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 semi-parametrically efficient estimators, confidence intervals, and significance testing for low-dimensional subcomponents of the high-dimensional parameter vector. 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

Version: January 27, 2016.

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, 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 coverted 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 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 method. Because of this we call the resulting method the "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 for the choice of λ are often not feasible.

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

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. 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 class-room presentation.

3.1. **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 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 macroeconomics. A well-known data set with information about GDP growth for many countries over a long period was collected by Barro and Lee. 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.

3.3. **Institutions and Economic Development – Data on settler mortality.** This data set was collected by Acemoglu, Johnson, and Robinson (2001) 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.

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 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), the Case-Shiller home price index (CS), and state-level GDP from the Bureau of Economic Analysis.

4. Prediction using Approximate Sparsity

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

The model reads:

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

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 are expressly allowed to 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

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

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 $||\beta||_1 = \sum_{j=1}^p |\beta_j|$ and $\hat{\Psi} = \text{diag}(\hat{\psi}_1, \ldots, \hat{\psi}_p)$ 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 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, 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 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.

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 really lacks theoretical justification and so does not provide such 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

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

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, 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 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

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

where e_i are iid standard normal variables distributed independently 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. The resulting iterative procedure is theoretically fully justified in the theoretical literature.

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 algoritm 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, they 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 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.dependent.lambda = "TRUE"`. The user can provide her own initial value for the penalty level by setting `homoscedastic = "none"` and supplying a starting 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`.

Example. (Prediction Using Lasso and Post-Lasso) 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 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
print(lasso.reg, all=FALSE) # can also do summary(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
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 = Y ~ X, post = TRUE)
##
##    X1     X2     X3
## 4.945  5.049  4.984
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
##           0.84           0.77
```

5. Inference on Target Regression Coefficients in Regression

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 treat each variable at a time – by simply writing each time the model in the above form and then applying estimation and inference procedures described below.

We assume approximate sparsity for the part of the regression function $x_i'\beta_0$, with the sufficient speed of decay of the sorted components of β_0, namely $\mathsf{a}(\beta_0) > 1$. This translates into the sparsity index $s \ll \sqrt{n}$. 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. 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 $x_i'\gamma_0$, with the 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 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 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 $w_i = (y_i, x_i')'$, η is the nuisance parameter, such that

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

where the second equation is the orthgonality 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, regularizaton has too much effect on the estimator of α_0.

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. Thanks due to the orthogonality property the etimator is first-order equivalent to the infeasible estimator $\tilde{\alpha}$ solving

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

where we've plugged in the true value of the nuisance parameter. The equivalence holds in a variety of models under plausible conditions.

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

5.1. **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 associtated 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 the high-dimensional settings, we can no longer rely on the full least-squares and have to rely on Lasso and Post-Lasso for partialling out. This amounts to using orthogonal estimating equations, where we estimate the nuisance parameters by Lasoo 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
```

5.2. **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 `rlassoEffect` 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.

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)
beta = c(rep(3,s), rep(0,p-s))
y = 1 + X%*%beta + rnorm(n)
```

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

```
lasso.effect = rlassoEffects(x=X, y=y, index=c(1,2,3,50))
print(lasso.effect)
```

```
##
## Call:
## rlassoEffects(x = X, y = y, index = c(1, 2, 3, 50))
##
## Coefficients:
##      V1       V2       V3      V50
## 2.94547  3.04876  2.98373  0.07519
summary(lasso.effect)
## [1] "Estimation of the effect of selected variables in a high-dimensional regression"
##     Estimate. Std. Error t value Pr(>|t|)
## V1    2.94547    0.08985  32.784   <2e-16 ***
## V2    3.04876    0.08202  37.169   <2e-16 ***
## V3    2.98373    0.07767  38.416   <2e-16 ***
## V50   0.07519    0.07845   0.958    0.338
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
confint(lasso.effect)
##          2.5 %    97.5 %
## V1   2.7693798 3.1215660
## V2   2.8879933 3.2095182
## V3   2.8315002 3.1359565
## V50 -0.0785745 0.2289456
```

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.

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

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

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

where y_i is the growth rate of GDP over a specified decade in country i, d_i is the log of the initial level of GDP at the beginning of the specified period, and the x_{ij}'s form a long list of country i's characteristics at the beginning of the specified period. We are interested in testing the hypothesis of convergence, namely that $\alpha_1 < 0$. Given that in the Barro and Lee 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. We can also employ the partialling out approach.

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] "Estimation of the effect of selected variables in a high-dimensional regression"
##      Estimate. Std. Error t value Pr(>|t|)
## [1,]  -0.04432    0.01532  -2.893  0.00381 **
## ---
## 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] "Estimation of the effect of selected variables in a high-dimensional regression"
```

```
##           Estimate. Std. Error t value Pr(>|t|)
## xgdpsh465  -0.04432    0.01685  -2.631  0.00851 **
## ---
## 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	-0.04	0.01532
partial reg via double selection	-0.04	0.01685

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.

6. 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 which might occur because of several reasons. In this case the instrumental variables (IV) methods are used for identification of the causal parameters.

We consider the linear instrumental variables model:

$$y_i = \alpha_0 d_i + \beta_0 x_i' + \varepsilon_i, \tag{1}$$

$$d_i = z_i'\Pi + \gamma_0 x_i' + v_i, \tag{2}$$

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.

6.1. **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, x_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, we (1) do Lasso or Post-Lasso regression of d_i on x_i, z_i to obtain $\hat{\gamma}$ and $\hat{\delta}$; (2) do Lasso or Post-Lasso regression of y_i on x_i to get $\hat{\theta}$; (3) do Lasso or Post-Lasso regression of $\hat{d}_i = x_i'\hat{\gamma} + z_i'\hat{\delta}$ on x_i to get $\hat{\vartheta}$; (4) let $\hat{\rho}_i^y := y_i - x_i'\hat{\theta}$, $\hat{\rho}_i^d := d_i - x_i'\hat{\vartheta}$, and $\hat{v}_i := x_i'\hat{\gamma} + z_i'\hat{\delta} - x_i'\hat{\vartheta}$. We then get estimator $\hat{\alpha}$ by using standard IV regression of $\hat{\rho}_i^y$ on $\hat{\rho}_i^d$ with $\hat{v}_i$ as the instrument. Perform inference on α using $\hat{\alpha}$ and conventional heteroscedasticity robust standard errors.

R Implementation. The wrap function `rlassoIV` handels all of the previos cases. It has the options `select.X` and `select.Z` which implement selection of either covarites or intsruments, 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 fuctions `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 used 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.

6.2. **Application: Economic Development and Institutions.** Estimating the causal effect of institutions on output is complicated by the simultaneity between institutions and output: specifically, better institutions may lead to higher incomes, but higher incomes may also lead to the development of better institutions. To help overcome this simultaneity, Acemoglu, Johnson, and Robinson (2001) use mortality rates for early European settlers as an instrument for institution quality. The validity of this instrument hinges on the argument that settlers set up better institutions in places where they are more

likely to establish long-term settlements; that where they are likely to settle for the long term is related to settler mortality at the time of initial colonization; and that institutions are highly persistent. The exclusion restriction for the instrumental variable is then motivated by the argument that GDP, while persistent, is unlikely to be strongly influenced by mortality in the previous century, or earlier, except through institutions.

In this application, we consider the problem of selecting controls. The input raw controls are the Latitude and the continental dummies. The technical ctonrols can include various polynomial transformations of the Latitude, possibly interacted with the continental dummies. Such flexible specificaitons 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)
summary(AJR.Xselect)
## [1] "Estimation of the effect of selected variables in a high-dimensional regression"
##    coeff.    se. t-value  p-value
## d1 0.6358 0.1888   3.367 0.000761 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
confint(AJR.Xselect)
##        2.5 %   97.5 %
## d1 0.2656562 1.005923
```

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 residualized quantities.

Let us investigate partilling 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)$res
ivfit.lm = tsls(y=rY,d=rD, x=NULL, z=rZ, intercept=FALSE)
print(cbind(ivfit.lm$coef, ivfit.lm$se),digits=3)
##   [,1] [,2]
## d 1.27 1.73
```

We see that the estimates are very noisy, as the standard errors are very large. This is expected in some ways 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)$res
ivfit.lasso = tsls(y=rY,d=rD, x=NULL, z=rZ, intercept=FALSE)
print(cbind(ivfit.lasso$coef, ivfit.lasso$se), digits=3)
##       [,1]  [,2]
## [1,] 0.636 0.189
```

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

6.3. **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 the economic outcomes. The analysis of the effects of such decisions is complicated by the possible endogeneity between the decisions and potential economic outcomes. To address the potential endogeneity, we employ an instrumental variables strategy based on on the random assignment of judges to 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.

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

```
data(EminentDomain)
z <- EminentDomain$logGDP$z
x <- EminentDomain$logGDP$x
y <- EminentDomain$logGDP$y
```

```
d <- EminentDomain$logGDP$d
```

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 socio-economic and demographic characteristics of the judges.

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 = rlassoIVselectZ(x=X, d=d, y=y, z=Z)
summary(lasso.IV.Z)
## [1] "Estimation of the effect of selected variables in a high-dimensional regression"
##     coeff.     se. t-value p-value
## d1 0.01659 0.01775   0.934    0.35
confint(lasso.IV.Z)
##          2.5 %     97.5 %
## d1 -0.01821032 0.05138162
```

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)
## Estimation of the effect of selected variables in a high-dimensional IV regression
##       coeff.       se. t-value p-value
## d1 -0.005441  0.188125  -0.029   0.977
confint(lasso.IV.XZ)
##         2.5 %    97.5 %
## d1 -0.3741584 0.3632765
```

Comparing the results we see, that the OLS estimates indicate that the influence of pro plaintiff appellate takings decisions in federal circuit court is siginifcantly positive. But the 2SLS estimates which account for the potential endogeneity render the results insignificant. Employing selection on the instruments confirms this result. Selection on both the x- and z-variables drives the estimate below zero, but yielding similar economic conclusions.

Finally, we compare all results

```
tab
```

	Estimate	Std. Error
ols regression	0.01	0.0052848
IV estimation	0.02	0.0183376
selection on Z	0.02	0.0177534
selection on X and Z	-0.01	0.1881246

7. Inference on Treatment Effects in a High-Dimensional Setting

In this section we consider estimation and inference on treatment effects when 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.

7.1. **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 programme 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 ortoghonal estimating equations for these parameters.[1]

In observational studies the potential outcomes are endgenous, i.e. related to the treatment variable D. In such cases, researchers use a binary instrument Z that affects the treatment but does not affect

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

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 – that are the 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 regression or propensity score weighting methods. Our identification and estimation method rely on the combination of two methods to create ortoghonal estimating equations for these parameters.

7.2. **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 vontrol 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 neccessary to absorb residual variation, but overfitting should be avoided. For uniformly valid post-selection inference, "orhtogonal "estimating equations are they key to the efficient estimation and valid inference.

R Impementation. 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 repitions can be specified with the argument `nRep` and the default is set to 500. By default no booststrap standared 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.

7.3. **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 con- ventional 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 = model.matrix(~(age + inc + fsize + educ + marr + twoearn + db +pira)^2, data=pension)[,-1]
```

Now we can compute the estimates of the target treatment effect parameters:

```
pension.late = rlassoLATE(X,d,y,z)
summary(pension.late)
## Estimation of the treatment effect in a high-dimensional setting
## Type: LATE
## Bootstrap: not applicable
##    coeff.  se. t-value p-value
## TE   5384 5024   1.072   0.284
pension.latet = rlassoLATET(X,d,y,z)
summary(pension.latet)
## Estimation of the treatment effect in a high-dimensional setting
## Type: LATET
## Bootstrap: not applicable
##    coeff.   se. t-value p-value
## TE   2112 12207   0.173   0.863
pension.ate =  rlassoATE(X,d,y)
summary(pension.ate)
## Estimation of the treatment effect in a high-dimensional setting
## Type: ATE
## Bootstrap: not applicable
##    coeff.  se. t-value p-value
## TE   6787 2956   2.296  0.0217 *
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
pension.atet =  rlassoATET(X,d,y)
summary(pension.atet)
## Estimation of the treatment effect in a high-dimensional setting
## Type: ATET
## Bootstrap: not applicable
##    coeff.  se. t-value p-value
## TE   2675 8187   0.327   0.744
```

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

	Estimate	Std. Error
ATE	6786.65	2956.48
ATET	2675.24	8187.06
LATE	5383.90	5023.50
LATET	2112.31	12206.90

8. Conclusion

We have proived 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 .