HIGH-DIMENSIONAL METRICS IN R

VICTOR CHERNOZHUKOV, CHRISTIAN HANSEN, MARTIN SPINDLER

Abstract. The package High-dimensional Metrics (`hdm`) is an evolving collection of statistical methods for estimation and quantification of uncertainty in high-dimensional approximately sparse models. It focuses on providing confidence intervals and significance testing for (possibly many) low-dimensional subcomponents of the high-dimensional parameter vector. Efficient estimators and uniformly valid confidence intervals for regression coefficients on target variables (e.g., treatment or policy variable) in a high-dimensional approximately sparse regression model, for average treatment effect (ATE) and average treatment effect for the treated (ATET), as well for extensions of these parameters to the endogenous setting are provided. Theory grounded, data-driven methods for selecting the penalization parameter in Lasso regressions under heteroscedastic and non-Gaussian errors are implemented. Moreover, joint/ simultaneous confidence intervals for regression coefficients of a high-dimensional sparse regression are implemented, including a joint significance test for Lasso regression. Data sets which have been used in the literature and might be useful for classroom demonstration and for testing new estimators are included. R and the package `hdm` are open-source software projects and can be freely downloaded from CRAN: `http://cran.r-project.org`.

Contents

1. Introduction 2
2. How to get started 3
3. Prediction using Approximate Sparsity 4
3.1. Prediction in Linear Models using Approximate Sparsity 4
3.2. A Joint Significance Test for Lasso Regression 6
R implementation 6
Example 7
4. Inference on Target Regression Coefficients 9
4.1. Intuition for the Orthogonality Principle in Linear Models via Partialling Out 10
4.2. Inference: Confidence Intervals and Significance Testing 12
4.3. Application: the effect of gender on wage 14
4.4. Application: Estimation of the treatment effect in a linear model with many confounding factors 16
5. Instrumental Variable Estimation in a High-Dimensional Setting 18
5.1. Estimation and Inference 18
R Implementation 19
5.2. Application: Economic Development and Institutions 19
5.3. Application: Impact of Eminent Domain Decisions on Economic Outcomes 21
6. Inference on Treatment Effects in a High-Dimensional Setting 23

Version: May 17, 2019.

6.1. Treatment Effects Parameters – a short Introduction 23
6.2. Estimation and Inference of Treatment effects 24
R Implementation 24
6.3. Application: 401(k) plan participation 25
7. The Lasso Methods for Discovery of Significant Causes amongst Many Potential Causes, with Many Controls 28
8. Methods for Valid Simultaneous Inference in High-Dimensional Models 29
9. Conclusion 31
Appendix A. Data Sets 32
A.1. Pension Data 32
A.2. Growth Data 32
A.3. Institutions and Economic Development – Data on Settler Mortality 33
A.4. Data on Eminent Domain 33
A.5. BLP data 33
A.6. CPS data 33
References 35

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 semiparametric models with nonparametric 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 R package `hdm` 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.

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 prespecified 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 **?**. 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

R is an open source software project and can be freely downloaded from the CRAN website along with its associated documentation. The R package `hdm` can be downloaded from `cran.r-project.org`. To install the hdm package from R we simply type,

```
install.packages("hdm")
```

The most current version of the package (development version) is maintained at R-Forge and can installed by

```
install.packages("hdm", repos = "http://R-Forge.R-project.org")
```

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 `hdm` package is installed, it can be loaded to the current R session by the command,

```
library(hdm)
```

Online help is available in two ways. For example, you can type:

```
help(package = "hdm")
help(rlasso)
```

The former command gives an overview over the available commands in the package, and the latter gives detailed information about a specific command.
More generally one can initiate a web-browser help session with the command,

```
help.start()
```

and navigate as desired. The browser approach is better adapted to exploratory inquiries, while the command line approach is better suited to confirmatory ones.
A valuable feature of R help files is that the examples used to illustrate commands are executable, so they can be pasted into an R session or run as a group with a command like,

```
example(rlasso)
```

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

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

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 heteroscedastic (**?**).
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,

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

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

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

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

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

where $||\beta||_1 = \sum_{j=1}^p |\beta_j|$, $\hat{\Psi} = \mathrm{diag}(\hat{\psi}_1, \ldots, \hat{\psi}_p)$ is a diagonal matrix consisting of penalty loadings, and $\mathbb{E}_n$ 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 heteroscedasticity 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,

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

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 **?**.

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 $1 - \gamma - o(1)$,

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

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 $\hat{\psi}_j = \sqrt{\mathbb{E}_n x_{i,j}^2}$, 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 case the penalty level is calculated as

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

where

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

where $X = [x_1, \ldots, x_n]'$ 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 **?**.

In the case of heteroscedasticity, the loadings are set to $\hat{\psi}_j = \sqrt{\mathbb{E}_n[x_{ij}^2 \hat{\varepsilon}_i^2]}$, where $\hat{\varepsilon}_i$ are preliminary estimates of the errors. The penalty level can be X-independent (**?**):

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

or it can be X-dependent and estimated by a multiplier bootstrap procedure (**?**):

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

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

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

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 (**?**) (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

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

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

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

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:

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

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.

R implementation. The function `rlasso` implements Lasso and post-Lasso, where the prefix "r" signifies that these are theoretically rigorous versions of Lasso and post-Lasso. The default option is post-Lasso, `post=TRUE`. This function returns an object of S3 class `rlasso` for which methods like `predict`, `print`, `summary` are provided.

`lassoShooting.fit` is the computational algorithm that underlies the estimation procedure, which implements a version of the Shooting Lasso Algorithm (**?**). The user has several options for choosing the non-default options. Specifically, the user can decide if an unpenalized `intercept` should be included (`TRUE` by default). The option `penalty` of the function `rlasso` allows different choices for the penalization parameter and loadings. It allows for homoscedastic or heteroscedastic errors with default

`homoscedastic = FALSE`. Moreover, the dependence structure of the design matrix might be taken into consideration for calculation of the penalization parameter with `X.dependent.lambda = TRUE`. In combination with these options, the option `lambda.start` allows the user to set a starting value for λ for the different algorithms. Moreover, the user can provide her own fixed value for the penalty level – instead of the data-driven methods discussed above – by setting `homoscedastic = "none"` and supplying the value via `lambda.start`.
The constants c and γ from above can be set in the option `penalty`. The quantities $\hat{\varepsilon}$, $\hat{\Psi}$, $\hat{\sigma}$ are calculated in a iterative manner. The maximum number of iterations and the tolerance when the algorithms should stop can be set with `control`.
The method `summary` of `rlasso`-objects displays additionally for model diagnosis the R^2 value, the adjusted R^2 with degrees of freedom equal to the number of selected parameters, and the sup-score statistic for joint significance – described above – with corresponding p-value.

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

```
set.seed(12345)
n = 100  #sample size
p = 100  # number of variables
s = 3  # nubmer of variables with non-zero coefficients
X = matrix(rnorm(n * p), ncol = p)
beta = c(rep(5, s), rep(0, p - s))
Y = X %*% beta + rnorm(n)
```

Next we estimate the model, print the results, and make in-sample and out-of sample predictions. We can use methods `print` and `summarize` to print the results, where the option `all` can be set to `FALSE` to limit the print only to the non-zero coefficients.

```
lasso.reg = rlasso(Y ~ X, post = FALSE)  # use lasso, not-Post-lasso
# lasso.reg = rlasso(X, Y, post=FALSE)
sum.lasso <- summary(lasso.reg, all = FALSE)  # can also do print(lasso.reg, all=FALSE)
##
## Call:
## rlasso.formula(formula = Y ~ X, post = FALSE)
##
## Post-Lasso Estimation:  FALSE
##
## Total number of variables: 100
## Number of selected variables: 11
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.09008 -0.45801 -0.01237  0.50291  2.25098
##
```

```
##             Estimate
## (Intercept)    0.057
## 1              4.771
## 2              4.693
## 3              4.766
## 13            -0.045
## 15            -0.047
## 16            -0.005
## 19            -0.092
## 22            -0.027
## 40            -0.011
## 61             0.114
## 100           -0.025
##
## Residual standard error: 0.8039
## Multiple R-squared:  0.9913
## Adjusted R-squared:  0.9902
## Joint significance test:
##  the sup score statistic for joint significance test is 64.02 with a p-value of     0
yhat.lasso = predict(lasso.reg)  #in-sample prediction
Xnew = matrix(rnorm(n * p), ncol = p)  # new X
Ynew = Xnew %*% beta + rnorm(n)  #new Y
yhat.lasso.new = predict(lasso.reg, newdata = Xnew)  #out-of-sample prediction

post.lasso.reg = rlasso(Y ~ X, post = TRUE)  #now use post-lasso
print(post.lasso.reg, all = FALSE)  # or use  summary(post.lasso.reg, all=FALSE)
##
## Call:
## rlasso.formula(formula = Y ~ X, post = TRUE)
##
## (Intercept)           1           2           3
##      0.0341      4.9241      4.8579      4.9644
yhat.postlasso = predict(post.lasso.reg)  #in-sample prediction
yhat.postlasso.new = predict(post.lasso.reg, newdata = Xnew)  #out-of-sample prediction
MAE <- apply(cbind(abs(Ynew - yhat.lasso.new), abs(Ynew - yhat.postlasso.new)), 2,
    mean)
names(MAE) <- c("lasso MAE", "Post-lasso MAE")
print(MAE, digits = 2)  # MAE for Lasso and Post-Lasso
##      lasso MAE Post-lasso MAE
##           0.91           0.79
```

In the example above the sup-score statistic for overall significance is 64.02 with a pvalue of 0. This means that the null hypothesis is rejected on level $\alpha = 0.05$ and the model seems to have explanatory power.

4. Inference on Target Regression Coefficients

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

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

Here d_i is a target regressor such as treatment, policy or other variable whose regression coefficient α_0 we would like to learn (**?**). 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 $\mathsf{a}(\beta_0) > 1$. 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 $\mathsf{a}(\pi_0^d) > 1$.

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 $\hat{\alpha}$ 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 **?**.
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:

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

The score associated to this equation is:

$$\psi(w_i, \alpha, \eta) = (y_i - x_i'\pi^y) - \alpha(d_i - x_i'\pi^d))(d_i - x_i'\pi^d), \quad \eta = (\pi^{y'}, \pi^{d'})',$$

$$\psi(w_i, \alpha_0, \eta_0) = (\rho_i^y - \alpha\rho_i^d)\rho_i^d, \quad \eta_0 = (\pi_0^{y'}, \pi_0^{d'}).$$

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

```
set.seed(1)
n = 5000
p = 20
X = matrix(rnorm(n * p), ncol = p)
colnames(X) = c("d", paste("x", 1:19, sep = ""))
xnames = colnames(X)[-1]
beta = rep(1, 20)
y = X %*% beta + rnorm(n)
dat = data.frame(y = y, X)
```

We can estimate α_0 by running full least squares:

```
# full fit
fmla = as.formula(paste("y ~ ", paste(colnames(X), collapse = "+")))
full.fit = lm(fmla, data = dat)
summary(full.fit)$coef["d", 1:2]
```

```
##   Estimate Std. Error
## 0.97807455 0.01371225
```

Another way to estimate α_0 is to first partial out the x-variables from y_i and d_i, and run least squares on the residuals:

```
fmla.y = as.formula(paste("y ~ ", paste(xnames, collapse = "+")))
fmla.d = as.formula(paste("d ~ ", paste(xnames, collapse = "+")))
# partial fit via ols
rY = lm(fmla.y, data = dat)$res
rD = lm(fmla.d, data = dat)$res
partial.fit.ls = lm(rY ~ rD)
summary(partial.fit.ls)$coef["rD", 1:2]
##   Estimate Std. Error
## 0.97807455 0.01368616
```

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:

```
# partial fit via post-lasso
rY = rlasso(fmla.y, data = dat)$res
rD = rlasso(fmla.d, data = dat)$res
partial.fit.postlasso = lm(rY ~ rD)
summary(partial.fit.postlasso)$coef["rD", 1:2]
##   Estimate Std. Error
## 0.97273870 0.01368677
```

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 = rlassoEffect(X[, -1], y, X[, 1], method = "partialling out")
summary(Eff)$coef[, 1:2]
##  Estimate. Std. Error
## 0.97273870 0.01368677
```

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 = rlassoEffect(X[, -1], y, X[, 1], method = "double selection")
summary(Eff)$coef[, 1:2]
##  Estimate. Std. Error
## 0.97807455 0.01415624
```

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 `summary`, `print`, `confint`, and `plot` 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 `?` and for a more general class of models in `?`. 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 (position 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.

```
set.seed(1)
n = 100  #sample size
p = 100  # number of variables
s = 3  # nubmer of non-zero variables
X = matrix(rnorm(n * p), ncol = p)
colnames(X) <- paste("X", 1:p, sep = "")
beta = c(rep(3, s), rep(0, p - s))
y = 1 + X %*% beta + rnorm(n)
data = data.frame(cbind(y, X))
colnames(data)[1] <- "y"
fm = paste("y ~", paste(colnames(X), collapse = "+"))
fm = as.formula(fm)
```

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

```
# lasso.effect = rlassoEffects(X, y, index=c(1,2,3,50))
lasso.effect = rlassoEffects(fm, I = ~X1 + X2 + X3 + X50, data = data)
print(lasso.effect)
##
```

```
## Call:
## rlassoEffects.formula(formula = fm, data = data, I = ~X1 + X2 +
##     X3 + X50)
##
## Coefficients:
##      X1       X2       X3      X50
## 2.94448  3.04127  2.97540  0.07196
```

```
summary(lasso.effect)
```

```
## [1] "Estimates and significance testing of the effect of target variables"
##     Estimate. Std. Error t value Pr(>|t|)
## X1    2.94448    0.08815  33.404   <2e-16 ***
## X2    3.04127    0.08389  36.253   <2e-16 ***
## X3    2.97540    0.07804  38.127   <2e-16 ***
## X50   0.07196    0.07765   0.927    0.354
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
confint(lasso.effect)
```

```
##           2.5 %    97.5 %
## X1   2.77171308 3.1172421
## X2   2.87685121 3.2056979
## X3   2.82244962 3.1283583
## X50 -0.08022708 0.2241377
```

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 **?**. Joint confidence intervals can be invoked by setting the option `joint` to `TRUE` in the method `confint` for objects of class `rlassoEffects`. We will also demonstrate the application of joint confidence intervals in an empirical application in the next section.

```
confint(lasso.effect, level = 0.95, joint = TRUE)
```

```
##          2.5 %    97.5 %
## X1   2.7279477 3.1610075
## X2   2.8371214 3.2454278
## X3   2.7833176 3.1674903
## X50 -0.1154509 0.2593615
```

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

```
plot(lasso.effect, main = "Confidence Intervals")
```

For logistic regression we provide the functions `rlassologit` and `rlassologitEffects`. Further information can be found in the help.

4.3. **Application: the effect of gender on wage.** In Labor 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.

```
library(hdm)
data(cps2012)
X <- model.matrix(~-1 + female + female:(widowed + divorced + separated + nevermarried +
    hsd08 + hsd911 + hsg + cg + ad + mw + so + we + exp1 + exp2 + exp3) + +(widowed +
    divorced + separated + nevermarried + hsd08 + hsd911 + hsg + cg + ad + mw + so +
    we + exp1 + exp2 + exp3)^2, data = cps2012)
dim(X)
## [1] 29217   136
X <- X[, which(apply(X, 2, var) != 0)]  # exclude all constant variables
dim(X)
## [1] 29217   116
index.gender <- grep("female", colnames(X))
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 <- rlassoEffects(x = X, y = y, index = index.gender)
summary(effects.female)
## [1] "Estimates and significance testing of the effect of target variables"
##                     Estimate. Std. Error t value Pr(>|t|)
## female              -0.154923   0.050162  -3.088 0.002012
## female:widowed       0.136095   0.090663   1.501 0.133325
## female:divorced      0.136939   0.022182   6.174 6.68e-10
## female:separated     0.023303   0.053212   0.438 0.661441
## female:nevermarried  0.186853   0.019942   9.370  < 2e-16
## female:hsd08         0.027810   0.120914   0.230 0.818092
## female:hsd911       -0.119335   0.051880  -2.300 0.021435
## female:hsg          -0.012890   0.019223  -0.671 0.502518
## female:cg            0.010139   0.018327   0.553 0.580114
## female:ad           -0.030464   0.021806  -1.397 0.162405
## female:mw           -0.001063   0.019192  -0.055 0.955811
```

```
## female:so           -0.008183   0.019357  -0.423 0.672468
## female:we           -0.004226   0.021168  -0.200 0.841760
## female:exp1          0.004935   0.007804   0.632 0.527139
## female:exp2         -0.159519   0.045300  -3.521 0.000429
## female:exp3          0.038451   0.007861   4.891 1.00e-06
##
## female              **
## female:widowed
## female:divorced     ***
## female:separated
## female:nevermarried ***
## female:hsd08
## female:hsd911       *
## female:hsg
## female:cg
## female:ad
## female:mw
## female:so
## female:we
## female:exp1
## female:exp2         ***
## female:exp3         ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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

```
joint.CI <- confint(effects.female, level = 0.95, joint = TRUE)
joint.CI
##                           2.5 %      97.5 %
## female              -0.29422452 -0.01562204
## female:widowed      -0.13367117  0.40586213
## female:divorced      0.07479695  0.19908182
## female:separated    -0.11671664  0.16332216
## female:nevermarried  0.12925782  0.24444915
## female:hsd08        -0.37450228  0.43012291
## female:hsd911       -0.26902488  0.03035480
## female:hsg          -0.06513949  0.03935993
## female:cg           -0.04168175  0.06195886
## female:ad           -0.09583693  0.03490944
```

```
## female:mw           -0.05470794  0.05258106
## female:so           -0.06272074  0.04635406
## female:we           -0.06562874  0.05717648
## female:exp1         -0.01649540  0.02636592
## female:exp2         -0.28414464 -0.03489402
## female:exp3          0.01684777  0.06005338
# plot(effects.female, joint=TRUE, level=0.95) # plot of the effects
```

This analysis allows a closer look how discrimination according to gender is related to other socio-economic variables.
As a side remark, the version 0.2 allows also now a formula interface for many functions including `rlassoEffects`. Hence, the analysis could also be done more compact as

```
effects.female <- rlassoEffects(lnw ~ female + female:(widowed + divorced + separated +
    nevermarried + hsd08 + hsd911 + hsg + cg + ad + mw + so + we + exp1 + exp2 +
    exp3) + (widowed + divorced + separated + nevermarried + hsd08 + hsd911 + hsg +
    cg + ad + mw + so + we + exp1 + exp2 + exp3)^2, data = cps2012, I = ~female +
    female:(widowed + divorced + separated + nevermarried + hsd08 + hsd911 + hsg +
        cg + ad + mw + so + we + exp1 + exp2 + exp3))
```

The one-sided option `I` gives the target variables for which inference is conducted.

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 **?** 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

```
data(GrowthData)
dim(GrowthData)
```

```
## [1] 90 63
y = GrowthData[, 1, drop = F]
d = GrowthData[, 3, drop = F]
X = as.matrix(GrowthData)[, -c(1, 2, 3)]
varnames = colnames(GrowthData)
```

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

```
xnames = varnames[-c(1, 2, 3)]  # names of X variables
dandxnames = varnames[-c(1, 2)]  # names of D and X variables
# create formulas by pasting names (this saves typing times)
fmla = as.formula(paste("Outcome ~ ", paste(dandxnames, collapse = "+")))
ls.effect = lm(fmla, data = GrowthData)
```

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

```
dX = as.matrix(cbind(d, X))
lasso.effect = rlassoEffect(x = X, y = y, d = d, method = "partialling out")
summary(lasso.effect)
## [1] "Estimates and significance testing of the effect of target variables"
##      Estimate. Std. Error t value Pr(>|t|)
## [1,]  -0.04981    0.01394  -3.574 0.000351 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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

```
dX = as.matrix(cbind(d, X))
doublesel.effect = rlassoEffect(x = X, y = y, d = d, method = "double selection")
summary(doublesel.effect)
## [1] "Estimates and significance testing of the effect of target variables"
##          Estimate. Std. Error t value Pr(>|t|)
## gdpsh465  -0.05001    0.01579  -3.167  0.00154 **
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We then collect results in a nice latex table:

```
library(xtable)
table = rbind(summary(ls.effect)$coef["gdpsh465", 1:2], summary(lasso.effect)$coef[,
    1:2], summary(doublesel.effect)$coef[, 1:2])
colnames(table) = c("Estimate", "Std. Error")  #names(summary(full.fit)£coef)[1:2]
rownames(table) = c("full reg via ols", "partial reg
```

```
via post-lasso ", "partial reg via double selection")
tab = xtable(table, digits = c(2, 2, 5))
```

```
tab
```

	Estimate	Std. Error
full reg via ols	-0.01	0.02989
partial reg via post-lasso	-0.05	0.01394
partial reg via double selection	-0.05	0.01579

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:

$$y_i = \alpha_0 d_i + \gamma_0 x_i' + \varepsilon_i, \tag{1}$$

$$d_i = z_i'\Pi + \beta_0 x_i' + v_i, \tag{2}$$

where $\mathbb{E}[\varepsilon_i(x_i', z_i')] = 0$, $\mathbb{E}[v_i(x_i', z_i')] = 0$, but $\mathbb{E}[\varepsilon_i v_i] \neq 0$ leading to endogeneity. In this setting d_i is a scalar endogenous variable of interest, z_i is a p_z-dimensional vector of instruments and x_i is a p_x-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 **?**, the extension with many instruments and many controls in **?**.

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 heteroscedasticity 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 α_0. We then perform inference on α_0 using $\hat{\alpha}$ and conventional heteroscedasticity robust standard errors.
In the case of the high-dimensional instrument z_i and high-dimensional x_i the algorithm described in **?** is adopted.

R 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=FALSE` 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 prepared 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 variables (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, **?** 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

```
data(AJR)
y = AJR$GDP
d = AJR$Exprop
z = AJR$logMort
```

```
x = model.matrix(~-1 + (Latitude + Latitude2 + Africa + Asia + Namer + Samer)^2,
    data = AJR)
dim(x)
## [1] 64 21
```

Then we estimate an IV model with selection on the X

```
# AJR.Xselect = rlassoIV(x=x, d=d, y=y, z=z, select.X=TRUE, select.Z=FALSE)
AJR.Xselect = rlassoIV(GDP ~ Exprop + (Latitude + Latitude2 + Africa + Asia + Namer +
    Samer)^2 | logMort + (Latitude + Latitude2 + Africa + Asia + Namer + Samer)^2,
    data = AJR, select.X = TRUE, select.Z = FALSE)
summary(AJR.Xselect)
## [1] "Estimation and significance testing of the effect of target variables in the IV regression mo
##        coeff.    se. t-value p-value
## Exprop 0.8450 0.2699   3.131 0.00174 **
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
confint(AJR.Xselect)
##            2.5 %   97.5 %
## Exprop 0.3159812 1.374072
```

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:

```
# parialling out by linear model
fmla.y = GDP ~ (Latitude + Latitude2 + Africa + Asia + Namer + Samer)^2
fmla.d = Exprop ~ (Latitude + Latitude2 + Africa + Asia + Namer + Samer)^2
fmla.z = logMort ~ (Latitude + Latitude2 + Africa + Asia + Namer + Samer)^2
rY = lm(fmla.y, data = AJR)$res
rD = lm(fmla.d, data = AJR)$res
rZ = lm(fmla.z, data = AJR)$res
# ivfit.lm = tsls(y=rY,d=rD, x=NULL, z=rZ, intercept=FALSE)
ivfit.lm = tsls(rY ~ rD | rZ, intercept = FALSE)
print(cbind(ivfit.lm$coef, ivfit.lm$se), digits = 3)
##    [,1] [,2]
## rD 1.27 1.73
```

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

```
# parialling out by lasso
rY = rlasso(fmla.y, data = AJR)$res
rD = rlasso(fmla.d, data = AJR)$res
rZ = rlasso(fmla.z, data = AJR)$res
# ivfit.lasso = tsls(y=rY,d=rD, x=NULL, z=rZ, intercept=FALSE)
ivfit.lasso = tsls(rY ~ rD | rZ, intercept = FALSE)
print(cbind(ivfit.lasso$coef, ivfit.lasso$se), digits = 3)
##     [,1] [,2]
## rD 0.845 0.27
```

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.

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 **?** and **?**.
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.

```
data(EminentDomain)
z <- as.matrix(EminentDomain$logGDP$z)
x <- as.matrix(EminentDomain$logGDP$x)
y <- EminentDomain$logGDP$y
d <- EminentDomain$logGDP$d
x <- x[, apply(x, 2, mean, na.rm = TRUE) > 0.05]  #
z <- z[, apply(z, 2, mean, na.rm = TRUE) > 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(y ~ cbind(d, x))
ED.2sls = tsls(y = y, d = d, x = x, z = z[, 1:2], intercept = FALSE)
```

Next, we estimate the model with selection on the instruments.

```
lasso.IV.Z = rlassoIV(x = x, d = d, y = y, z = z, select.X = FALSE, select.Z = TRUE)
# or lasso.IV.Z = rlassoIVselectZt(x=X, d=d, y=y, z=z)
summary(lasso.IV.Z)
## [1] "Estimates and significance testing of the effect of target variables in the IV regression mod
##    coeff.    se. t-value p-value
## d1 0.4146 0.2902   1.428   0.153
confint(lasso.IV.Z)
##         2.5 %    97.5 %
## d1 -0.1542764 0.9834796
```

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

```
lasso.IV.XZ = rlassoIV(x = x, d = d, y = y, z = z, select.X = TRUE, select.Z = TRUE)
summary(lasso.IV.XZ)
## Estimates and Significance Testing of the effect of target variables in the IV regression model
##      coeff.      se. t-value p-value
## d1 -0.02383  0.12851  -0.185   0.853
confint(lasso.IV.XZ)
##         2.5 %    97.5 %
## d1 -0.2757029 0.2280335
```

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

```
library(xtable)
table = matrix(0, 4, 2)
table[1, ] = summary(ED.ols)$coef[2, 1:2]
table[2, ] = cbind(ED.2sls$coef[1], ED.2sls$se[1])
table[3, ] = summary(lasso.IV.Z)[, 1:2]
## [1] "Estimates and significance testing of the effect of target variables in the IV regression mod
##    coeff.    se. t-value p-value
## d1 0.4146 0.2902   1.428   0.153
table[4, ] = summary(lasso.IV.XZ)[, 1:2]
## Estimates and Significance Testing of the effect of target variables in the IV regression model
```

```
##       coeff.       se. t-value p-value
## d1 -0.02383  0.12851  -0.185   0.853
colnames(table) = c("Estimate", "Std. Error")
rownames(table) = c("ols regression", "IV estimation ", "selection on Z", "selection on X and Z")
tab = xtable(table, digits = c(2, 2, 7))
```

```
tab
```

	Estimate	Std. Error
ols regression	0.01	0.0098659
IV estimation	-0.01	0.0337664
selection on Z	0.41	0.2902492
selection on X and Z	-0.02	0.1285065

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 **?**.

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, $i = 1, \ldots, n$. 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.[1]

[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.

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):

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

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):

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

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.

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 **?** 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:

```
data(pension)
y = pension$tw
d = pension$p401
z = pension$e401
X = pension[, c("i2", "i3", "i4", "i5", "i6", "i7", "a2", "a3", "a4", "a5", "fsize",
    "hs", "smcol", "col", "marr", "twoearn", "db", "pira", "hown")]  # simple model
xvar = c("i2", "i3", "i4", "i5", "i6", "i7", "a2", "a3", "a4", "a5", "fsize", "hs",
    "smcol", "col", "marr", "twoearn", "db", "pira", "hown")
xpart = paste(xvar, collapse = "+")
form <- as.formula(paste("tw ~ ", paste(c("p401", xvar), collapse = "+"), "|", paste(xvar,
    collapse = "+")))
formZ <- as.formula(paste("tw ~ ", paste(c("p401", xvar), collapse = "+"), "|", paste(c("e401",
    xvar), collapse = "+")))
```

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).

```
# pension.ate = rlassoATE(X,d,y)
pension.ate = rlassoATE(form, data = pension)
summary(pension.ate)
## Estimation and significance testing of the treatment effect
## Type: ATE
## Bootstrap: not applicable
##    coeff.   se. t-value  p-value
## TE  10180  1931   5.273 1.34e-07 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
# pension.atet = rlassoATET(X,d,y)
pension.atet = rlassoATET(form, data = pension)
summary(pension.atet)
```

```
## Estimation and significance testing of the treatment effect
## Type: ATET
## Bootstrap: not applicable
##    coeff.   se. t-value p-value
## TE  12628  2944   4.289 1.8e-05 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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

```
pension.late = rlassoLATE(X, d, y, z, always_takers = FALSE)
# pension.late = rlassoLATE(formZ, data=pension, always_takers = FALSE)
summary(pension.late)
## Estimation and significance testing of the treatment effect
## Type: LATE
## Bootstrap: not applicable
##    coeff.   se. t-value p-value
## TE  12250  2745   4.463 8.1e-06 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
pension.latet = rlassoLATET(X, d, y, z, always_takers = FALSE)
# pension.latet = rlassoLATET(formZ, data=pension, always_takers = FALSE)
summary(pension.latet)
## Estimation and significance testing of the treatment effect
## Type: LATET
## Bootstrap: not applicable
##    coeff.   se. t-value  p-value
## TE  15323  3645   4.204 2.63e-05 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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

```
library(xtable)
table = matrix(0, 4, 2)
table[1, ] = summary(pension.ate)[, 1:2]
## Estimation and significance testing of the treatment effect
## Type: ATE
```

```
## Bootstrap: not applicable
##    coeff.   se. t-value  p-value
## TE  10180  1931   5.273 1.34e-07 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
table[2, ] = summary(pension.atet)[, 1:2]
## Estimation and significance testing of the treatment effect
## Type: ATET
## Bootstrap: not applicable
##    coeff.   se. t-value p-value
## TE  12628  2944   4.289 1.8e-05 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
table[3, ] = summary(pension.late)[, 1:2]
## Estimation and significance testing of the treatment effect
## Type: LATE
## Bootstrap: not applicable
##    coeff.   se. t-value p-value
## TE  12250  2745   4.463 8.1e-06 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
table[4, ] = summary(pension.latet)[, 1:2]
## Estimation and significance testing of the treatment effect
## Type: LATET
## Bootstrap: not applicable
##    coeff.   se. t-value  p-value
## TE  15323  3645   4.204 2.63e-05 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
colnames(table) = c("Estimate", "Std. Error")
rownames(table) = c("ATE", "ATET ", "LATE", "LATET")
tab = xtable(table, digits = c(2, 2, 2))
```

Finally, we estimate a model including all interaction effects:

```
# generate all interactions of X's
xvar2 = paste("(", paste(xvar, collapse = "+"), ")^2", sep = "")
```

	Estimate	Std. Error
ATE	10180.09	1930.68
ATET	12628.46	2944.43
LATE	12249.51	2744.92
LATET	15323.18	3645.28

```
# ATE and ATE with interactions
forminteract = formula(paste("tw ~", xvar2, " + p401", "|", xvar2, sep = ""))
# LATE and LATET with interactions
formZinteract = formula(paste("tw ~", xvar2, " + p401", "|", xvar2, " + e401", sep = ""))
```

```
# pension.ate= rlassoATE(forminteract, data = pension) summary(pension.ate)
# pension.atet= rlassoATET(forminteract, data = pension) summary(pension.atet)
# pension.late= rlassoLATE(formZinteract, data = pension, always_takers = FALSE)
# summary(pension.late) pension.latet= rlassoLATET(formZinteract, data = pension,
# always_takers = FALSE) summary(pension.latet) table= matrix(0, 4, 2) table[1,]=
# summary(pension.ate)[,1:2] table[2,]= summary(pension.atet)[,1:2] table[3,]=
# summary(pension.late)[,1:2] table[4,]= summary(pension.latet)[,1:2]
# colnames(table)= c('Estimate', 'Std. Error') rownames(table)= c('ATE', 'ATET ',
# 'LATE', 'LATET') tab= xtable(table, digits=c(2, 2,2))
```

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 $\alpha = (\alpha_l)_{l=1}^{p_1}$ and $\beta = (\beta_l)_{l=1}^{p_2}$, we can use Lasso-based method of **?** for estimating $(\alpha_l)_{l=1}^{p_1}$ and constructing a joint confidence band on $(\alpha_l)_{l=1}^{p_1}$ and then checking which α_l's are significantly different from zero. The approach is based on building 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 R 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 **?** procedure, implemented by `rlasso.effects` command in R package `hdm`.

```
# library(hdm) library(stats)
set.seed(1)
n = 100
p1 = 20
p2 = 20
D = matrix(rnorm(n * p1), n, p1)  # Causes
W = matrix(rnorm(n * p2), n, p2)  # Controls
X = cbind(D, W)  # Regressors
Y = D[, 1] * 5 + W[, 1] * 5 + rnorm(n)  #Outcome
confint(rlassoEffects(X, Y, index = c(1:p1)), joint = TRUE)
##          2.5 %     97.5 %
## V1   4.5145877 5.21430498
## V2  -0.3142909 0.30494650
## V3  -0.3524109 0.18678880
## V4  -0.2542430 0.28738914
## V5  -0.2765802 0.27627177
## V6  -0.3214676 0.29422684
## V7  -0.2262507 0.30094168
## V8  -0.0473541 0.47366372
## V9  -0.1865636 0.39023520
## V10 -0.2372356 0.26411185
## V11 -0.3147091 0.20945872
## V12 -0.3091905 0.26572176
## V13 -0.1741550 0.37682465
## V14 -0.3235734 0.38543162
## V15 -0.3219763 0.31312486
## V16 -0.2649505 0.33100700
## V17 -0.1792080 0.41696169
## V18 -0.3693247 0.04695928
## V19 -0.1073109 0.39368776
## V20 -0.2157182 0.25543839
# BCK Joint Confidence Band for Reg Coefficients 1 to 20
```

As you can see the procedure correctly tells that only the first cause D_{i1}, among the large number of causes, $D_{i1}, ..., D_{i20}$, is a statistically significant cause of Y (see the confidence interval for variable V1).

8. Methods for Valid Simultaneous Inference in High-Dimensional Models

The `hdm` packages provides methods for multiple testing adjustment as described in **?**. Various methods to conduct valid simultaneous inference on a set of target coefficients are implemented in the S3 method `p_adjust()`. For example, consider the problem of simultaneously testing 80 target coefficients in a

simulated data example. For instance, it is possible to take the correlation structure among covariates into account by using the stepdown procedure of **?**.

```
library(mvtnorm)
set.seed(1)
n = 100
p = 80
s = 9
covar = toeplitz(0.9^(0:(p - 1)))
diag(covar) = rep(1, p)
mu = rep(0, p)
X = mvtnorm::rmvnorm(n = n, mean = mu, sigma = covar)  # Regressors
beta = c(s:1, rep(0, p - s))
Y = X %*% beta + rnorm(n, sd = 5)  #Outcome
# Estimate rlassoEffects
rl = rlassoEffects(X, Y, index = c(1:p))

# unadjusted
p.unadj = p_adjust(rl, method = "none")
# Number of rejections at a prespecified significance level
sum(p.unadj[, 2] < 0.05)
## [1] 12
```

```
# Romano-Wolf Stepdown Correction
p.rw = p_adjust(rl, method = "RW", B = 1000)
# Number of rejections at a prespecified significance level (5%)
sum(p.rw[, 2] < 0.05)
## [1] 6
```

More methods to adjust for multiple testing as provided by the `p.adjust()` command from the R base package are available, for instance the Bonferroni adjustment or FDR-controlling methods like the Benjamini-Hochberg correction.

```
# Adjust with Bonferroni correction
p.bonf = p_adjust(rl, method = "bonferroni")
# Number of rejections at a prespecified significance level
sum(p.bonf[, 2] < 0.05)
## [1] 5
# Romano-Wolf Stepdown Correction
p.bh = p_adjust(rl, method = "BH")
# Number of rejections at a prespecified significance level
sum(p.bh[, 2] < 0.05)
## [1] 10
```

9. Conclusion

We have provided an introduction to some of the capabilities of the R package `hdm`. Inevitably, new applications will demand new features and, as the project is in its initial phase, unforeseen bugs will show up. In either case comments and suggestions of users are highly appreciated. We shall update the documentation and the package periodically. The most current version of the R package and its accompanying vignette will be made available at the homepage of the maintainer and `cran.r-project.org`. See the R command `vignette()` for details on how to find and view vignettes from within R .

APPENDIX A. DATA SETS

In this section we describe briefly the data sets which are contained in the package and used afterwards. They might also be of general interest either for illustrating methods or for classroom presentation.

A.1. **Pension Data.** In the United States 401(k) plans were introduced to increase private individual saving for retirement. They allow the individual to deduct contributions from taxable income and allow tax-free accrual of interest on assets held within the plan (within an account). Employers provide 401(k) plans, and employers may also match a certain percentage of an employee's contribution. Because 401(k) plans are provided by employers, only workers in firms offering plans are eligible for participation. This data set contains individual level information about 401(k) participation and socio-economic characteristics.
The data set can be loaded with

```
data(pension)
```

A description of the variables and further references are given in **?** and at the help page, for this example called by

```
help(pension)
```

The sample is drawn from the 1991 Survey of Income and Program Participation (SIPP) and consists of 9,915 observations. The observational units are household reference persons aged 25-64 and spouse if present. Households are included in the sample if at least one person is employed and no one is self-employed. All dollar amounts are in 1991 dollars. The 1991 SIPP reports household financial data across a range of asset categories. These data include a variable for whether a person works for a firm that offers a 401(k) plan. Households in which a member works for such a firm are classified as eligible for a 401(k). In addition, the survey also records the amount of 401(k) assets. Households with a positive 401(k) balance are classified as participants, and eligible households with a zero balance are considered nonparticipants. Available measures of wealth in the 1991 SIPP are total wealth, net financial assets, and net non-401(k) financial assets. Net non-401(k) assets are defined as the sum of checking accounts, U.S. saving bonds, other interest-earning accounts in banks and other financial institutions, other interest-earning assets (such as bonds held personally), stocks and mutual funds less non-mortgage debt, and IRA balances. Net financial assets are net non-401(k) financial assets plus 401(k) balances, and total wealth is net financial assets plus housing equity and the value of business, property, and motor vehicles.

A.2. **Growth Data.** Understanding what drives economic growth, measured in GDP, is a central question of macroeconomics. A well-known data set with information about GDP growth for many countries over a long period was collected by **?**. This data set can be loaded by

```
data(GrowthData)
```

This data set contains the national growth rates in GDP per capita (Outcome) for many countries with additional covariates. A very important covariate is gdpsh465, which is the initial level of per-capita GDP. For further information we refer to the help page and the references herein, in particular the online descriptions of the data set.

A.3. **Institutions and Economic Development – Data on Settler Mortality.** This data set was collected by **?** to analyse the effect of institutions on economic development. The data can be loaded with

```
data(AJR)
```

The data set contains measurements of GDP, settler morality, an index measuring protection against expropriation risk and geographic information (latitude and continent dummies). There are 64 observations on 11 variables.

A.4. **Data on Eminent Domain.** Eminent domain refers to the government's taking of private property. This data set was collected to analyse the effect of the number of pro-plaintiff appellate takings decisions on economic outcome variables such as house price indices.
The data set is loaded into R by

```
data(EminentDomain)
```

The data set consists of four "sub data sets"which have the following structure:

- y: outcome variable, a house price index or a local GDP index,
- d: the treatment variable, represents the number of pro-plaintiff appellate takings decisions in federal circuit court c and year t
- x: exogenous control variables that include a dummy variable for whether there were relevant cases in that circuit-year, the number of takings appellate decisions, and controls for the distribution of characteristics of federal circuit court judges in a given circuit-year
- z: instrumental variables, here characteristics of judges serving on federal appellate panels

The four data sets differ mainly in the dependent variable, which can be repeat-sales FHFA/OFHEO house price index for metro (FHFA) and non-metro (NM) areas , the Case-Shiller home price index (CS), and state-level GDP from the Bureau of Economic Analysis.

A.5. **BLP data.** This data set was analyzed in the seminal contribution of **?** and stems from annual issues of the Automotive News Market Data Book. The data set includes information on all models marketed during the the period beginning 1971 and ending in 1990 containing 2217 model/years from 997 distinct models. A detailed description is given in **?**, p. 868–871. The function `constructIV` constructs instrumental variables along the lines described and used in **?**. The data set is loaded by

```
data(BLP)
```

It contains information on the price (in logarithm), the market share, and car characteristics like miles per gallon, miles per dollar, horse power per weight, space and air conditioning.

A.6. **CPS data.** The CPS is a monthly U.S. household survey conducted jointly by the U.S. Census Bureau and the Bureau of Labor Statistics. The data were collected for the year 2012. The sample comprises white non-Hispanic, ages 25-54, working full time full year (35+ hours per week at least 50 weeks), exclude living in group quarters, self-employed, military, agricultural, and private household sector, allocated earning, inconsistent report on earnings and employment, missing data. It can be inspected with the command

```
data(cps2012)
```

References