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 estimating and drawing inferences in high-dimensional approximately sparse models. This vignette offers a brief introduction and a tutorial to the implemented methods. 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

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 readily available high-dimensional data which have many measured characteristics available per

Version: January 14, 2016.

individual observation as in, for example, large survey data sets, 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 of 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 like medicine, biology, sociology or psychology to mention a few.

The methods which are implemented in this package are distinctive from already available methods in other packages in mainly the following three major ways:

1) First, we provide efficient estimators and uniformaly 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. 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.

2) Second, we provide theoretically grounded, data-driven choice of the penalty level λ in the Lasso regressions is both theoretical grounded and data-driven. Because of this we call it "rigorous"LASSO (=`rlasso`). The prefix **r** in function names should underscore this. In high-dimensions setting cross-validation is very popular, but it lacks a theoretical justification and some theoretical proposals are often not feasible.

3) Third, we provide a version of Lasso regressions that expressly handle and allow for non-Gaussian and heteroscedastic errors.

In this vignette, we first show how to get started with package. Then we introduce briefly the data sets which are contained in the package and used later for illustration. Next the functions for LASSO and Post-LASSO estimation under heteroscedastic and non-Gaussian errors are presented. They are the core for the further applications. Next, different econometric models in a high-dimensional setting are introduced and illustrated by an empirical application.

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 needs to be made accessible to the current R session by the command,

```
library(hdm)
```

Online help is available in two ways. If you know precisely the command you are looking for, e.g. in order to check the details, try:

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

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(lasso)
```

3. 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 for other researchers either for illustrating methods or for class-room presentation.

3.1. **Pension data.** In the United States 401(k) plans were introduced to increase 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 information about 401(k) participation and socio-economic characteristics of the individuals.

The data set can be loaded with

```
data(pension)
```

A description of the variables and further references are given at the help page

```
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.

3.2. **Growth Data.** The question what drives Economic Growth, measured in GDP, is a central question of Economics. A famous data set with information about GDP growth for many countries and long period was collected by Barro and Lee. This data set is also provided in the data set and can be loaded by

```
data(GrowthData)
```

This data sets 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.

3.3. **Institutions and Economic Development – Data on settler mortality.** This data set was introduced by paper Acemoglu, Johnson, and Robinson (2001) to analyse the effect of institutions on economic development. The data is contained in the package and can be accessed with

```
data(AJR)
```

The data set contains GDP, Settler Morality, an index measuring protection against expropriation risk and Geographic Information (Latitude and Continent dummies). In total 11 variables and 64 observations.

3.4. **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 number of pro-plaintiff appellate takings decisions on economic relevant outcome variables like house prices, measured by some index.

The data set is loaded into R by

```
data(EminentDomain)
```

The data set consists of four groups of variables:

- y: outcome variable, a house price 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

4. Prediction using Approximate Sparsity

4.1. **Prediction in Linear Models using Approximate Sparsity.** We consider linear high dimensional approximate sparse 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 the regressors, using ℓ_1-based penalization methods.

The model reads:

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

where y_i are observations of the response variable, $x_i = (x_{i,j}, \ldots, x_{i,p})$'s are observations of $p-$dimensional fixed regressors, and ε_i's are iid, mean-zero centered disturbances, where possibly $p \gg n$. An important point is that the errors ε_i are *not* restricted to be Gaussian or homoscedastic. This means that we allow for both non-Gaussian and heteroscedastic errors.

The model can be exactly sparse, namely

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

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

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

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

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

In order to get theoretically valid results in these settings, 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 $||\beta||_1 = \sum_{j=1}^p |\beta_j|$ and $\hat{\Psi} = \text{diag}(\hat{\psi}_1, \ldots, \hat{\psi}_p)$ is a diagonal matrix consisting of regressor dependent 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 the heteroskedastiticy 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 the data, but it also shrinks the fitted coefficients towards zero, causing a potentially significant bias. In order to remove some of this bias, let us consider the Post-LASSO estimator that applies ordinary least squares regression 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

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

where $\hat{T}^C = \{1, \ldots, p\} \setminus \hat{T}$. In words, the estimator is ordinary least squares applied to the data after removing the regressors that were not selected by LASSO.

A crucial point for the performance of LASSO is the choice of the penalization parameter λ. In high-dimensions setting cross-validation is very popular but lacking a theoretical justification, and other proposals are often not feasible. The choice of the penalization parameter λ in the Lasso regressions in this package is theoretical grounded and feasible. Therefore we call the resulting procedure the "rigorous"Lasso and hence add as 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. The 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

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

where Φ denotes the cumulative standard normal distribution, $\hat{\sigma}$ is a preliminary estimate of $\sigma = \sqrt{\mathbb{E}\varepsilon^2}$, and $c > 1$ is a theoretical constant , which is set to $c = 1.1$ by default, and γ is the probability level, which is set to $\gamma = .01$ by default. [1] In the X-dependent case the penalty level is calculated as

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

where

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

[1]The probability is probability of mistakenly not removing some of the X's, when all of them have zero coefficients.

where $X = [x_1, \ldots, x_n]'$ and eg_i are iid $N(0,1)$, generated independently from X; this quantity is approximated by simulation. The X-independent penalty is more conservative, while the X-dependent penalty level is least conservative. 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 given by $\hat{\psi}_j = \sqrt{\mathbb{E}_n[x_{ij}^2 \hat{\varepsilon}_i^2]}$, where $\hat{\varepsilon}_i$ are preliminary estimates of the errors. The penalty level can be X-independent:

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

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

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

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

The estimation proceeds by iteration. The estimates of residuals $\hat{\varepsilon}_i$ are initialized by running least squares of y_i on five most 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.

R implementation. [TO BE UPDATED]. 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 option `penalty` of the function `rlasso` allows different choices for the penalization parameter and loadings. It allows for homoscedastic or heterosceastic 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.design = "dependent"`. The default is `"dependent"`. With `lambda.start` initial values for the algorithm can be set to get started. In some cases the user might want to use a predefined, fixed penalization parameter. This can be done by a special option in the following way: set homoscedastic to "none" and supply a values `lambda.start`. Then this value is used as penalty parameter with independent design and heteroscedastic errors to weight the regressors.

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`.

Example. Consider generated data from a sparse linear model:

```
set.seed(1)
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 it, show the results and make predictions for the old and for new X-variables

```
# use Lasso for fitting and prediction
lasso.reg = rlasso(Y~X,post=FALSE)  # use Lasso, not-Post-Lasso
## c in penalty not provided.  Set to default c=0.5 for Lasso
print(lasso.reg, all=FALSE)
##
## Call:
## rlasso(formula = Y ~ X, post = FALSE)
##
##       X1        X2        X3       X83       X85
##  4.69459   4.83269   4.79304  -0.04094   0.01293
# summary(lasso.reg, all=FALSE) # use this option to summarize results
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

# now use Post-Lasso for fitting and prediction
post.lasso.reg = rlasso(Y~X,post=TRUE)
print(post.lasso.reg, all=FALSE)    #use this option to print results
##
## Call:
## rlasso(formula = Y ~ X, post = TRUE)
##
##    X1     X2     X3
## 4.945  5.049  4.984
#summary(post.lasso.reg, all=FALSE)  #use this option to summarize results
yhat.postlasso = predict(post.lasso.reg)  #in-sample prediction
yhat.postlasso.new = predict(post.lasso.reg, newdata=Xnew)  #out-of-sample prediction
# compute Mean Absolute Error for Lasso and Post-Lasso predictions:
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)
##      Lasso MAE Post-Lasso MAE
```

```
##           0.84           0.77
```

The methods `print` and `summarize` have the option `all`, which by setting to `FALSE` shows only the non-zero coefficients.

5. Inference on Target Regression Coefficients in Regression, Using Approximate Sparsity

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

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

We assume approximate sparsity for the part of the regression function $x_i'\beta$, with the sufficient speed of decay of the sorted components of β_0, namely $a_\beta > 1$. This translated into the effective sparsity index $s = o(n)$. In general d_i is correlated to x_i, so α_0 can not be consistently estimated by the regression of y_i on d_i. Write

$$d_i = x_i'\gamma_0 + v_i, \quad \mathbb{E}v_i x_i = 0.$$

To estimate α_0, we also impose approximate sparsity on the regression function $x_i'\gamma_0$, with the sufficent speed of decay of sorted coefficients of γ, namely $a_\gamma > 1$.

Note that we can not use naive estimates of α based simply on applying Lasso and Post-Lasso to the first equations. Such strategy in general does not produce root-n consistent and asymptotically normal estimators of α, because there is too much omitted variable bias resulting from estimating the nuisance function $x_i'\beta_0$ in high-dimensional setting. In order to overcome the ommitted variable bias we need to use orthogonalized estimating equations for α_0. These equations in turn rely on the classical ideas of partialling out.

5.1. Intuition to Partialling Out.

One way to think about estimation of α is to think of the equation

$$u_i = \alpha v_i + \epsilon_i,$$

where u_i is the residual that is left after partialling out the linear effect of x_i from y_i and v_i 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 $\mathbb{E}u_i x_i = 0$, i.e. $u_i = y_i - x_i'\delta_0$ where $x_i'\delta_0$ is the linar projection of y_i on x_i. After partialling out, α is the regression coefficient in the bivariate regression of u_i on v_i. This is the Frisch-Waugh-Lovell theorem.

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)
```

One way is to conduct a full OLS fit and then report the parameter of interest

```
# 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
```

An alternative is first to partial out the influence of the x-variables and then do a regression of the corresponding 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 they are asymptotically equivalent due to asymptotically negligible effect from estimating projection parameters on the first-order asymptotics of the partial least squares estimator α.

We can also try Lasso and Post-Lasso for partialling out. Let's see how this strategy would work in our low-dimensional example:

```
# 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 those given above. In fact they are asymptotically equivalent to those reported above in the low-dimensional settings.

In low-dimensional settings this is entirely expected, as any sensible way of approximately partialling out will work to produce first-order equivalent estimators of α_0. In the high-dimensional settings, we can no longer rely on the least-squares and have to rely on lasso and post-lasso for estimating the projection parameters. This results in estimators of α_0 that are root-n consistent and asymptotically normal, in fact achieving the semi-parametric efficiency bounds.

Let us note that the above strategy, based on partialling out via lasso or post-lasso is implemented in the package by the function `rlassoEffect`.

```
Eff= rlassoEffect(X[,-1],y,X[,1], method="partialling out")
cbind(Eff$coef, Eff$se)
##                       [,1]       [,2]
## (Intercept) -5.007175e-17 0.01368677
## dr           9.727387e-01 0.01368677
```

Another first-order equivalent strategy implemented relies on the double-selection method, which uses the union of $x_{i,j}$'s selected in two projection equations for partialling out. This is equivalent to doing OLS of y_i on d_i and the union of controls selected in the two projection equations.

```
Eff= rlassoEffect(X[,-1],y,X[,1], method="double selection")
cbind(Eff$coef, Eff$se)
##            [,1]
## [1,] 0.01415624
```

5.2. **Inference.** Besides estimation inference is highly relevant for empirical applications. Using LASSO as a method for penalized estimation of the coefficients of a sparse linear model is useful for obtaining forecasting rules and for estimating which variables have a strong association to an outcome in a sparse framework. However, naively using the results obtained from such a procedure to draw inferences about model parameters may be invalid. The reason is that only under very strong conditions LASSO is consistent concerning model selection which hardly apply in empirical applications and omission of relevant variables cannot be excluded in general distorting the validity of traditional approaches. This caveat applies to both low- **and** high-dimensional settings when model selection is considered. Moreover, even in the case of consistent model selection non-uniformity of the results might lead to poor finite sample approximations. Pötscher and Leeb have analyzed these issues in a series of papers.

Valid inference on low-dimensional parameters in a high-dimensional setting as described in the previous section is possible by the so-called double selection method.

We are interested in doing inference on a low dimensional parameter α. For ease of exposition, we consider the case that the parameter of interest is a scalar. This gives us the following model

$$y_i = \alpha_0 d_i + x_i' \beta_0 + \xi_i, i = 1, \ldots, n,$$

with x_i p-dimensional vector of controls, d_i the low-dimensional object we are concerned about, and ξ_i iid errors with $\mathbb{E}[\xi_i|d_i, x_i] = 0$. To perform valid inference, we introduce an auxiliary regression:

$$d_i = x_i'\theta + v_i,$$

where $\mathbb{E}[v_i|x_i] = 0$ and captures the relation between variable of interest and the control variables. The double selection method is designed to guard against moderately sized model selection mistakes and consists of the following steps:

(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.

The additional selection step controls the omitted variable bias. In essence the procedure is a selection version of Frisch-Waugh procedure for estimating linear regression.

5.3. **Functions for Lasso and Inference.** The package contains several functions for estimation and inference by LASSO. In this section we introduce the core functions which are then illustrated in an example.

The core functions for estimation are `LassoShooting.fit` and `rlasso`. `LassoShooting.fit` is the working horse for the estimation and implements a version of the Shooting Lasso Algorithm. It allows variable dependent penalties or weights. The function `rlasso` implements LASSO and Post-LASSO estimation with non-Gaussian errors and under heteroskedasticity. The default option is `post=TRUE`. The user can also decide if an unpenalized `intercept` should be included (`TRUE` by default) and if the regressors should be `normalized` (`TRUE` by default). The choice of `penalty` is passed by a list. It can be chosen between homoscedastic errors (`FALSE` by default) and if the penalty should take into consideration the design of the `X` matrix which gives tighter bounds on the penalization parameter. The two options for `X.design` are `dependent` and the default `independent`. This function returns an object of S3 class `rlasso` for which methods like `predict`, `print`, `summary` are provided. The function `rlassoEffect` does inference for selected variables. Those can be specified either by the variable names, an integer valued vector giving their position in `x` or by logical indicating the variables for which inference should be conducted. It returns an object of S3 class `rlassoEffect` for which the methods `summary`, `print`, `confint`, and `plot` are provided. `rlassoEffect` is a wrap function for `rlassoEffectone` which does inference for a single, specified variable.

5.4. **Example.** In this section we demonstrate the functionality for LASSO estimation and inference at a simulated data set. In the next sections the methods will be illustrated at hand of empirical applications using the data sets introduced earlier.

First we generate data in a sparse linear model:

```
set.seed(1234)
n = 250 #sample size
p = 100 # number of variables
```

```
s = 10 # nubmer of non-zero variables
X = matrix(rnorm(n*p), ncol=p)
beta = c(rep(2,s), rep(0,p-s))
y = 1 + X%*%beta + rnorm(n)
```

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

```
lasso.effect = rlassoEffect(x=X, y=y, index=c(1,2,50))
## Error in as.matrix(d, ncol = 1):  argument "d" is missing, with no default
print(lasso.effect)
## Error in print(lasso.effect):  object 'lasso.effect' not found
summary(lasso.effect)
## Error in summary(lasso.effect):  object 'lasso.effect' not found
confint(lasso.effect)
## Error in confint(lasso.effect):  object 'lasso.effect' not found
```

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

```
plot(lasso.effect, main="Confidence Interval of Selected Variables")
## Error in plot(lasso.effect, main = "Confidence Interval of Selected Variables"):  object
'lasso.effect' not found
```

5.5. **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. Such a hypothesis implies that the effect of the initial level of GDP on the growth rate should be negative. As pointed out in Barro and Sala-i-Martin, this hypothesis is rejected using a simple bivariate regression of growth rates on the initial level of GDP. (In this data set, linear regression yields an insignificant positive coefficient of 0.0013.) In order to reconcile the data and the theory, the literature has focused on estimating the effect conditional on the pertinent characteristics of countries. Covariates that describe such characteristics can include variables measuring education and science policies, strength of market institutions, trade openness, savings rates and others. The theory then predicts that for countries with similar other characteristics the effect of the initial level of GDP on the growth rate should be negative. Thus, we are interested in a specification of the form:

$$y_i = \alpha_0 + \alpha_1 \log G_i + \sum_{j=1}^{p} \beta_j X_{ij} + \varepsilon_i,$$

where y_i is the growth rate of GDP over a specified decade in country i, G_i is 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 standard data-sets, such as Barro and Lee data, the number of covariates p we can condition on is large, at least relative to the sample size n, covariate selection becomes a crucial issue in this analysis. In particular, previous findings came under severe criticism for relying on ad hoc procedures for covariate selection. In fact, in some cases, all of the previous findings have been questioned. Since the number of covariates is high, there is no simple way to resolve the model selection problem using only classical tools. Indeed the number of possible lower-dimensional models is very large, although [16] and [22] attempt to search over several millions of these models. We suggest ℓ_1-penalization and post- ℓ_1-penalization methods to address this important issue. In Section 8, using these methods we estimate the growth model (5.5) and indeed find rather strong support for the hypothesis.

First, we load and prepare the data

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

Now we can estimate the influence of initial GDP level to test the Economic hypothesis

```
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= "+")))
full.fit= lm(fmla, data=GrowthData)
fmla.y= as.formula(paste("Outcome ~ ", paste(xnames, collapse= "+")))
fmla.d= as.formula(paste("gdpsh465~ ", paste(xnames, collapse= "+")))
# partial fit via ols
rY= lm(fmla.y, data =GrowthData)$res
rD= lm(fmla.d, data =GrowthData)$res
partial.fit.ls= lm(rY~rD)
# partial ls via lasso
rY= rlasso(fmla.y, data =GrowthData)$res
rD= rlasso(fmla.d, data =GrowthData)$res
partial.fit.lasso= lm(rY~rD)
```

```
dX = as.matrix(cbind(d,X))
partial.fit.lasso2 = rlassoEffect(x=dX, y=y, index=1, intercept=TRUE, normalize=TRUE, post=TRUE)
## Error in as.matrix(d, ncol = 1):  argument "d" is missing, with no default
#partial.fit.lasso2 = rlassoEffectone(x=X,y=y,d=d)
summary(partial.fit.lasso2)
## Error in summary(partial.fit.lasso2):  object 'partial.fit.lasso2' not found
```

```
library(xtable)
table= matrix(0, 4, 2)
table[1,]= summary(full.fit)$coef["gdpsh465",1:2]
table[2,]= summary(partial.fit.ls)$coef["rD",1:2]
table[3,]= summary(partial.fit.lasso)$coef["rD",1:2]
table[4,]= summary(partial.fit.lasso2)["d",1:2]
## Error in summary(partial.fit.lasso2):  object 'partial.fit.lasso2' not found
colnames(table)= names(summary(full.fit)$coef["gdpsh465",])[1:2]
rownames(table)= c("full reg", "partial reg", "partial reg via lasso 1", "partial reg
via lasso 2")
# adjust partial ls standard error by sqrt{n/(n-p)}
n= dim(GrowthData)[1]; p= dim(GrowthData)[2]
table[2,2]= table[2,2]*sqrt(n/(n-p))
tab= xtable(table, digits=c(2, 2,7))
```

```
tab
```

	Estimate	Std. Error
full reg	-0.01	0.0298877
partial reg	-0.01	0.0307801
partial reg via lasso 1	-0.04	0.0153192
partial reg via lasso 2	0.00	0.0000000