High Dimensional Metrics in Julia

VICTOR CHERNOZHUKOV, CHRISTIAN HANSEN, MARTIN SPINDLE

October - 13 - 2022

Contents

1 Introduction

Analysis of high-dimensional models, models in which the number of parameters to be estimated is large relative to the sample size, is becoming increasingly important. Such models arise naturally in modern data sets which have many measured characteristics available per individual observation as in, for example, population census data, scanner data, and text data. Such models also arise naturally even in data with a small number of measured characteristics in situations where the exact functional form with which the observed variables enter the model is unknown and we create many technical variables, a dictionary, from the raw characteristics. Examples covered by this scenario include semi-parametric models with non-parametric nuisance functions. More generally, models with many parameters relative to the sample size often arise when attempting to model complex phenomena. With increasing availability of such data sets in economics and other data science fields, new methods for analyzing those data have been developed. The `Julia` package `HDMjl` contains implementations of recently developed methods for high-dimensional approximately sparse models, mainly relying on forms of lasso and post-lasso as well as related estimation and inference methods. The methods are illustrated with econometric applications, but are also useful in other disciplines such as medicine, biology, sociology or psychology.

The methods which are implemented in this package are distinct from already available methods in other packages in the following four major ways:

1. First, we provide a version of Lasso regression that expressly handles and allows for non-Gaussian and heteroscedastic errors.
2. Second, we implement a theoretically grounded, data-driven choice of the penalty level λ in the Lasso regressions. To underscore this choice, we call the Lasso implementation in this package "rigorous"Lasso (=`rlasso`). The prefix `r` in function names should underscore this. In high dimensional settings cross-validation is very popular; but it lacks a theoretical justification for use in the present context and some theoretical proposals for the choice of λ are often not feasible. Moreover, the theoretically grounded, data-driven choice redundancies cross-validation which is time-consuming particularly in large data sets.
3. Third, we provide efficient estimators and uniformly valid confidence intervals for various low-dimensional causal/structural parameters appearing in high-dimensional approximately sparse models. For example, we provide efficient estimators and uniformly valid confidence intervals for a regression coefficient on a target variable (e.g., a treatment or policy variable) in a high-dimensional sparse regression model. Target variable in this context means the object not interest, e.g. a pre-specified regression coefficient. We also provide estimates and confidence intervals for average treatment effect (ATE) and average treatment effect for the treated (ATET), as well extensions of these parameters to the endogenous setting.
4. Fourth, joint/ simultaneous confidence intervals for estimated coefficients in a high-dimensional approximately sparse models are provided, based on the methods and theory developed in Belloni, Chernozhukov, and Kato (2014). They proposed uniformly valid confidence regions for regressions coefficients in a high-dimensional sparse Z-estimation problems, which include median, mean, and many other regression problems as special cases. In this article we apply this method to the coefficients of a Lasso regression and highlight this method with an empirical example.

2 How to get started

Julia is an open source software project and can be freely downloaded from the julialan g.org website along with its associated documentation. The Julia package `HDMjl` can be downloaded from juliahub. To install the `HDMjl` package from Julia we simply type.

```
] add HDMjl
```

The most current version of the package (development version) can installed by

```
using  Pkg; Pkg.add("HDMjl")
```

You may also install the dev version of the package by directly acquiring it from the repository by using

```
] add https://github.com/d2cml-ai/HDMjl.jl
```

or

```
import Pkg; Pkg.add(url = "https://github.com/d2cml-ai/HDMjl.jl")
```

Provided that your machine has a proper internet connection and you have write permission in the appropriate system directories, the installation of the package should proceed automatically. Once the `HDMjl` package is installed, it can be loaded to the current Julia session by the command

```
using HDMjl
```

3 Prediction using Approximate Sparsity

3.1 Prediction in Linear Models using Approximate Sparsity.

Consider high dimensional approximately sparse linear regression models. These models have a large number of regressors p, possibly much larger than the sample size n, but only a relatively small number $s = o(n)$ of these regressors are important for capturing accurately the main features of the regression function. The latter assumption makes it possible to estimate these models effectively by searching for approximately the right set of regressors.

The model reads

$$y_i = x_i'\beta_0 + \varepsilon_i, \quad E\left[\varepsilon_i x_i\right] = 0, \quad \beta_0 \in \mathbb{R}^p, \quad i = 1, \ldots, n$$

where y_i are observations of the response variable, $x_i = (x_{i,j}, \ldots, x_{i,p})$'s are observations of p-dimensional regressors, and ε_i's are centered disturbances, where possibly $p \gg n$. Assume that the data sequence is i.i.d. for the sake of exposition, although the framework covered is considerably more general. An important point is that the errors ε_i may be non-Gaussian or heteroskedastic (Belloni, Chen, Chernozhukov, and Hansen, 2012).

The model can be exactly sparse, namely

$$\|\beta_0\|_0 \leq s = o(n)$$

or approximately sparse, namely that the values of coefficients, sorted in decreasing order, $(|\beta_0|_{(j)})_{j=1}^p$ obey,

$$|\beta_0|_{(j)} \leq \mathrm{A} j^{-\mathrm{a}(\beta_0)}, \quad \mathrm{a}(\beta_0) > 1/2, \quad j = 1, \ldots, p$$

An approximately sparse model can be well-approximated by an exactly sparse model with sparsity index

$$s \propto n^{1/(2a(\beta_0))}.$$

In order to get theoretically justified performance guarantees, we consider the Lasso estimator with data-driven penalty loadings:

$$\hat{\beta} = \arg\min_{\beta \in \mathbb{R}^p} \mathbb{E}_n \left[(y_i - x_i'\beta)^2 \right] + \frac{\lambda}{n} \|\hat{\Psi}\beta\|_1$$

where $\|\beta\|_1 = \sum_{j=1}^p |\beta_j|$, $\hat{\Psi} = \mathrm{diag}\left(\hat{\psi}_1, \ldots, \hat{\psi}_p\right)$ is a diagonal matrix consisting of penalty loadings, and En abbreviates the empirical average. The penalty loadings are chosen to insure basic equivariance of coefficient estimates to rescaling of $x_{i,j}$ and can also be chosen to address heteroskedasticity in model errors. We discuss the choice of λ and $\hat{\Psi}$ below.

Regularization by the **l** 1-norm naturally helps the Lasso estimator to avoid overfitting, but it also shrinks the fitted coefficients towards zero, causing a potentially significant bias. In order to remove some of this bias, consider the Post-Lasso estimator that applies ordinary least squares to the model $\hat{T}$ selected by Lasso, formally,

$$\hat{T} = \mathrm{support}(\hat{\beta}) = \{j \in \{1, \ldots, p\} : |\hat{\beta}| > 0\}.$$

The Post-Lasso estimate is then defined as

$$\tilde{\beta} \in \arg\min_{\beta \in \mathbb{R}^p} \mathbb{E}_n \left\{ y_i - \sum_{j=1}^p x_{i,j}\beta_j \right)^2 : \beta_j = 0 \quad \text{if } \hat{\beta}_j = 0, \quad \forall j.$$

In words, the estimator is ordinary least squares applied to the data after removing the regressors that were not selected by Lasso. The Post-Lasso estimator was introduced and analysed in Belloni and Chernozhukov (2013).

A crucial matter is the choice of the penalization parameter λ. With the right choice of the penalty level, Lasso and Post-Lasso estimators possess excellent performance guarantees: They both achieve the near-oracle rate for estimating the regression function, namely with probability $1 - \gamma - o(1)$,

$$\sqrt{\mathbb{E}_n \left[\left(x_i' \left(\hat{\beta} - \beta_0 \right) \right)^2 \right]} \lesssim \sqrt{(s/n) \log p}$$

In high-dimensions setting, cross-validation is very popular in practice but lacks theoretical justification and so may not provide such a performance guarantee. In sharp contrast, the

choice of the penalization parameter λ in the Lasso and Post-Lasso methods in this package is theoretical grounded and feasible. Therefore we call the resulting method the "rigorous"Lasso method and hence add a prefix **r** to the function names.

In the case of homoscedasticity, we set the penalty loadings $\hat{\psi}_j = \sqrt{\mathrm{E}_n x_{i,j}^2}$, which insures basic equivariance properties. There are two choices for penalty level λ : the X-independent choice and X dependent choice. In the X-independent choice we set the penalty level to

$$\lambda = 2c\sqrt{n}\hat{\sigma}\Phi^{-1}(1-\gamma/(2p)),$$

where Φ denotes the cumulative standard normal distribution, $\hat{\sigma}$ is a preliminary estimate of $\sigma = \sqrt{\mathbb{E}\varepsilon^2}$, and c is a theoretical constant, which is set to $c = 1.1$ by default for the Post-Lasso method and $c = .5$ for the Lasso method, and γ is the probability level, which is set to $\gamma = .1$ by default. The parameter γ can be interpreted as the probability of mistakenly not removing X 's when all of them have zero coefficients. In the X-dependent choice the penalty level is calculated as

$$\lambda = 2c\hat{\sigma}\Lambda(1-\gamma \mid X),$$

where

$$\Lambda(1-\gamma \mid X) = (1-\gamma) \text{ - quantile of } n\left\|\mathbb{E}_n\left[x_i e_i\right]\right\|_\infty \mid X,$$

where $X = [x_1, ..., x_n]'$ and e_i are iid $N(0,1)$, generated independently from X; this quantity is approximated by simulation. The X-independent penalty is more conservative than the X-dependent penalty. In particular the X-dependent penalty automatically adapts to highly correlated designs, using less aggressive penalization in this case Belloni, Chernozhukov, and Hansen (2010).

In the case of heteroskedasticity, the loadings are set to $\hat{\psi}_j = \sqrt{\mathbb{E}_n\left[x_{ij}^2\hat{\varepsilon}_i^2\right]}$, where $\hat{\varepsilon}_i$ are preliminary estimates of the errors. The penalty level can be X-independent (Belloni, Chen, Chernozhukov, and Hansen, 2012):

$$\lambda = 2c\sqrt{n}\Phi^{-1}(1-\gamma/(2p))$$

or it can be X-dependent and estimated by a multiplier bootstrap procedure (Chernozhukov, Chetverikov, and Kato, 2013)

$$\lambda = c \times c_W(1-\gamma),$$

where $c_W(1-\gamma)$ is the $1-\gamma$-quantile of the random variable W, conditional on the data, where

$$W := n \max_{1\le j\le p}\left|2\mathbb{E}_n\left[x_{ij}\hat{\varepsilon}_i e_i\right]\right|,$$

where e_i are iid standard normal variables distributed independently from the data, and $\hat{\varepsilon}_i$ denotes an estimate of the residuals.

Estimation proceeds by iteration. The estimates of residuals $\hat{\varepsilon}_i$ are initialized by running least squares of y_i on five regressors that are most correlated to y_i. This implies conservative starting values for λ and the penalty loadings, and leads to the initial Lasso and Post-Lasso estimates, which are then further updated by iteration. The resulting iterative procedure is fully justified in the theoretical literature.

3.2 A Joint Significance Test for Lasso Regression.

A basic question frequently arising in empirical work is whether the Lasso regression has explanatory power, comparable to a F-test for the classical linear regression model. The construction of a joint significance test follows (Chernozhukov, Chetverikov, and Kato, 2013) (Appendix M), and can be described as: Based on the model $y_i = a_0 + x_i' b_0 + \varepsilon_i$, the null hypothesis of joint statistical in-significance is $b_0 = 0$. The alternative is that of the joint statistical significance: $b_0 \neq 0$. The null hypothesis implies that

$$\mathrm{E}\left[(y_i - a_0)\, x_i\right] = 0$$

and restriction can be tested using the sup-score statistic:

$$S = \left\| \sqrt{n} \mathbb{E}_n \left[(y_i - \hat{a}_0)\, x_i\right] \right\|_\infty$$

where $\hat{a}_i = \mathbb{E}_n [y_i]$. The critical value for this statistic can be approximated by the multiplier bootstrap procedure, which simulates the statistic:

$$S^* = \left\| \sqrt{n} \mathbb{E}_n \left[(y_i - \hat{a}_0)\, x_i g_i\right] \right\|_\infty$$

where g_i 's are iid $N(0,1)$, conditional on the data. The $(1-\alpha)$-quantile of S^* serves as the critical value, $c(1-\alpha)$. We reject the null if $S > c(1-\alpha)$ in favor of statistical significant, and we keep the null of non-significance otherwise. This test procedure is implemented in the package when calling the summary-method of rlasso-objects.

Example. (Prediction Using Lasso and Post-Lasso) Consider generated data from a sparse linear model:

```
using Random
Random.seed!(1234);
n = 100;
p = 100;
s = 3;
X = randn(n, p);
beta = vcat(fill(5, s), zeros(p - s));
Y = X * beta + randn(n);
```

The Post-Lasso procedure fits an OLS regression excluding the variables not previously selected by Lasso. The rlasso algorithm uses the standard errors of the residuals from this regression to evaluate whether there has been a gain in the goodness of the fit in the current iteration. Just like most of the functions in the package, rlasso returns a dictionary with the results of the regression.

We can estimate the models using Lasso

```
rlasso(X, Y, post = true)

Dict{String, Any} with 15 entries:
  "tss"          => 8466.4
  "dev"          => [8.56479, -6.50463, -1.51601, 8.77722,
                     -5.1338, -7.07815, 7...
  "model"        => [0.970656 0.262456 ... 1.86802 -0.460151;
                     -0.979218 -0.022244...
  "loadings"     => [1.99904, 1.58734, 1.67152, 1.86208, 1.859,
                     1.8311, 1.45692...
  "sigma"        => 1.56664
  "lambda0"      => 81.3601
  "lambda"       => [162.642, 129.146, 135.995, 151.499,
                     151.248, 148.978, 118....
  "intercept"    => -0.196115
  "iter"         => 16
  "residuals"    => [0.119342, -0.25299, -1.17148, 0.282269,
                     -0.643579, -0.5525...
  "rss"          => 242.981
  "index"        => Bool[1, 1, 1, 0, 0, 0, 0, 0, 0, 0  ...  0,
                     0, 0, 0, 0, 0, 0, ...
  "beta"         => [4.31658, 4.39195, 4.45657, 0.0, 0.0, 0.0,
                     0.0, 0.0, 0.0, 0...
  "options"      => Dict{String, Any}("intercept"=>true,
                     "post"=>false, "meanx"...
  "coefficients" => [-0.196115, 4.31658, 4.39195, 4.45657, 0.0,
                     0.0, 0.0, 0.0, ...

post_lasso_reg = rlasso(X, Y, post = true) #now use post-lasso
post_lasso_reg["coefficients"]'

1x101 adjoint(::Vector{Float64}) with eltype Float64:
    -0.00682754  5.00958  4.93178  5.17705  ...  0.0  0.0
    0.0  0.0  0.0  0.0  0.0
```

4 Inference on Target Regression Coefficient

Here we consider inference on the target coefficient α in the model:

$$y_i = d_i\alpha_0 + x_i'\beta_0 + \epsilon_i, \quad \mathbb{E}_i\left(x_i', d_i'\right)' = 0$$

Here d_i is a target regressor such as treatment, policy or other variable whose regression coefficient α_0 we would like to learn (Belloni, Chernozhukov, and Hansen, 2014). If we are interested in coefficients of several or even many variables, we can simply write the model in the above form treating each variable of interest as d_i in turn and then applying the estimation and inference procedures described below.

We assume approximate sparsity for $x_i'\beta_0$ with sufficient speed of decay of the sorted components of β_0, namely $\mathrm{a}(\beta_0) > 1$. This condition translates into having a sparsity index

$s \ll \sqrt{n}$. In general d_i is correlated to x_i, so α_0 cannot be consistently estimated by the regression of y_i on d_i. To keep track of the relationship of d_i to x_i, write

$$d_i = x_i'\pi_0^d + \rho_i^d, \quad \mathbb{E}\rho_i^d x_i = 0$$

To estimate α_0, we also impose approximate sparsity on the regression function $x_i'\pi_0^d$ with sufficient speed of decay of sorted components of π_0^d, namely a(ˆd_0) > 1$.

The Orthogonality Principle. Note that we can not use naive estimates of α_0 based simply on applying Lasso and Post-Lasso to the first equation. Such a strategy in general does not produce root-n consistent and asymptotically normal estimators of α, due to the possibility of large omitted variable bias resulting from estimating the nuisance function $x_i'\beta_0$ in high-dimensional setting. In order to overcome the omitted variable bias, we need to use orthogonalized estimating equations for α_0. Specifically we seek to find a score $\psi(w_i, \alpha, \eta)$, where $w_i = (y_i, x_i')'$ and η is the nuisance parameter, such that

$$\mathbb{E}\psi(w_i, \alpha_0, \eta_0) = 0, \quad \frac{\partial}{\partial \eta}\mathbb{E}\psi(w_i, \alpha_0, \eta_0) = 0$$

The second equation is the orthogonality condition, which states that the equations are not sensitive to the first-order perturbations of the nuisance parameter η near the true value. The latter property allows estimation of these nuisance parameters η_0 by regularized estimators $\hat{\eta}$, where regularization is done via penalization or selection. Without this property, regularization may have too much effect on the estimator of α_0 for regular inference to proceed. The estimators α of α_0 solve the empirical analog of the equation above,

$$\mathbb{E}_n\psi(w_i, \hat{\alpha}, \hat{\eta}) = 0,$$

where we have plugged in the estimator $\hat{\eta}$ for the nuisance parameter.

Due to the orthogonality property the estimator is first-order equivalent to the infeasible estimator $\tilde{\alpha}$ solving

$$\mathbb{E}_n\psi(w_i, \tilde{\alpha}, \eta_0) = 0,$$

where we use the true value of the nuisance parameter. The equivalence holds in a variety of models under plausible conditions. The systematic development of the orthogonality condition for inference on low-dimensional parameters in modern high-dimensional settings is given in Chernozhukov, Hansen, and Spindler (2015a).

In turns out that in the linear model the orthogonal equations are closely connected to the classical ideas of partialling out.

4.1 Intuition for the Orthogonality Principle in Linear Models via Partialling Out.

One way to think about estimation of α_0 is to think of the regression model:

$$\rho_i^y = \alpha_0\rho_i^d + \epsilon_i$$

where ρ_i^y is the residual that is left after partialling out the linear effect of x_i from y_i and ρ_i^d is the residual that is left after partialling out the linear effect of x_i from d_i, both done in the population. Note that we have $\mathbb{E}\rho_i^y x_i = 0$, i.e. $\rho_i^y = y_i - x_i'\pi_0^y$ where $x_i'\pi_0^y$ is the linear projection of y_i on x_i. After partialling out, α_0 is the population regression coefficient in the univariate regression of ρ_i^y on ρ_i^d. This is the Frisch-Waugh-Lovell theorem. Thus, $\alpha = \alpha_0$ solves the population equation:

$$\mathbb{E}\left(\rho_i^y - \alpha\rho_i^d\right)\rho_i^d = 0$$

The score associated to this equation is:

$$\psi\left(w_i, \alpha, \eta\right) = \left(y_i - x_i'\pi^y\right) - \alpha\left(d_i - x_i'\pi^d\right)\left(d_i - x_i'\pi^d\right), \quad \eta = \left(\pi^{y'}, \pi^{d'}\right)',$$
$$\psi\left(w_i, \alpha_0, \eta_0\right) = \left(\rho_i^y - \alpha\rho_i^d\right)\rho_i^d, \quad \eta_0 = \left(\pi_0^{y'}, \pi_0^{d'}\right).$$

It is straightforward to check that this score obeys the orthogonality principle; moreover, this score is the semi-parametrically efficient score for estimating the regression coefficient α_0.

In low-dimensional settings, the empirical version of the partialling out approach is simply another way to do the least squares. Let's verify this in an example. First, we generate some data

```
using DataFrames, HDMjl, CSV, GLM

url = "https://raw.githubusercontent.com/d2cml-ai/HDMjl.jl/prueba/data/4_1.csv"
dta = DataFrame(CSV.read(download(url), DataFrame));
n, p = size(dta);
y = dta[:,"y"];
d = dta[:,"d"];
x = Matrix(dta[:,3:end]);
```

We can estimate α_0 by running full least squares:

```
full_fit = lm(hcat(ones(length(y)), Matrix(dta[:,2:end])), y);
DataFrame(
    Estimate = coef(full_fit)[2],
    Std_Error = stderror(full_fit)[2])
```

```
1×2 DataFrame
 Row   Estimate  Std_Error
       Float64   Float64

   1   0.978075  0.0137122
```

Another way to estimate $alpha_0$ is to first partial out the x-variables from y_i and d_i, and run least squares on the residuals:

```
rY_1 = lm(hcat(ones(length(y)), Matrix(dta[:,3:end])), y);
rY = y - predict(rY_1)
rD_1 = lm(hcat(ones(length(y)), Matrix(dta[:,3:end])), d);
rD = d - predict(rD_1);
```

```
partial_fit_ls = lm(hcat(ones(length(y)), rD), rY)
DataFrame(
    Estimate = coef(partial_fit_ls)[2],
    Std_Error = stderror(partial_fit_ls)[2]
    )
```

One can see that the estimates are identical, while standard errors are nearly identical. In fact, the standard errors are asymptotically equivalent due to the orthogonality property of the estimating equations associated with the partialling out approach.

In high-dimensional settings, we can no longer rely on the full least-squares and instead may rely on Lasso or Post-Lasso for partialling out. This amounts to using orthogonal estimating equations, where we estimate the nuisance parameters by Lasso or Post-Lasso. Let's try this in the above example, using Post-Lasso for partialling out:

```
rY_1 = HDMjl.rlasso(hcat(ones(length(y)), Matrix(dta[:,3:end])), y);
rY = rY_1["residuals"]
rD_1 = HDMjl.rlasso(hcat(ones(length(y)), Matrix(dta[:,3:end])), d);
rD = rD_1["residuals"]
partial_fit_postlasso = lm(hcat(ones(length(y)), rD), rY)
DataFrame(
    Estimate = coef(partial_fit_postlasso)[2],
    Std_Error = stderror(partial_fit_postlasso)[2]
    )
```

We see that this estimate and standard errors are nearly identical to the previous estimates given above. In fact they are asymptotically equivalent to the previous estimates in the low-dimensional settings, with the advantage that, unlike the previous estimates, they will continue to be valid in the high-dimensional settings.

The orthogonal estimating equations method – based on partialling out via Lasso or post-Lasso – is implemented by the function `rlassoEffect`, using `method= "partialling out"`:

```
Eff = HDMjl.rlassoEffect(x, y, d, method = "partialling out");
HDMjl.r_summary(Eff);
```

Another orthogonal estimating equations method – based on the double selection of covariates – is implemented by the the function `rlassoEffect`, using `method= "double selection"`. Algorithmically the method is as follows:

1. Select controls x_{ij}'s that predict y_i by Lasso.
2. Select controls x_{ij}'s that predict d_i by Lasso.
3. Run OLS of y_i on d_i and the union of controls selected in steps 1 and 2.

```
Eff = HDMjl.rlassoEffect(x, y, d, method = "double selection");
HDMjl.r_summary(Eff);
```

4.2 Inference: Confidence Intervals and Significance Testing.

The function `rlassoEffects` does inference – namely construction of confidence intervals and significance testing – for target variables. Those can be specified either by the variable names, an integer valued vector giving their position in x or by a logical vector indicating the variables for which inference should be conducted. It returns an object of S3 class

`rlassoEffects` for which the methods `r_summary`, `r_print`, and `r_confint` are provided. `rlassoEffects` is a wrap function for the function `rlassoEffect` which does inference for a single target regressor. The theoretical underpinning is given in Belloni, Chernozhukov, and Hansen (2014) and for a more general class of models in Belloni, Chernozhukov, and Kato (2014). The function `rlassoEffects` might either be used in the form `rlassoEffects(x, y, index)` where `x` is a matrix, `y` denotes the outcome variable and index specifies the variables of `x` for which inference is conducted. This can done by an integer vector (postion of the variables), a logical vector or the name of the variables. An alternative usage is as `rlassoEffects(formula, data, I)` where I is a one-sided formula which specifies the variables for which is inference is conducted. For further details we refer to the help page of the function and the following examples where both methods for usage are shown.

Here is an example of the use.

```
url = "https://raw.githubusercontent.com/d2cml-ai/HDMjl.jl/prueba/data/4_2.csv"
data = DataFrame(CSV.read(download(url), DataFrame));
n, p = size(data);
y = data[:,1];
#d = dta[:,"d"];
x = Matrix(data[:,2:end]);
```

We can do inference on a set of variables of interest, e.g. the first, second, third, and the fiftieth:

```
lasso_effects = HDMjl.rlassoEffects(x, y, index = [1,2,3,50]);
HDMjl.r_print(lasso_effects, digits = 4)
HDMjl.r_summary(lasso_effects);
HDMjl.r_confint(lasso_effects);
```

The two methods are first-order equivalent in both low-dimensional and high-dimensional settings under regularity conditions. Not surprisingly we see that in the numerical example of this section, the estimates and standard errors produced by the two methods are very close to each other.

It is also possible to estimate joint confidence intervals. The method relies on a multiplier bootstrap method as described in Belloni, Chernozhukov, and Kato (2014). Joint confidence intervals can be invoked by setting the option joint to `true` in the method `confint` for objects of class `rlassoEffects`.

```
HDMjl.r_confint(lasso_effects, joint = true);
```

We will also demonstrate the application of joint confidence intervals in an empirical application in the next section.

Finally, we can also plot the estimated effects with their confidence intervals:

```
plot
```

For logistic regression we provide the functions `rlassologit` and `rlassologitEffects`.

4.3 Application: the effect of gender on wage.

In Labour Economics an important question is how the wage is related to the gender of the employed. We use US census data from the year 2012 to analyse the effect of gender and interaction effects of other variables with gender on wage jointly. The dependent variable

is the logarithm of the wage, the target variable is `female` (in combination with other variables). All other variables denote some other socio-economic characteristics, e.g. marital status, education, and experience. For a detailed description of the variables we refer to the help page.

First, we load and prepare the data.

```
url = "https://github.com/cran/hdm/raw/master/data/cps2012.rda"
cps2012 = load(download(url))["cps2012"];
n, p = size(cps2012);
size(cps2012)
x_formula = @formula(lnw ~ -1 + female + female*widowed + female*divorced + female*sepa
                     female*hsd08 + female*hsd911 + female*hsg + female*cg + female*ad +
x_dframe = ModelFrame( x_formula, cps2012)
x1 = ModelMatrix(x_dframe)
X = x1.m[:,Not(1:16)];
X = hcat(x1.m[:,1:16], X)

fom_1 =     ["widowed", "divorced", "separated", "nevermarried", "hsd08", "hsd911", "hs
            "we", "exp1", "exp2", "exp3"];
data = cps2012[:,fom_1];
sub_data = ones(size(data)[1])
for i in 1:size(data)[2]
    if i <= (size(data)[2] -1)
        sub_data = hcat(sub_data, Matrix(data[:, i] .* data[:, Not(1:i)]) )
    end
end
sub_data = sub_data[:,2:end]
size(sub_data)
x = hcat(X, sub_data)
size(x)
filter = var.(eachcol(x)) .!= 0
x = x[:,filter]
print(size(x))
index_gender = [1,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31];
y = cps2012.lnw;
```

The parameter estimates for the target parameters, i.e. all coefficients related to gender (i.e. by interaction with other variables) are calculated and summarized by the following commands

```
effects_female = HDMjl.rlassoEffects(x, y, index = index_gender);
HDMjl.r_summary(effects_female);
```

Finally, we estimate and plot confident intervals, first "pointwise" and then the joint confidence intervals. % ————

```
joint_CI = HDMjl.r_confint(effects_female, 0.95, joint = true);
joint_CI;
```

This analysis allows a closer look how discrimination according to gender is related to other socio- economic variables.

4.4 Application: Estimation of the treatment effect in a linear model with many confounding factors.

A part of empirical growth literature has focused on estimating the effect of an initial (lagged) level of GDP (Gross Domestic Product) per capita on the growth rates of GDP per capita. In particular, a key prediction from the classical Solow-Swan-Ramsey growth model is the hypothesis of convergence, which states that poorer countries should typically grow faster and therefore should tend to catch up with the richer countries, conditional on a set of institutional and societal characteristics. Covariates that describe such characteristics include variables measuring education and science policies, strength of market institutions, trade openness, savings rates and others.

Thus, we are interested in a specification of the form:

$$y_i = \alpha_0 d_i + \sum_{j=1}^{p} \beta_j x_{ij} + \varepsilon_i,$$

where y_i is the growth rate of GDP over a specified decade in country i, d_i is the log of the initial level of GDP at the beginning of the specified period, and the $x_{ij}'s$ form a long list of country i's characteristics at the beginning of the specified period. We are interested in testing the hypothesis of convergence, namely that $\alpha_1 < 0$. Given that in the Barro and Lee (1994) data, the number of covariates p is large relative to the sample size n, covariate selection becomes a crucial issue in this analysis. We employ here the partialling out approach (as well as the double selection version) introduced in the previous section.

First, we load and prepare the data

```
url = "https://github.com/cran/hdm/raw/master/data/GrowthData.rda";
GrowthData = load(download(url))["GrowthData"];
y = GrowthData[:, 1];
d = GrowthData[:, 3];
X = Matrix(GrowthData[:, Not(1, 2, 3)]);
X_1 = Matrix(GrowthData[:, Not(1, 2)]);
```

Now we can estimate the effect of the initial GDP level. First, we estimate by OLS:

```
ls_effect = lm(hcat(ones(length(y)), X_1)[:, 1:42], y);
```

Second, we estimate the effect by the partialling out by Post-Lasso:

```
lasso_effect = HDMjl.rlassoEffect(X, y, d, method = "partialling out");
HDMjl.r_summary(lasso_effect);
```

Third, we estimate the effect by the double selection method:

```
doublesel_effect = HDMjl.rlassoEffect(X, y, d, method = "double selection");
HDMjl.r_summary(doublesel_effect);
```

We then collect results in a nice latex table:

```
using PrettyTables
# table = zeros(4,2)
# table[1,:] = [coef(ED_ols)[2], stderror(ED_ols)[2]]
# table[2,:] = [ED_2sls["coefficients"][1,2], ED_2sls["se"][1]]
```

```
# table[3,:] = Matrix(HDMjl.r_summary(lasso_IV_Z)[:,2:3]);
# table[4, :] = Matrix(HDMjl.r_summary(lasso_IV_XZ)[:, 2:3]);
```

We see that the OLS estimates are noisy, which is not surprising given that p is comparable to n. The partial regression approaches, based on Lasso selection of covariates in the two projection equations, in contrast yields much more precise estimates, which does support the hypothesis of conditional convergence. We note that the partial regression approaches represent a special case of the orthogonal estimating equations approach.

5 Instrumental Variable Estimation in a High-Dimensional Setting

In many applied settings the researcher is interested in estimating the (structural) effect of a variable (treatment variable), but this variable is endogenous, i.e. correlated with the error term. In this case, instrumental variables (IV) methods are used for identification of the causal parameters.

We consider the linear instrumental variables model:

$$y_i = \alpha_0 d_i + \gamma_0 x_i' + \epsilon_i,$$

$$d_i = z_i' \prod + \beta_0 x_i' + \nu_i,$$

where $\mathbb{E}\left[\epsilon_i(x_i', z_i')\right] = 0$, $\mathbb{E}\left[\nu(x_i', z_i')\right] = 0$, but $\mathbb{E}[\epsilon_i, \nu_i] \neq 0$ leading to endogeneity. In this setting d_i is a scalar endogenous variable of interest, z_i is a pz-dimensional vector of instruments and x_i is a px-dimensional vector of control variables.

In this section we present methods to estimate the effect α_0 in a setting where either x is high-dimensional or z is high-dimensional. Instrumental variables estimation with very many instruments was analysed in Belloni, Chen, Chernozhukov, and Hansen (2012), the extension with many instruments and many controls in Chernozhukov, Hansen, and Spindler (2015b).

5.1 Estimation and Inference.

To get efficient estimators and uniformly valid confidence intervals for the structural parameters there are different strategies which are asymptotically equivalent where again orthogonalization (via partialling out) is a key concept.

In the case of the high-dimensional instrument z_i and low-dimensional x_i. We predict the endogenous variable d_i using (Post-)Lasso regression of d_i on the instruments z_i and x_i, forcing the inclusion of x_i. Then we compute the IV estimator (2SLS) $\hat{\alpha}$ of the parameter α_0 using the predicted value $\hat{d}_i$ as instrument and using x_i's as controls. We then perform inference on α_0 using $\hat{\alpha}$ and conventional heteroskedasticity robust standard errors.

In the case of the low-dimensional instrument z_i and high-dimensional x_i, we first partial out the effect of x_i from d_i, y_i, and z_i by (Post-)Lasso. Second, we then use the residuals to compute the IV estimator (2SLS) $\hat{\alpha}$ of the parameter 0. We then perform inference on α_0 using $\hat{\alpha}$ and conventional heteroskedasticity robust standard errors.

In the case of the high-dimensional instrument z_i and high-dimensional x_i the algorithm described in Chernozhukov, Hansen, and Spindler (2015b) is adopted.

Julia Implementation. The wrap function `rlassoIV` handles all of the previous cases. It has the options **select.X** and **select.Z** which implement selection of either covariates or instruments, both with default values set to **true**. The class of the return object depends on the chosen options, but the methods summary, print and confint are available for all. The functions **rlassoSelectX** and **rlassoSelectZ** do selection on x-variables only and z-variables only. Selection on both is done in **rlassoIV**. All functions employ the option **post=true** as default, which uses post-Lasso for partialling out. By setting **post=true** we can employ Lasso instead of Post-Lasso. Finally, the package provides the standard function **tsls**, which implements the "classical" two-stage least squares estimation.

Function usage both the family of **rlassoIV**-functions and the family of the functions for treatment effects , which are introduced in the next section, allow use with both formula-interface and by handing over the prepard model matrices. Hence the general pattern for use with formula is function(formula, data, …) where formula consists of two-parts and is a member of the class Formula. These formulas are of the pattern y d + x | x + z where y is the outcome variable, x are exogenous variables, d endogenous varialbes (if several ones are allowed depends on the concrete function), and z denote the instrumental variables. A more primitive use of the functions is by simply hand over the required group of variables as matrices: function(x=x, d=d, y=y, z=z). In some of the following examples both alternatives are displayed.

5.2 Application: Economic Development and Institutions.

Estimating the causal effect of institutions on output is complicated by the simultaneity between institutions and output: specifically, better institutions may lead to higher incomes, but higher incomes may also lead to the development of better institutions. To help overcome this simultaneity, Acemoglu, Johnson, and Robinson (2001) use mortality rates for early European settlers as an instrument for institution quality. The validity of this instrument hinges on the argument that settlers set up better institutions in places where they are more likely to establish long-term settlements, that where they are likely to settle for the long term is related to settler mortality at the time of initial colonization, and that institutions are highly persistent. The exclusion restriction for the instrumental variable is then motivated by the argument that GDP, while persistent, is unlikely to be strongly influenced by mortality in the previous century, or earlier, except through institutions.

In this application, we consider the problem of selecting controls. The input raw controls are the Latitude and the continental dummies. The technical controls can include various polynomial transformations of the Latitude, possibly interacted with the continental dummies. Such flexible specifications of raw controls results in the high-dimensional x, with dimension comparable to the sample size.

First, we process the data

```
using Statistics, StatsModels, RData
url = "https://github.com/cran/hdm/raw/master/data/AJR.rda";
AJR = load(download(url))["AJR"];
y = AJR[!,"GDP"]
d = AJR[!,"Exprop"]
z = AJR[!,"logMort"];
```

```
x_formula = @formula(GDP ~ -1 + Latitude + Latitude2 + Africa + Asia + Namer + Samer
    + Latitude*Latitude2 + Latitude*Africa + Latitude*Asia + Latitude*Namer + Latitude*
    + Latitude2*Africa + Latitude2*Asia + Latitude2*Namer + Latitude2*Samer
    + Africa*Asia + Africa*Namer + Africa*Samer
    + Asia*Namer + Asia*Samer
    + Namer*Samer)
x_dframe = ModelFrame( x_formula, AJR)
x1 = ModelMatrix(x_dframe)
x = x1.m
```

Then we estimate an IV model with selection on the X

```
AJR_Xselect  = HDMjl.rlassoIV(x, d, y, z, select_X=true, select_Z=false);
HDMjl.r_print(AJR_Xselect)
HDMjl.r_summary(AJR_Xselect);
```

It is interesting to understand what the procedure above is doing. In essence, it partials out x_i from y_i, d_i and z_i using Post-Lasso and applies the 2SLS to the residual quantities.

Let us investigate partialling out in more detail in this example. We can first try to use OLS for partialling out:

```
rY_1 = lm(@formula(GDP ~ Latitude + Latitude2 + Africa + Asia + Namer + Samer + Latitud
          + Latitude2*Africa + Latitude2*Asia + Latitude2*Namer + Latitude2*Samer + Afr
          + Asia*Namer + Asia*Samer + Namer*Samer), AJR)
rY = y - predict(rY_1)

rD_1 = lm(@formula(Exprop ~ Latitude + Latitude2 + Africa + Asia + Namer + Samer + Lati
          + Latitude2*Africa + Latitude2*Asia + Latitude2*Namer + Latitude2*Samer + Afr
          + Asia*Namer + Asia*Samer + Namer*Samer), AJR)
rD = d - predict(rD_1)

rZ_1 = lm(@formula(logMort ~ Latitude + Latitude2 + Africa + Asia + Namer + Samer + Lat
          + Latitude2*Africa + Latitude2*Asia + Latitude2*Namer + Latitude2*Samer + Afr
          + Asia*Namer + Asia*Samer + Namer*Samer), AJR)
rZ = z - predict(rZ_1);

ivfit_lm = HDMjl.tsls(rD, rY, rZ, nothing, intercept=true)
DataFrame(Estimate = ivfit_lm["coefficients"][1,2], Std_Error = ivfit_lm["se"])
```

We see that the estimates exhibit large standard errors. The imprecision is expected because dimension of x is quite large, comparable to the sample size.

Next, we replace the OLS operator by post-Lasso for partialling out

```
x_formula1 = @formula(GDP ~ Latitude + Latitude2 + Africa + Asia + Namer + Samer
    + Latitude*Latitude2 + Latitude*Africa + Latitude*Asia + Latitude*Namer + Latitude*
    + Latitude2*Africa + Latitude2*Asia + Latitude2*Namer + Latitude2*Samer
    + Africa*Asia + Africa*Namer + Africa*Samer
    + Asia*Namer + Asia*Samer
    + Namer*Samer)
x_dframe1 = ModelFrame( x_formula, AJR)
x1_1 = ModelMatrix(x_dframe)
```

```
xx = x1.m;

rY_1 = HDMjl.rlasso(xx, y);
rY = rY_1["residuals"]
rD_1 = HDMjl.rlasso(xx, d);
rD = rD_1["residuals"]
rZ_1 = HDMjl.rlasso(xx, z);
rZ = rZ_1["residuals"]

ivfit_lasso = HDMjl.tsls(rD, rY, rZ, intercept = true)
DataFrame(Estimate = ivfit_lasso["coefficients"][1,2], Std_Error = ivfit_lasso["se"][1]
```

This is exactly what command `rlassoIV` is doing by calling the command `rlassoSelectX`, so the numbers we see are identical to those reported first. In comparison to OLS results, we see precision is improved by doing selection on the exogenous variables.

```
using Statistics, GLM
url = "https://github.com/cran/hdm/raw/master/data/EminentDomain.rda";
EminentDomain = load(download(url))["EminentDomain"];
z = EminentDomain["logGDP"]["z"];
x = EminentDomain["logGDP"]["x"];
d = EminentDomain["logGDP"]["d"];
y = EminentDomain["logGDP"]["y"];
x = x[:, (mean(x, dims = 1) .> 0.05)'];
z = z[:, (mean(z, dims = 1) .> 0.05)'];
```

5.3 Application: Impact of Eminent Domain Decisions on Economic Outcomes.

Here we investigate the effect of pro-plaintiff decisions in cases of eminent domain (government's takings of private property) on economic outcomes. The analysis of the effects of such decisions is complicated by the possible endogeneity between judicial decisions and potential economic outcomes. To address the potential endogeneity, we employ an instrumental variables strategy based on the random assignment of judges to the federal appellate panels that make the decisions. Because judges are randomly assigned to three-judge panels, the exact identity of the judges and their demographics are randomly assigned conditional on the distribution of characteristics of federal circuit court judges in a given circuit-year.

Under this random assignment, the characteristics of judges serving on federal appellate panels can only be related to property prices through the judges' decisions; thus the judge's characteristics will plausibly satisfy the instrumental variable exclusion restriction. For further information on this application and the data set we refer to Chen and Yeh (2010) and Belloni, Chen, Chernozhukov, and Hansen (2012).

First, we load the data an construct the matrices with the controls (x), instruments (z), outcome (y),and treatment variables (d). Here we consider regional GDP as the outcome variable.

```
using Statistics, GLM
url = "https://github.com/cran/hdm/raw/master/data/EminentDomain.rda";
EminentDomain = load(download(url))["EminentDomain"];
z = EminentDomain["logGDP"]["z"];
```

```
x = EminentDomain["logGDP"]["x"];
d = EminentDomain["logGDP"]["d"];
y = EminentDomain["logGDP"]["y"];
x = x[:, (mean(x, dims = 1) .> 0.05)'];
z = z[:, (mean(z, dims = 1) .> 0.05)'];
```

As mentioned above, y is the economic outcome, the logarithm of the GDP, d the number of pro plaintiff appellate takings decisions in federal circuit court c and year t, x is a matrix with control variables, and z is the matrix with instruments. Here we consider socio-economic and demographic characteristics of the judges as instruments.

First, we estimate the effect of the treatment variable by simple OLS and 2SLS using two instruments:

```
ED_ols = lm(hcat(ones(length(vec(y))), hcat(d, x)), vec(y));
ED_2sls = HDMjl.tsls(d, y, z[:,1:2], x, intercept = false);
```

Next, we estimate the model with selection on the instruments

```
lasso_IV_Z = HDMjl.rlassoIV(x, d, y, z, select_X = false, select_Z = true);
HDMjl.r_summary(lasso_IV_Z);
```

% ——— Finally, we do selection on both the x and z variables. % ———

```
lasso_IV_XZ = HDMjl.rlassoIV(x, d, y, z, select_X = true, select_Z = true);
HDMjl.r_summary(lasso_IV_XZ);
HDMjl.r_confint(lasso_IV_XZ);
HDMjl.r_print(lasso_IV_XZ)
```

Comparing the results we see, that the OLS estimates indicate that the influence of pro plaintiff appellate takings decisions in federal circuit court is significantly positive, but the 2SLS estimates which account for the potential endogeneity render the results insignificant. Employing selection on the instruments yields similar results. Doing selection on both the x- and z-variables results in extremely imprecise estimates.

Finally, we compare all results

```
using PrettyTables
table = zeros(4,2)
table[1,:] = [coef(ED_ols)[2], stderror(ED_ols)[2]]
table[2,:] = [ED_2sls["coefficients"][1,2], ED_2sls["se"][1]]
table[3,:] = Matrix(HDMjl.r_summary(lasso_IV_Z)[:,2:3]);
table[4, :] = Matrix(HDMjl.r_summary(lasso_IV_XZ)[:, 2:3]);


index = ["ols regression", "IV estimation ", "selection on Z", "selection on X and Z"]
pretty_table(hcat(index, table), show_row_number = false, header = [" ", "Estimate", "S
```

6 Inference on Treatment Effects in a High-Dimensional Setting

In this section, we consider estimation and inference on treatment effects when the treatment variable d enters non-separably in determination of the outcomes. This case is more

complicated than the additive case, which is covered as a special case of Section 3. However, the same underlying principle – the orthogonality principle – applies for the estimation and inference on the treatment effect parameters. Estimation and inference of treatment effects in a high-dimensional setting is analysed in Belloni, Chernozhukov, Fern´andez-Val, and Hansen (2013).

6.1 Treatment Effects Parameters – a short Introduction.

In many situations researchers are asked to evaluate the effect of a policy intervention. Examples are the effectiveness of a job related training program or the effect of a newly developed drug. We consider n units or individuals, $i = 1, ..., n$. For each individual we observe the treatment status. The treatment variable D_i takes the value 1, if the unit received (active) treatment, and 0, if it received the control treatment. For each individual we observe the outcome for only one of the two potential treatment states. Hence, the observed outcome depends on the treatment status and is denoted by $Y_i(D_i)$. One important parameter of interest is the average treatment effect (ATE):

$$\mathbb{E}[Y(1) - Y(0)] = \mathbb{E}[Y(1)] - \mathbb{E}[Y(0)].$$

This quantity can be interpreted as the average effect of the policy intervention.

Researchers might also be interested in the average treatment effect on the treated (ATET) given by

$$\mathbb{E}[Y(1) - Y(0)|D = 1] = \mathbb{E}[Y(1)|D = 1] - \mathbb{E}[Y(0)|D = 1].$$

This is the average treatment effect restricted to the population the treated individuals. When treatment D is randomly assigned conditional on confounding factors X, the ATE and ATET can be identified by regression or propensity score weighting methods. Our identification and estimation method rely on the combination of two methods to create orthogonal estimating equations for these parameters.

In observational studies, the potential treatments are endogenous, i.e. jointly determined with the outcome variable. In such cases, causal effects may be identified with the use of a binary instrument Z that affects the treatment but is independent of the potential outcomes. An important parameter in this case is the local average treatment effect (LATE):

$$E[Y(1) - Y(0)|D(1) > D(0)].$$

The random variables $D(1)$ and $D(0)$ indicate the potential participation decisions under the instrument states 1 and 0. LATE is the average treatment effect for the subpopulation of compliers – those units that respond to the change in the instrument. Another parameter of interest is the local average treatment effect of the treated (LATET):

$$\mathbb{E}\left[Y(1) - Y(0)|D(1) > D(0), D = 1\right],$$

which is the average effect for the subpopulation of the treated compliers.

When the instrument Z is randomly assigned conditional on confounding factors X, the LATE and LATET can be identified by instrumental variables regression or propensity score weighting methods. Our identification and estimation method rely on the combination of two methods to create orthogonal estimating equations for these parameters.[1]

6.2 Estimation and Inference of Treatment effects.

We consider the estimation of the effect of an endogenous binary treatment, D, on an outcome variable, Y , in a setting with very many potential control variables. In the case of endogeneity, the presence of a binary instrumental variable, Z, is required for the estimation of the LATE and LATET.

When trying to estimate treatment effects, the researcher has to decide what conditioning variables to include. In the case of a non-randomly assigned treatment or instrumental variable, the researcher must select the conditioning variables so that the instrument or treatment is plausibly exogenous. Even in the case of random assignment, for a precise estimation of the policy variable selection of control variables is necessary to absorb residual variation, but overfitting should be avoided. For uniformly valid post-selection inference, "orthogonal"estimating equations as described above are they key to the efficient estimation and valid inference. We refer to Belloni, Chernozhukov, Fernandez-Val, and Hansen (2013) for details.

% R Implementation. The package contains the functions rlassoATE, rlassoATET, rlassoLATE, and rlassoLATE to estimate the corresponding treatment effects. All functions have as arguments the outcome variable y, the treatment variable d, and the conditioning variables x. The functions rlassoLATE, and rlassoLATE have additionally the argument z for the binary instrumental variable. For the calculation of the standard errors bootstrap methods are implemented, with options to use Bayes, normal, or wild bootstrap. The number of repetitions can be specified with the argument nRep and the default is set to 500. By default no bootstrap standard errors are provided (bootstrap="none"). For the functions the logicals intercept and post can be specified to include an intercept and to do Post-Lasso at the selection steps. The family of treatment functions returns an object of class rlassoTE for which the methods print, summary, and confint are available.

6.3 Application: 401(k) plan participation.

Though it is clear that 401(k) plans are widely used as vehicles for retirement saving, their effect on assets is less clear. The key problem in determining the effect of participation in 401(k) plans on accumulated assets is saver heterogeneity coupled with nonrandom selection into participation states. In particular, it is generally recognized that some people have a higher preference for saving than others. Thus, it seems likely that those individuals with the highest unobserved preference for saving would be most likely to choose to participate in tax-advantaged retirement savings plans and would also have higher savings in other assets than individuals with lower unobserved saving propensity. This implies that conventional estimates that do not allow for saver heterogeneity and selection of the participation state will be biased upward, tending to overstate the actual savings effects of 401(k) and IRA participation.

Again, we start first with the data preparation:

[1] It turns out that the orthogonal estimating equations are the same as doubly robust estimating equations, but emphasizing the name "doubly robust" could be confusing in the present context.

```
using CSV
url = "https://raw.githubusercontent.com/d2cml-ai/HDMjl.jl/prueba/data/7_.csv"
data = DataFrame(CSV.read(download(url), DataFrame));
n, p = size(data);
p1 = 20;
X = data[:,2:end]
Y = data[:,1];
```

Now we can compute the estimates of the target treatment effect parameters. For ATE and ATET we report the the effect of eligibility for 401(k)

```
rls_eff = HDMjl.rlassoEffects(X, Y, index = [1:p1;]);
```

For LATE and LATET we estimate the effect of 401(k) participation (d) with plan eligibility (z) as instrument.

The results are summarized into a table, which is then processed using xtable to produce the latex output:

Finally, we estimate a model including all interaction effects:

7 The Lasso Methods for Discovery of Significant Causes amongst Many Potential Causes, with Many Controls

Here we consider the model

$$\underbrace{Y_i}_{\text{Outcome}} = \underbrace{\sum_{l=1}^{p_1} D_{il}\alpha_\ell}_{\text{Causes}} + \underbrace{\sum_{j=1}^{p_2} W_{ij}\beta_j}_{\text{Controls}} + \underbrace{\epsilon_i}_{\text{Noise}}$$

where the number of potential causes p_1 could be very large and the number of controls p_2 could also be very large. The causes are randomly assigned conditional on controls. Under approximate sparsity of $\alpha = (\alpha_l)_{l_1}^{p1}$ and $\beta = (\beta_l)_{l=1}^{p2}$, we can use Lasso-based method of Belloni, Chernozhukov, and Kato (2014) for estimating $(\alpha_l)_{l=1}^{p1}$ and constructing a joint confidence band on $(\alpha_l)_{l=1}^{p1}$ and then checking which αl's are significantly different from zero. The approach is based on buliding orthogonal estimating equations for each of $(\alpha_l)_{l=1}^{p_1}$, and can be interpreted as doing Frisch-Waugh procedure for each coefficient of interest, where we do partialling out via Lasso or OLS-after-Lasso.

This procedure is implemented in the `Julia` package `hdm`. Here is an example in which $n = 100$, $p_1 = 20$, and $p_2 = 20$, so that total number of regressors is $p = p_1 + p_2 = 40$. In this example $\alpha_1 = 5$ and $\beta_1 = 5$, i.e. there is only one true cause D_{i1}, among the large number of causes, $D_{i1}, ..., D_{i20}$, and only one true control W_{i1}. This example is made super-simple for clarity sake. The Belloni, Chernozhukov, and Kato (2014) procedure, implemented by rlassoEffects function in Julia package HDMjl.