gravity applications often require PPML models with three sets of fixed effects, Tom Zylkin also made available to the Stata community a specialized command called `ppml_panel_sg`, which extended an earlier algorithm described in Figueiredo et al. (2015) and also built on the capabilities of `reghdfe` (Larch et al., 2019).

The command `ppmlhdfe` discussed in this paper implements PPML estimation with multiple HDFE, offering the full functionality of factorial variables to control for fixed effects. Thus, it can estimate the same models as `poi2hdfe` and `ppml_panel_sg` as well as more sophisticated models with multiple or interacted fixed effects, including models with heterogeneous slopes.[4]

To our knowledge, there are at present three other packages recently made available in R that also permit the estimation of Poisson regressions with multiple levels of fixed effects`alpaca` (Stammann, 2018), `FENmlm` (Bergé, 2018), and `glmhdfe` (Hinz et al., 2019). Of these, `alpaca` is the most similar to `ppmlhdfe` in that it combines a within-transformation step with a Newton-Raphson estimation algorithm roughly equivalent to the IRLS method used here. `FENmlm` also involves a combination of these two steps, but uses a nonlinear Gauss-Seidel method to update the fixed effects à la Figueiredo et al. (2015).[5] `glmhdfe`, meanwhile, does not use within-transformation and instead opts to

[1] For a complete set of features of the command see http://scorreia.com/software/reghdfe/.

[2] As of December 2018, it had more than 7,000 hits at SSC, making it the 14th most downloaded Stata package. Google Scholar shows more than 200 citations.

[3] For a discussion of this estimation method see Davidson and MacKinnon (1993).

[4] The command can be installed directly from the Statistical Software Components (SSC) archive. The development version, as well as complementary material, may be found at the dedicated Github repository: https://github.com/sergiocorreia/ppmlhdfe.

[5] To put the comparison another way, our method consists of within-transforming the data to take care of the fixed effects, using weighted least squares to update the remaining non-fixed effect coefficients (i.e., "β" in what follows), updating the weights, and repeating. In Bergé (2018), the within-transformation step is used to construct a concentrated Hessian for use in updating β. A version of the concentrated Hessian appears in the weighted least squares step used here; thus, while the two algorithms are presented in different ways, they still share much in common.

embed Gauss-Seidel updating within IRLS, similar to the Stata command `ppml_panel_sg`. One conceptual advantage our approach has over the former two methods is that we devise a way to solve the model without completely within-transforming the data from scratch in each iteration, thereby enabling us to realize significant speed gains. In addition, a general advantage of using a within-transformation approach over Gauss-Seidel is that it allows us to easily handle models with heterogenous slopes as noted above.

3 Estimation Approach

3.1 The iteratively reweighted least squares algorithm

GLMs are a class of regression models based on the exponential family of distributions that were introduced by Nelder and Wedderburn (1972). GLMs include popular nonlinear regression models such as logit, probit, cloglog, and Poisson. Following Ch. 12 of Hardin and Hilbe (2018), the exponential family is given by

$$f_y(y;\theta,\phi) = exp\left\{\frac{y\theta - b(\theta)}{a(\phi)} + c(y,\phi)\right\}, \tag{1}$$

where a(.), b(.), and c(.), are specific functions and ϕ and θ are parameters. For these models,

$$E(y) = \mu = b'(\theta) \tag{2}$$

and

$$V(y) = b''(\theta)a(\phi). \tag{3}$$

Given a set of n independent observations, each indexed by i, we can relate the expected value to a set of covariates ($\mathbf{x}_i$) by means of a link function g(.). More specifically it is assumed that

$$E(y_i) = \mu_i = g^{-1}(\mathbf{x}_i\beta), \tag{4}$$

and the likelihood for the GLM may be written as

$$L(\theta,\phi;y_1,y_2,...,y_n) = \prod_{i=1}^{n} exp\left\{\frac{y_i\theta_i - b(\theta_i)}{a(\phi)} + c(y_i,\phi)\right\}. \tag{5}$$

Estimates for β are obtained by solving the first-order conditions for maximization of the (pseudo) likelihood. Application of the Gauss-Newton algorithm with the expected Hessian leads to the following updating equation:

$$\beta^{(r)} = \left(\mathbf{X}'\mathbf{W}^{(r-1)}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{W}^{(r-1)}\mathbf{z}^{(r-1)}, \tag{6}$$

where $\mathbf{X}$ is the design matrix of explanatory variables, $\mathbf{W}^{(r-1)}$ is a weighting matrix, $\mathbf{z}^{(r-1)}$ is a transformation of the dependent variable, and r is an index for iteration (for

details, see Hardin and Hilbe, 2018). Equation (6) makes it clear that the estimates of β are obtained by recursive application of weighted least squares. This approach is known as Iteratively Reweighted Least Squares, or IRLS.

3.2 The Poisson regression model

In the case of Poisson regression we have

$$E(y_i) = \mu_i = \exp(\mathbf{x}_i\beta) \tag{7}$$

and the regression weights to implement IRLS simplify to

$$\mathbf{W}^{(r-1)} = \text{diag}\left\{\exp(\mathbf{x}_i\beta^{(r-1)})\right\} \tag{8}$$

while the dependent variable for the intermediary regression becomes

$$z_i^{(r-1)} = \left\{\frac{y - \exp(\mathbf{x}_i\beta^{(r-1)})}{\exp(\mathbf{x}_i\beta^{(r-1)})} + \mathbf{x}_i\beta^{(r-1)}\right\}. \tag{9}$$

Implementation of the IRLS updating regression in equation (6) requires only computation of the vector of fitted values $\mathbf{x}_i\beta^{(r-1)}$ obtained in the previous iteration.

Dealing with HDFE

The difficulty of implementing IRLS in the presence of HDFE comes from the fact that $\mathbf{X}$ may contain a large number of fixed effects that render the direct calculation of $(\mathbf{X}'\mathbf{W}^{(r-1)}\mathbf{X})$ impractical, if not impossible. The solution is to use an alternative updating formula that estimates only the coefficients of the non-fixed effect covariates (say, δ), thus reducing the dimensionality of the problem. This is because equation (6) is a weighted linear regression and therefore we can rely on the FWL theorem to expurgate the fixed effects. This means that instead of equation (6), we can use the following updating equation:

$$\delta^{(r)} = \left(\widetilde{\mathbf{X}}'\mathbf{W}^{(r-1)}\widetilde{\mathbf{X}}\right)^{-1}\widetilde{\mathbf{X}}'\mathbf{W}^{(r-1)}\widetilde{\mathbf{z}}^{(r-1)}, \tag{10}$$

where $\widetilde{\mathbf{X}}$ and $\widetilde{\mathbf{z}}$ are weighted within-transformed versions of the main covariate matrix $\mathbf{X}$ and working dependent variable $\mathbf{z}$, respectively. Moreover, the FWL theorem also implies that the residuals computed from equation (10) are the same as those from equation (6). This observation has two very useful implications for our purposes. First, it implies we can perform the needed updates to $\mathbf{W}$ and $\mathbf{z}$ using

$$\mathbf{X}\beta^{(r)} = \mathbf{z}^{(r-1)} - \mathbf{e}^{(r)}, \tag{11}$$

where $\mathbf{e}^{(r)}$ is a vector collecting the residuals computed using (10). New values for $\mathbf{W}^{(r)}$ and $\mathbf{z}^{(r)}$ then directly follow from (8) and (9), as in the original IRLS loop.[6] Second, it also means that once $\delta^{(r)}$ converges to the correct estimate $\hat{\delta}$, the estimated variance-covariance matrix for the weighted least squares regression in (10) will be the correct variance-covariance matrix for $\hat{\delta}$, and standard adjustments for heteroskedasticity and clustering similarly require no further special steps.

Accelerating HDFE-IRLS

The user-written command `poi2hdfe` implemented the updating equation (10) using `reghdfe` as the workhorse for running the HDFE weighted least-squares regressions. This is a computationally intensive procedure requiring estimation of an HDFE regression model in every IRLS iteration. There are, however, several workarounds in `ppmlhdfe` that make it much more efficient. For instance, `ppmlhdfe` directly embeds the Mata routines of `reghdfe`, thus taking advantage of the fact that some of the computations need to be done only once, as they remain the same for every IRLS iteration. But the most significant speed improvements come from the modifications we have introduced to the standard HDFE-IRLS algorithm aimed at reducing the number of calls to `reghdfe`. These modifications are as follows:

First, we within-transform (or "partial out") the original untransformed variables $\mathbf{z}$ and $\mathbf{X}$ in the first IRLS iteration only. From the second iteration onwards, we exploit the fact that, given an arbitrary linear combination of the fixed effects $\mathbf{d}$, partialing out $\mathbf{z}-\mathbf{d}$ is numerically equivalent to partialing out $\mathbf{z}$. Hence, if the eventual solution to the partial-out step is $\widetilde{\mathbf{z}} = \mathbf{z}^{(r)} - \mathbf{d}^{(r)}$, it is often much faster to partial out $\mathbf{z}^{(r)} - \mathbf{d}^{(r-1)}$ than it is to start from the untransformed $\mathbf{z}$ variable (since $\mathbf{d}^{(r-1)}$ is generally a reasonable initial guess for $\mathbf{d}^{(r)}$). In practice, we progressively update $\widetilde{\mathbf{z}}$ by starting each within-transformation step using the new starting value $\widetilde{\mathbf{z}}^{*(r)} = \widetilde{\mathbf{z}}^{(r-1)} + \mathbf{z}^{(r)} - \mathbf{z}^{(r-1)}$, where $\mathbf{z}^{(r)}$ and $\mathbf{z}^{(r-1)}$ are computed as in (6).[7] We similarly are able to within-transform $\mathbf{X}$ in a progressive fashion by starting from the last values of $\widetilde{\mathbf{X}}$ in each iteration rather than starting over again from the original untransformed $\mathbf{X}$ variables.

Second, another artifact that helps speed up convergence involves the choice of the criterion for the inner loops of `reghdfe`. In our implementation, this criterion becomes tighter as we approach convergence, thus avoiding unnecessary `reghdfe` iterations. In sum, we are able to progressively within-transform both $\mathbf{X}$ and $\mathbf{z}$ while simultaneously updating the weights and $\mathbf{z}$ needed for the IRLS step, only requiring the within-transformation procedure to reach completion as the full algorithm converges.

[6] A similar principle is also employed in Stammann (2018). Her formulation of the weighted least squares step differs in that she differences out the $\mathbf{x}_i\beta^{(r-1)}$ term from the traditional IRLS dependent variable. This approach should nonetheless be roughly equivalent to IRLS computationally. Another difference between her algorithm and ours comes from the special acceleration techniques we have programmed into `ppmlhdfe`, as we discuss next.

[7] Note that, if $\mathbf{d}^{(r)} \approx \mathbf{d}^{(r-1)}$, then $\widetilde{\mathbf{z}}^{(r)} = \mathbf{z}^{(r)} - \mathbf{d}^{(r)} \approx \mathbf{z}^{(r-1)} - \mathbf{d}^{(r-1)} + \mathbf{z}^{(r)} - \mathbf{z}^{(r-1)} = \widetilde{\mathbf{z}}^{(r-1)} + \mathbf{z}^{(r)} - \mathbf{z}^{(r-1)}$. Moreover, since $\mathbf{X}$ is the same in every iteration it only needs to be partialled out in the initial iteration.

In practice, these innovations can reduce the total number of calls to `reghdfe` by 50% or more, leading to substantial speed gains in computation.[8]

Existence of MLE

Santos Silva and Tenreyro (2010) and Santos Silva and Tenreyro (2011) noted that for some data configurations, maximum likelihood estimates for the Poisson regression may not exist. As a result, estimation algorithms may be unable to converge or may converge to the incorrect estimates. This situation bears some resemblance to the well-known problem of separation in the binary-choice model. In the case of Poisson regression, this happens if the log-likelihood increases monotonically as one or more coefficients tends to infinity. As shown by Santos Silva and Tenreyro (2010), this may occur if there is multicollinearity among the regressors for the subsample of positive values of the dependent variable. To overcome this problem, they suggest identifying and dropping problematic regressors.[9] However, which regressor(s) to drop is an ambiguous decision with implications for the identification of the remaining parameters. Moreover, in Poisson models with multiple HDFEs this strategy may not even be feasible.

In a recent paper (Correia et al., 2019), we discuss the necessary and sufficient conditions for the existence of estimates in a wide class of GLM models and show that, in the case of Poisson regression, it is always possible to find MLE estimates if some observations are dropped from the sample. These observations—separated observations—do not convey relevant information for the estimation process and thus can be safely discarded. After dropping these observations, some regressors will become collinear and thus must also be dropped. Additionally, in the same paper, we propose a method to identify separated observations that will succeed even in the presence of HDFEs. By default, `ppmlhdfe` implements this method (the **ir** method) plus three other methods to identify separated observations.[10]

4 The `ppmlhdfe` Command

`ppmlhdfe` requires the installation of the latest versions of `ftools` and `reghdfe`.

4.1 Syntax

The syntax for `ppmlhdfe` is similar to that of `reghdfe`:

[8] For example, in the trade data example that follows, `ppmlhdfe` at its default settings reaches convergence after 36 calls to `reghdfe`. If we instead disable these features (by adding the options `start_inner_tol(1e-08)` and `use_exact_partial(1)`), the total number of calls increases to 98 (effectively an 170% increase in computing time). The exact same answer is achieved in either case.

[9] This approach is implemented in their user-written package `ppml`.

[10] For a better understanding of the methods implemented in `ppmlhdfe`, see our primer on separation available on `ppmlhdfe`'s Github repository.

`ppmlhdfe` *depvar* [*indepvars*] [*if*] [*in*] [*weight*] , [`absorb(`*absvars*`)`
`exposure(`*varname*`) offset(`*varname*`) d(`*varname*`) d vce(`*vcetype*`)`
`verbose(#) nolog(#) tolerance(#) guess(`*string*`) separation(`*string*`)`
`maxiteration(#) keepsingletons` *display_options*]

depvar is the dependent variable. It must be nonnegative but it is not restricted to integer values. Use of time-series operators or factor variables (if they specify one level of the group) is allowed.

indepvars represents the set of explanatory variables in the regression. Both factor and time series operators are allowed.

Options

`absorb(`*absvars* [, *savefe*]) *absvars* contains a list of all categorical variables that identify the fixed effects to be absorbed. Each variable represents one set of fixed effects. Factor variable notation can be used. If you want to save the estimates of the fixed effects, you can either assign a name to the new variable when specifying *absvars*, as in *newvar*=*absvar*, or you can use the option *savefe*, in which case all fixed effects estimates are saved using the stub *__hdfe#__*.

`exposure(`*varname*`)` includes ln(*varname*) in model with coefficient constrained to 1.

`offset(`*varname*`)` includes *varname* in model with coefficient constrained to 1.

`d(`*varname*`)` creates a new variable with the sum of the fixed effects. The option is required if you are absorbing fixed effects and planning on running `predict` afterwards.

`d` works as above but automatically names the variable *__ppmlhdfe__*.

`vce(`*vcetype*`)` where *vcetype* may be `robust` (default) or `cluster` *fvvarlist* (allowing two- and multi-way clustering).

`verbose(#)` controls the amount of debugging information to show. Default is 0 but higher integer values will present increasing detail. The value -1 will prevent the displaying of any messages.

`nolog(#)` hides the iteration output.

`tolerance(#)` criterion for convergence. Default is 1e-8.

`guess(`*string*`)` rule for setting initial values; valid options are `simple` (the default) and `ols`.

`separation(`*string*`)` set rules for dropping separating observations; valid options are `fe`, `ir`, `simplex`, and `mu` (or any combination of those). By default the first three are used (`fe simplex ir`). To disable all separation checks, set the option to `none`.

`maxiteration(#)` maximum number of iterations. The default is 10,000.

`version` reports the current version and installed dependencies. Should not be used

with any arguments.

keepsingletons does not drop singleton groups.

eform report exponentiated coefficients (incidence-rate ratios).

irr synonym for eform.

More specialized options can be found in the documentation available on the Github repository.

4.2 Saved results

ppmlhdfe saves the following results to e():

Scalars

e(N)	number of observations
e(num_singletons)	number of dropped singleton observations
e(num_separated)	number of separated observations
e(N_full)	number of observations, including dropped, singleton and separated obs.
e(drop_singletons)	1 if singleton obs. were searched for and dropped
e(rank)	rank of e(V)
e(df)	residual degrees of freedom
e(df_m)	model degrees of freedom
e(df_a)	degrees of freedom lost due to the fixed effects
e(df_a_initial)	number of categories in the fixed effects: e(df_a) - e(df_a_redundant)
e(df_a_redundant)	number of redundant fixed effect categories
e(N_hdfe)	number of absorbed fixed-effects
e(N_hdfe_extended)	number of absorbed fixed-effects plus fixed-slopes
e(rss)	residual sum of squares
e(rmse)	root mean squared error
e(chi2)	chi-squared
e(r2_p)	pseudo_R_squared
e(ll)	log-likelihood
e(ll_0)	log-likelihood of fixed-effect-only regression
e(N_clustervars)	number of clustervars
e(N_clust#)	number of clusters in the #th cluster variable
e(N_clust)	number of clusters; minimum of all the e(clust#)
e(ic)	number of iterations
e(ic2)	number of iterations when partialling out fixed effects

Macros

e(cmd)	ppmlhdfe
e(cmdline)	command as typed
e(separation)	list of methods used to detect and drop separated observations
e(dofmethod)	dofmethod employed in the regression
e(depvar)	name of dependent variable
e(indepvars)	name of independent variables

e(absvars)	name of the absorbed variables or interactions
e(extended_absvars)	expanded absorbed variables or interactions
e(title)	title in estimation output
e(clustvar)	name of cluster variable
e(clustvar#)	name of the #th cluster variable
e(vce)	*vcetype* specified in vce()
e(chi2type)	title used to label Std. Err.
e(offset)	linear offset variable
e(properties)	**b V**
e(predict)	ppmlhdfe_p; program used to implement predict
e(estat_cmd)	reghdfe_estat; program used to implement estat
e(marginsok)	predictions allowed by margins
e(marginsnotok)	predictions disallowed by margins
e(footnote)	reghdfe_footnote; program used to display the degrees-of-freedom table

Matrices

e(b)	coefficient vector
e(V)	variance-covariance matrix of the estimators
e(dof_table)	number of categories, redundant categories, and degrees-of-freedom absorbed by each set of fixed effects

Functions

e(sample)	marks estimation sample

5 Examples

We start out with a very simple example that shows the advantage of ppmlhdfe's approach for dealing with the nonexistence of MLE estimates. As explained earlier, ppmlhdfe takes great care to identify separated observations and then restricts the sample in a way that guarantees the existence of meaningful maximum-likelihood estimates. Our illustrative data consists of six observations and three explanatory variables:

```
input y x1 x2 x3
0 1 2 1
0 0 0 2
0 2 3 3
1 1 2 4
2 2 4 5
3 1 2 6
end
```

If we try to estimate a Poisson regression with the glm command, Stata fails to converge. The poisson command produces estimates for all three coefficients associated with the Xs, but a quick inspection of results makes it clear that the results are unreliable:[11]

```
. poisson y x1 x2 x3, nolog
Poisson regression                              Number of obs     =          6
                                                LR chi2(3)        =       8.89
                                                Prob > chi2       =     0.0308
```

[11] A similar situation occurs if we estimate the Poisson regression using glm with the *irls* option.

```
Log likelihood = -4.0415302                     Pseudo R2         =     0.5237
```

y	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
x1	-31.29521	8467.059	-0.00	0.997	-16626.43	16563.83
x2	15.84339	4233.529	0.00	0.997	-8281.722	8313.408
x3	.7970409	.4608654	1.73	0.084	-.1062388	1.70032
_cons	-4.032453	2.868066	-1.41	0.160	-9.653759	1.588853

The user-written command `ppml` identifies the existence of a data problem and drops the variable `x3`. However, the fitted regression still shows problems:

```
. ppml y x1 x2 x3
note: checking the existence of the estimates
Number of regressors excluded to ensure that the estimates exist: 1
Excluded regressors:  x3
Number of observations excluded: 0
note: starting ppml estimation
Iteration 1:   deviance =  5.984675
(output ommitted)
Iteration 15:  deviance =  5.489052
Warning:  variance matrix is nonsymmetric or highly singular
Number of parameters: 3
Number of observations: 6
Pseudo log-likelihood: -6.5473014
R-squared: .3399843
Option strict is: off
WARNING: The model appears to overfit some observations with y=0
```

y	Coef.	Robust Std. Err.	z	P>|z|	[95% Conf.	Interval]
x1	-32.31708	.	.	.	.	.
x2	16.58591	.	.	.	.	.
_cons	-.8167293	.	.	.	.	.

If ran with the *strict* option, `ppml` will simply drop all regressors. The approach of `ppmlhdfe` is quite different. Instead of searching for problematic regressors, it looks for problematic observations. In this case, `ppmlhdfe` drops the third observation. It then drops `x2` to avoid perfect multicollinearity. The results are more plausible:[12]

```
. ppmlhdfe y x1 x2 x3, nolog
(simplex method dropped 1 separated observation)
note: 1 variable omitted because of collinearity: x2
Converged in 6 iterations and 6 HDFE sub-iterations (tol = 1.0e-08)
PPML regression                                 No. of obs      =          5
                                                Residual df     =          2
```

[12] To our knowledge, besides `ppmlhdfe`, no package in any other statistical software is capable of dealing with the separation problem in a robust way. Please see our comparisons available at the Github repository. In the site, we also provide a set of *csv* files which contain examples of separation that package developers can use to verify the robustness of their code.

```
                                                Wald chi2(2)    =      50.78
Deviance               =  .4775093816           Prob > chi2     =     0.0000
Log pseudolikelihood = -4.041530113             Pseudo R2       =     0.4532
```

y	Coef.	Robust Std. Err.	z	P>|z|	[95% Conf.	Interval]
x1	.3914642	.1733026	2.26	0.024	.0517975	.731131
x2	0	(omitted)				
x3	.7969293	.1582404	5.04	0.000	.4867838	1.107075
_cons	-4.031679	1.119578	-3.60	0.000	-6.226012	-1.837347

Next we replicate Example 1 shown on page 359 of the Stata 15 manual for the command [XT] **xtpoisson** with the fixed effects option. This example uses the *ships* dataset and estimates a Poisson regression of the number of *ship* accidents on several regressors. It treats the variable *ship* as a fixed effect to control for five different types of ships. The regression is estimated with a control for exposure (*service*) and the coefficients are reported as incidence rate-ratios. The syntax needed to replicate the example is [13]

```
. webuse ships, clear
. xtpoisson acc op_75_79 co_65_69 co_70_74 co_75_79, exp(service) irr fe nolog
```

To obtain equivalent results with `ppmlhdfe`, we do

```
. ppmlhdfe acc op_75_79 co_65_69 co_70_74 co_75_79, a(ship) exp(service) irr nolog
Converged in 6 iterations and 6 HDFE sub-iterations (tol = 1.0e-08)
HDFE PPML regression                            No. of obs      =         34
Absorbing 1 HDFE group                          Residual df     =         25
                                                Wald chi2(4)    =     111.06
Deviance               =  38.69505154           Prob > chi2     =     0.0000
Log pseudolikelihood = -68.28077143             Pseudo R2       =     0.8083
```

accident	exp(b)	Robust Std. Err.	z	P>|z|	[95% Conf.	Interval]
op_75_79	1.468831	.1484359	3.80	0.000	1.204902	1.790572
co_65_69	2.008002	.2202475	6.36	0.000	1.619572	2.489592
co_70_74	2.26693	.3256501	5.70	0.000	1.710649	3.004107
co_75_79	1.573695	.3117262	2.29	0.022	1.067358	2.320232
_cons	.0011254	.0001061	-72.03	0.000	.0009356	.0013538
ln(service)	1	(exposure)				

```
Absorbed degrees of freedom:
```

Absorbed FE	Categories	- Redundant	= Num. Coefs
ship	5	0	5

where we are absorbing *ship* as a fixed effect. A few points are worth mentioning. As expected the estimated coefficients for the variables are the same as those obtained with

[13] Note that the variable *ship* is already set as the *panelvar* in the ships dataset.

the [XT] **xtpoisson** command. However, the results for the estimates of the standard errors are different because, by default, `ppmlhdfe` reports robust standard errors.[14] The values for the log-likelihoods presented by the two commands are also different. The command [XT] **xtpoisson** reports the value of the conditional log-likelihood, while `ppmlhdfe` reports the actual Poisson log-likelihood (and could thus possibly be used for likelihood ratio tests against a Poisson regression if one were willing to accept the Poisson distribution assumption). Given that we are working with a small dataset, we could replicate the results obtained with `ppmlhdfe` using the [R] **poisson** command as in

```
poisson acc op_75_79 co_65_69 co_70_74 co_75_79 i.ship, exp(service) irr vce(robust)
```

Note that we can absorb any categorical variable as a fixed effect. For example, if we were interested only in the coefficients for *op_75_79* and *co_75_79*, we could absorb *ship*, *co_70_74*, and *co_75_79* as

```
. ppmlhdfe acc op_75_79 co_65_69, a(ship co_70_74 co_75_79) exp(service) irr nolog
Converged in 7 iterations and 23 HDFE sub-iterations (tol = 1.0e-08)
HDFE PPML regression                              No. of obs      =         34
Absorbing 3 HDFE groups                           Residual df     =         25
                                                  Wald chi2(2)    =      71.60
Deviance             =  38.69505154               Prob > chi2     =     0.0000
Log pseudolikelihood = -68.28077143               Pseudo R2       =     0.8083
```

accident	exp(b)	Robust Std. Err.	z	P>|z|	[95% Conf.	Interval]
op_75_79	1.468831	.1484359	3.80	0.000	1.204902	1.790572
co_65_69	2.008002	.2202475	6.36	0.000	1.619572	2.489592
_cons	.0015435	.0001204	-83.00	0.000	.0013247	.0017984
ln(service)	1	(exposure)				

```
Absorbed degrees of freedom:
```

Absorbed FE	Categories	- Redundant	= Num. Coefs	
ship	5	0	5	
co_70_74	2	1	1	
co_75_79	2	1	1	?

```
? = number of redundant parameters may be higher
```

and the results would be exactly the same for the variables that were explicitly kept in the model.

Our third example provides a natural application of `ppmlhdfe`. Here we estimate a gravity model using the ancillary data and example provided with the `ppml_panel_sg` command. The dataset contains annual bilateral trade data for 35 countries from 1986 to 2004. The objective is to estimate the impact that *fta*—a free trade agreement variable—has on trade. In this example we want to control for country pair fixed effects and country time fixed effects (for both importer and exporter countries). Additionally,

[14] If we used the option vce(robust) in the [XT] **xtpoisson** command, the results would still be different. This is because in **xtpoisson** (and other xt commands), Stata replaces vce(robust) with vce(cluster *ships*), but does not apply a small-sample adjustment for the number of clusters (see [U]20.22).

we want our standard errors clustered at the level of the country-pair.

```
. use http://fmwww.bc.edu/RePEc/bocode/e/EXAMPLE_TRADE_FTA_DATA if category=="TOTAL", clear
(Example gravity data for ppml_panel_sg (35 countries, 1988-2004, every 4 yrs))
. cap egen imp=group(isoimp)
. cap egen exp=group(isoexp)
. ppmlhdfe trade fta, a(imp#year exp#year imp#exp) cluster(imp#exp) nolog
Converged in 11 iterations and 36 HDFE sub-iterations (tol = 1.0e-08)
HDFE PPML regression                              No. of obs      =      5,950
Absorbing 3 HDFE groups                           Residual df     =      1,189
Statistics robust to heteroskedasticity           Wald chi2(1)    =      21.04
Deviance             =  377332502.3               Prob > chi2     =     0.0000
Log pseudolikelihood = -188710931.7               Pseudo R2       =     0.9938
Number of clusters (imp#exp)=      1,190
                          (Std. Err. adjusted for 1,190 clusters in imp#exp)
```

trade	Coef.	Robust Std. Err.	z	P>|z|	[95% Conf.	Interval]
fta	.1924455	.0419527	4.59	0.000	.1102197	.2746713
_cons	16.45706	.0217308	757.32	0.000	16.41447	16.49965

Absorbed degrees of freedom:

Absorbed FE	Categories	- Redundant	= Num. Coefs	
imp#year	175	0	175	
exp#year	175	5	170	
imp#exp	1190	1190	0	*

* = FE nested within cluster; treated as redundant for DoF computation

Notice that we have used Stata factorial notation when specifying the three variables to be absorbed, which is generally faster than creating the categorical variables for the interactions beforehand.

Because our trade example includes high-dimensional fixed effects, it also gives us an opportunity to unpack the key mechanisms behind how `ppmlhdfe` works from a programming perspective. `ppmlhdfe` itself is a complex program that is mainly coded in Mata and also takes advantage of the existing inner workings of `reghdfe` whereever possible. But the essential algorithm used to implement HDFE-IRLS is quite portable and can be programmed in any language that already offers robust algorithms for within-transformation and standard weighted regression. The following simple Stata code helps to illustrate[15].

```
use http://fmwww.bc.edu/RePEc/bocode/e/EXAMPLE_TRADE_FTA_DATA if category=="TOTAL", clear
egen imp = group(isoimp)
egen exp = group(isoexp)
egen pair = group(isoexp isoimp)
local accelerate = 1
local crit 1
local iter 0
```

[15]This code is available at the Github repository.

```
local last .
local inner_tol = 1e-4
while (`crit´ > 1e-8 | `inner_tol´ > `crit´) {
    loc ++iter
    di as text _n "Iteration `iter´ (crit=`crit´) (HDFE tol=`inner_tol´)"
    if (`iter´==1) {
        qui su trade, mean
        qui gen double mu = (trade + r(mean)) / 2
        qui gen double eta = log(mu)
        qui gen double z = eta + trade / mu - 1
        qui gen double last_z = z
        qui gen double reg_z = z
    }
    else if (`accelerate´) {
        qui replace last_z = z
        qui replace z = eta + trade / mu - 1
        qui replace reg_z = z - last_z + z_resid
        qui replace fta = fta_resid
    }
    else {
        qui replace z = eta + trade / mu - 1
        qui replace reg_z = z
    }
    * Tighten HDFE tolerance
    if (`crit´ < 10 * `inner_tol´) {
        local inner_tol = `inner_tol´ / 10
    }
    cap drop *resid
    * Perform HDFE Weighted Least Squares
    qui reghdfe reg_z [aw=mu], a(imp#year exp#year imp#exp) ///
    res(z_resid) keepsing v(-1) tol(`inner_tol´)
    qui reghdfe fta [aw=mu], a(imp#year exp#year imp#exp) ///
    res(fta_resid) keepsing v(-1) tol(`inner_tol´)
    reg z_resid fta_resid [aw=mu], noconstant cluster(pair) nohead
    predict double resid, resid

    * Update eta = Xb; mu = exp(eta)
    qui replace eta = z - resid
    qui replace mu = exp(eta)
    local crit = abs(_b[fta_resid] - `last´)
    local last = _b[fta_resid]
 }
```

As the while loop at the center of this example code converges, the final point estimate and standard error for *fta* will be the same as those computed by `ppmlhdfe` above. Some key operations to point out are the crucial IRLS updating step—`replace eta = z - resid`—and the acceleration step that allows us to avoid having to within-transform `z` from scratch in each iteration—`replace z = z_resid + z - z_last`. The `reghdfe` command, when used without any righthand-side covariates, may be used to perform each within-transformation step. Also notice that we progressively tighten the `reghdfe` tolerance when we approach convergence; as we have noted, full within-transformation is only required for the final set of estimates.[16]

[16] For simplicity, this example code uses convergence of the estimated coefficient for *fta* as the stopping criterion. However, in the complete algorithm we require full convergence of the deviance, as in standard IRLS implementations.

Our final example showcases how to combine `ppmlhdfe` with the `esttab` package to construct publication-quality regression tables. To construct the labels for the fixed effects, we use the `estfe` command included with `reghdfe`. The example consists of a database of more than a million observations, containing all SSCI citations collected between 1991 and 2009 for 100,404 articles published in 170 economics journals between 1991 and 2006.[17] The database has an ID for journal, article, the number of authors, the two-digit main JEL classification code[18] (124 codes), the publication year, the type of article (proceeding, journal article, review, or note) and the number of citations collected in each year. Nearly 58 percent of all observations are zeros. Our dependent variable is the number of citations, and we will pretend that our main interest is understanding whether the number of authors in an article does in fact increase the number of citations once we control for a variety of controls. We run the following specifications:

```
. use citations_example, clear
. estimates clear
. ppmlhdfe cit nbaut, a(issn type jel2 pubyear)
. eststo
. ppmlhdfe cit nbaut, a(issn#c.year type jel2 pubyear)
. eststo
. ppmlhdfe cit nbaut, a(issn#year type jel2 pubyear)
. eststo
. estfe *, labels(type "Article type FE" jel2 "JEL code FE" pubyear "Publication year FE" ///
issn "ISSN FE" issn#c.year "Year trend by ISSN" issn#year "ISSN-Year FE")
```

The results are summarized in the following table produced by the `esttab` command. They show that the number of authors has a very robust impact on the citation count.

```
. esttab, indicate(`r(indicate_fe)´, labels("Yes" "")) b(3) se(3) varwidth(25) label ///
>         stat(N ll, fmt(%12.0fc %13.1fc)) se starlevels(* 0.1 ** 0.05 *** 0.01) compress
```

	(1) cit	(2) cit	(3) cit
nbaut	0.190***	0.190***	0.189***
	(0.003)	(0.003)	(0.003)
Constant	0.054***	0.081***	0.110***
	(0.006)	(0.006)	(0.006)
Article type FE	Yes	Yes	Yes
JEL code FE	Yes	Yes	Yes
Publication year FE	Yes	Yes	Yes
ISSN FE	Yes		
Year trend by ISSN		Yes	
ISSN-Year FE			Yes
N	1,083,701	1,083,701	1,080,051
ll	-1,714,495.9	-1,685,149.1	-1,655,106.1

[17] For a detailed description of the dataset see Cardoso et al. (2010). The dataset can be downloaded from the Github package repository.

[18] The JEL code is a detailed system of classification for articles in economics.

```
Standard errors in parentheses
* p<0.1, ** p<0.05, *** p<0.01
```

What are the exact differences between the regressions? In all three regressions we introduced a fixed effect for the type of article, the JEL code, and the publication year. In the first specification we treat the journal as any other fixed effect. However, in the second specification we are assuming that there is a trend in the number of citations that is specific to each journal. Finally, the last specification introduces even more flexibility and permits the existence of a year-specific impact for each journal. While this is simply an illustrative example of the capabilities of `ppmlhdfe`, if taken with due care the analysis of the estimates of the absorbed effects may also reveal interesting information.

6 Conclusion

`ppmlhdfe` is a new user-written command that allows for fast estimation of (pseudo) Poisson regression models. It has a similar syntax to `reghdfe` and many of the same functionalities. Moreover, `ppmlhdfe` takes great care to check for existence of maximum likelihood results and introduces some promising new concepts for accelerating nonlinear estimation with high-dimensional covariates. Finally, we note that the estimation approach of `ppmlhdfe` could easily be extended to any other model from the GLM family.

References

Abowd, J., R. Creecy, and F. Kramarz. 2002. Computing Person and Firm Effects Using Linked Longitudinal Employer–Employee Data. Technical Report 2002-06, U.S. Census Bureau. URL `http://lehd.dsd.census.gov/led/library/techpapers/tp-2002-06.pdf`.

Bergé, L. 2018. Efficient Estimation of Maximum Likelihood Models with Multiple Fixed-Effects: The R Package FENmlm. CREA Discussion Paper Series, Center for Research in Economic Analysis, University of Luxembourg. URL `https://EconPapers.repec.org/RePEc:luc:wpaper:18-13`.

Blackburn, M. L. 2007. Estimating Wage Differentials without Logarithms. *Labour Economics* 14(1): 73–98.

Cardoso, A. R., P. Guimarães, and K. F. Zimmermann. 2010. Trends in Economic Research: An International Perspective. *Kyklos* 63(4): 479–494.

Cornelissen, T. 2008. The Stata Command felsdvreg to Fit a Linear Model with Two High-Dimensional Fixed Effects. *Stata Journal* 8(2): 170–189.

Correia, S. 2016. reghdfe: Estimating Linear Models with Multi-Way Fixed Effects. 2016 Stata conference, Stata Users Group.

Correia, S., P. Guimarães, and T. Zylkin. 2019. Verifying the Existence of Maximum Likelihood Estimates for Generalized Linear Models. Technical report, University of Richmond.

Davidson, R., and J. MacKinnon. 1993. *Estimation and Inference in Econometrics*. NY: Oxford University Press.

Davies, R. B., and C. M. Guy. 1987. The Statistical Modeling of Flow Data when the Poisson Assumption is Violated. *Geographical Analysis* 19(4): 300–314.

Figueiredo, O., P. Guimarães, and D. Woodward. 2015. Industry Localization, Distance Decay, and Knowledge Spillovers: Following the Patent Paper Trail. *Journal of Urban Economics* 89(C): 21–31.

Gourieroux, C., A. Monfort, and A. Trognon. 1984. Pseudo Maximum Likelihood Methods: Theory. *Econometrica* 52(3): 681–700.

Guimarães, P., and P. Portugal. 2010. A Simple Feasible Procedure to Fit Models with High-Dimensional Fixed Effects. *Stata Journal* 10(4): 628–649.

Hardin, J. W., and J. Hilbe. 2018. *Generalized Linear Models and Extensions, Fourth Edition*. College Station, Texas: Stata Press.

Hinz, J., A. Hudlet, and J. Wanner. 2019. Separating the Wheat from the Chaff: Fast Estimation of GLMs with High-Dimensional Fixed Effects. Retrieved from https:/github.com/julianhinz/R_glmhdfe.

Larch, M., J. Wanner, Y. V. Yotov, and T. Zylkin. 2019. Currency Unions and Trade: A PPML Re-assessment with High-dimensional Fixed Effects. *Oxford Bulletin of Economics and Statistics (forthcoming)* . URL `https://onlinelibrary.wiley.com/doi/abs/10.1111/obes.12283.`

Manning, W. G., and J. Mullahy. 2001. Estimating Log Models: to Transform or not to Transform? *Journal of Health Economics* 20(4): 461–494.

Nelder, J., and R. Wedderburn. 1972. Generalized Linear Models. *Journal of the Royal Statistical Society, Series A* 135: 370–384.

Santos Silva, J. M. C., and S. Tenreyro. 2006. The Log of Gravity. *The Review of Economics and Statistics* 88(4): 641–658.

———. 2010. On the Existence of the Maximum Likelihood Estimates in Poisson Regression. *Economics Letters* 107(2): 310–312.

———. 2011. Poisson: Some Convergence Issues. *Stata Journal* 11(2): 215–225.

Stammann, A. 2018. Fast and Feasible Estimation of Generalized Linear Models with High-Dimensional k-way Fixed Effects. *ArXiv e-prints* . URL `https://arxiv.org/pdf/1707.01815v2.pdf.`