Journal of Statistical Software

April 2017, Volume 77, Issue 10. *doi: 10.18637/jss.v077.i10*

NonpModelCheck: An R Package for Nonparametric Lack-of-Fit Testing and Variable Selection

Adriano Zanin Zambom
Loyola University Chicago

Michael G. Akritas
Penn State University

Abstract

We describe the R package **NonpModelCheck** for hypothesis testing and variable selection in nonparametric regression. This package implements functions to perform hypothesis testing for the significance of a predictor or a group of predictors in a fully nonparametric heteroscedastic regression model using high-dimensional one-way ANOVA. Based on the p values from the test of each covariate, three different algorithms allow the user to perform variable selection using false discovery rate corrections. A function for classical local polynomial regression is implemented for the multivariate context, where the degree of the polynomial can be as large as needed and bandwidth selection strategies are built in.

Keywords: high dimensional one-way ANOVA, local polynomial regression, false discovery rate.

1. Introduction

In modern science, with the advances of computing and data storage, new challenges are often found when building models that can predict the behavior of a response variable according to massive sets of covariates and large number of data points. To address these issues, one fundamental procedure that can be used is that of model checking which, besides testing parameters and assumptions, can also be viewed as testing hypotheses about the significance of a covariate or a group of covariates in the model. This is of great importance for model building due to the bias or lack of power that misspecification of covariates can cause. There are basically two main principles for model building: parsimony and sparseness. Parsimony is the idea of using few parameters and sparseness is the concept of prediction with only a few variables. Connecting these two principles, variable selection methods have been proposed, with the objective of finding a model that contains only significant covariates. Hence, model checking and variable selection have become extremely important tools for building predictive models, being implemented by several computer packages and software.

Recent research on variable selection and predictive models have significantly contributed to a wide variety of applications. However, most attention has been given to linear models, i.e., when the expected response at $\mathbf{x}$, the observed vector of covariates, is assumed to be $m(\mathbf{x}) = \mathbf{x}^\top \boldsymbol{\beta}$, where $\boldsymbol{\beta}$ is the parameter vector to be estimated. One of the reasons why such an analysis is preferred is because of the readily available statistical software. With the linear model, a new class of methodologies has recently been developed, which uses penalty functions to perform variable selection through the choice of a shrinking parameter, such as LASSO (Tibshirani 1996), SCAD (Fan and Li 2001), least angle regression (Efron, Hastie, Johnstone, and Tibshirani 2004), adaptive LASSO (Zou 2006), and Dantzig selector (Candes and Tao 2005). Some of these methods are already available in R (R Core Team 2017), such as packages **ncvreg** for SCAD (Breheny and Huang 2011; Breheny 2017), **glmnet** for LASSO and elastic net (Friedman, Hastie, and Tibshirani 2010), **lars** for least angle regression (Hastie and Efron 2013).

Another approach explores the connection of model checking and variable selection. Let y_i denote the ith response observation, x_{ij} the ith observation of the jth predictor. Abramovich, Benjamini, Donoho, and Johnstone (2006) showed that the application of the false discovery rate (FDR) procedure of Benjamini and Hochberg (1995) on p values resulting from testing $H_0^j : \beta_j = 0$ in the aforementioned linear model, can be translated into minimizing a model selection criterion of the form

$$\sum_{i=1}^{n} \left(y_i - \sum_{j \in S} \hat{\beta}_j^S x_{ij} \right)^2 + \sigma^2 |S| \lambda,$$

where S is a subset of $\{1, 2, \ldots, d\}$ specifying the model, $\hat{\beta}_j^S$ denotes the least squares estimator from fitting model S, $|S|$ is the cardinality of the subset S, and the penalty parameter λ depends both on d and $|S|$.

The assumption of linearity of the expected response function may not always be correct. A model checking or variable selection procedure with such an assumption may not detect significant covariates that have nonlinear effects. Because of this, procedures for both model checking and variable selection have been developed under more general/flexible models; see, for example, Claeskens (2004), Li and Liang (2008), Wang and Xia (2009), Meinshausen, Meier, and Bühlmann (2009), Huang, Horowitz, and Wei (2010), Storlie, Bondell, Reich, and Zhang (2011), Antoniadis, Gijbels, and Lambert-Lacroix (2014), Gijbels, Verhasselt, and Vrinssen (2015) and references therein. In a completely nonparametric regression, Zambom and Akritas (2014) introduced a variable selection procedure using its conceptual connection with model checking. The idea is to perform backward elimination considering the Benjamini and Yekutieli (2001) method applied to the p values resulting from testing the significance of each covariate. The extension to testing a group of covariates and running group selection was explored in Zambom and Akritas (2015). This new methodology inspired the development of package **NonpModelCheck** (Zambom 2015), which provides functions for testing, in a nonparametric regression context, the significance of a covariate or a group of covariates, and functions to perform variable selection using the results of these tests.

The remainder of this paper is organized as follows. In Section 2 we summarize the methodology of hypothesis testing in nonparametric regression introduced by Zambom and Akritas (2014) and Zambom and Akritas (2015). The function for model checking in package **NonpModelCheck** is presented in Sections 2.1 and 2.2, along with some examples. Section 3 introduces the modified variable selection algorithm, based on backward elimination or forward selection and FDR corrections, followed by the description of the function `npvarselect()` and performance comparisons with methods in the literature. The package is available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=NonpModelCheck`.

2. Nonparametric model checking

The effects of model misspecification in linear regression have been explored in several papers, such as Larson and Bancroft (1963), Ramsey (1969), Deegan Jr. (1976), among others. Also, in the nonlinear context, it has been seen that inappropriate assumptions or misspecified models can cause serious consequences, see for example White (1981), and Hartford and Davidian (2000).

One important question in checking appropriateness of a model, is to check if the covariates under consideration are significantly related to the response variable. An example of such a situation is when a researcher is doubtful of a few of the measured predictors, and wants to decide if they are necessary to accurately predict the response. Hypothesis testing can be performed on these problems in order to test a specific covariate or a group of covariates at once. Moreover, since recent experiments are based on a rather large number of predictors, often researchers do not know in advance which of those are significant to the model. Hence, the aforementioned connection between hypothesis testing and variable selection is an important tool for model building, and will be explored in the next section. Next, we will formalize the hypothesis testing for the significance of one or several covariates in the mean regression function in nonparametric regression. Denote the response variable by Y and the set of d available covariates by $\mathbf{U}$. The nonparametric regression model, allowing for heterocedasticity, can be written as

$$Y = m(\mathbf{U}) + \sigma(\mathbf{U})\epsilon, \tag{1}$$

where $\sigma(\mathbf{U})$ is the variance function and ϵ is the random error with zero mean and constant variance independent of $\mathbf{U}$. Writing the set of covariates as $\mathbf{U} = (\mathbf{X}, \mathbf{Z})$, where $\mathbf{X}$ has dimension r and $\mathbf{Z}$ has dimension s $(r+s=d)$, the concept of model checking as a hypothesis test for covariates can be written in the following way

$$H_0 : m(\mathbf{x}, \mathbf{z}) = m_1(\mathbf{x}). \tag{2}$$

This null hypothesis is basically stating that the mean regression function depends only on some of the observed covariates, specifically, only on $\mathbf{x}$. Note that if $s = 1$, we are only testing if a single covariate should be dropped from the model, but for $s > 1$ this is a multivariate test.

One class of procedures is based on the idea that the null hypothesis residuals, $\xi = Y - m_1(\mathbf{X}_1)$, satisfy $\mathsf{E}(\xi|\mathbf{X}) = 0$ under H_0 but under the alternative are equal to $\mathsf{E}(\xi|\mathbf{X}) = m(\mathbf{X}) - m_1(\mathbf{X}_1)$. Hence, it is straightforward to see that $\mathsf{E}(\xi\mathsf{E}(\xi|\mathbf{X})|\mathbf{X}) = (m(\mathbf{X}) - m_1(\mathbf{X}_1))^2$ under the alternative and zero under the null. Using this idea, Fan and Li (1996) proposed a test statistic based on $\mathsf{E}[\xi f_1(\mathbf{X}_1)\mathsf{E}(\xi f_1(\mathbf{X}_1)|\mathbf{X})f(\mathbf{X})]$, a weighted version of $\mathsf{E}(\xi\mathsf{E}(\xi|\mathbf{X})|\mathbf{X})$, avoiding the randomness in the denominator (when estimating $m(\mathbf{X}) = \int yf(x,y)dy/f_X(x)$). Note that $\mathsf{E}[\xi f_1(\mathbf{X}_1)\mathsf{E}(\xi f_1(\mathbf{X}_1)|\mathbf{X})f(\mathbf{X})] = \mathsf{E}[(m(\mathbf{X}) - m_1(\mathbf{X}_1))^2 f_1(\mathbf{X})^2 f(\mathbf{X})]$ under the alternative and zero under the null. They show that their test statistic is asymptotically normal under H_0. Lavergne and Vuong (2000) propose a test statistic based on a different estimator of the same quantity as Fan and Li (1996), which is also proven to be asymptotically normal under H_0. A related class of procedures is based on the direct estimation of $\mathsf{E}[(m(\mathbf{X}) - m_1(\mathbf{X}_1))^2 W(\mathbf{X})]$, for some weight function W; see, for example, Ait-Sahalia, Bickel, and Stoker (2001). The use of such test statistics is complicated by the need to correct for their bias. Other tests are also available, such as the bootstrap-based procedure of Delgado and Manteiga (2001) (which is computationally intensive) and the generalized likelihood ratio test of Fan, Zhang, and Zhang (2001) and Fan and Jiang (2005).

The R package **NonpModelCheck** provides a function for hypothesis testing of covariate(s) using the test statistic proposed by Zambom and Akritas (2014), which, in simulation, outperformed the procedures described above. The key idea of the test statistic proposed by Zambom and Akritas (2014) is to construct a high dimensional one-way ANOVA, with factor levels being the values of the covariates $\mathbf{Z}_i$, $i = 1, \ldots, n$, and $\hat{\xi}_i = Y_i - \hat{m}_1(\mathbf{x}_i)$ being the observations from factor level $\mathbf{Z}_i$. With this setup, testing for no factor effects is an intuitive test for H_0. In order to estimate the residuals $\hat{\xi}_i$, one needs to estimate $m_1(\mathbf{x})$, which is done by local polynomial regression for its smoothness and asymptotic properties.

Due to the fact that, in high dimensional ANOVA, theoretical assumptions require more than one observation per cell, factor levels here are augmented by including residuals from nearby covariate values. To be specific, consider the simple situation when $s = 1$, that is, when Z is univariate. In this case, each cell is augmented by including additional $(p-1)$ $\hat{\xi}_\ell$'s which correspond to the $(p-1)/2$ Z_ℓ values that are nearest to Z_i on either side. More precisely, the $(\hat{\xi}_i, Z_i)$, $i = 1, \ldots, n$, are considered to be arranged so that $Z_{i_1} < Z_{i_2}$ whenever $i_1 < i_2$, and for each Z_i, $(p-1)/2 < i \leq n - (p-1)/2$, define the nearest neighbor window W_i as

$$W_i = \left\{ j : |\hat{F}_Z(Z_j) - \hat{F}_Z(Z_i)| \leq \frac{p-1}{2n} \right\}, \tag{3}$$

where $\hat{F}_Z$ is the empirical distribution function of Z. Hence, the augmented cell corresponding to Z_i is defined by the values of $\hat{\xi}$ with indices in W_i. Hence, the vector of $(n-p+1)p$ constructed "observations" in the augmented one-way ANOVA design is

$$\hat{\boldsymbol{\xi}} = (\hat{\xi}_j, j \in W_{(p-1)/2+1}, \ldots, \hat{\xi}_j, j \in W_{n-(p-1)/2})^\top. \tag{4}$$

Let MST $=$ MST$(\hat{\boldsymbol{\xi}})$ and MSE $=$ MSE$(\hat{\boldsymbol{\xi}})$ denote the balanced one-way ANOVA mean squares due to treatment and error, respectively, computed on the data $\hat{\boldsymbol{\xi}}$. The test statistic is

$$n^{1/2}(\text{MST} - \text{MSE}), \tag{5}$$

which has an asymptotic normal distribution with mean 0 and variance $\frac{2p(2p-1)}{3(p-1)}\tau^2$, where $\tau = \int \left[\int \sigma^2(\mathbf{x}, z) f_{\mathbf{X}|Z=z}(\mathbf{x}) d\mathbf{x} \right]^2 f_Z(z) dz$. Zambom and Akritas (2014) suggest the following estimator of τ^2

$$\hat{\tau}^2 = \frac{1}{4(n-3)} \sum_{j=2}^{n-2} (\hat{\xi}_j - \hat{\xi}_{j-1})^2 (\hat{\xi}_{j+2} - \hat{\xi}_{j+1})^2. \tag{6}$$

For the case when $\mathbf{Z}$ is multidimensional, the task of ordering the factor levels according to $\mathbf{Z}_i$ becomes a challenge. This is overcome by replacing $\mathbf{Z}_i$ by a nonlinear version of Bair, Hastie, Paul, and Tibshirani (2006)'s first supervised principal component (SPC), $P_i^\lambda = \mathbf{Z}_i^\top \mathbf{C}_\lambda$, defined in the following way. Consider the p values p_j, obtained by applying the univariate tests $H_0^j : m(\mathbf{X}, Z_j) = m(\mathbf{X}), j = 1, \ldots, s$. Let $\mathbf{Z}^\lambda$ have the same coordinates of $\mathbf{Z}$ except those whose corresponding p values p_j are smaller than a threshold parameter λ. Then, P_i^λ is defined to be the first principal component of $\mathbf{Z}^\lambda$. Therefore, when $\mathbf{Z}$ has dimension larger than 1, the windows W_i are formed using P_i^λ instead of Z_i, and the computation of the test statistic $n^{1/2}(\text{MST} - \text{MSE})$ and $\hat{\tau}$ also depend on P_i^λ.

It is important to note that, computationally the test is carried out by a one-sided test, comparing

$$\frac{n^{1/2}(\text{MST} - \text{MSE})}{\sqrt{\frac{2p(2p-1)}{3(p-1)}\hat{\tau}^2}}$$

with the standard Normal distribution, since the test statistic tends to get larger as we depart from the null hypothesis.

2.1. Hypothesis testing for a single covariate with NonpModelCheck

In this section we give specific examples on how to use the function `npmodelcheck()` for testing hypotheses as in (2). To use this function, the user needs to provide a few parameters/attributes which can basically be divided into 3 groups: 1. data and index of covariates to be tested, 2. parameters for the local polynomial fit and 3. parameters for dimension reduction. The input data is basically a vector or matrix of observed covariates and a vector of observed responses, and the user is also required to input the index of the covariate(s) to be tested. For the parameters of the local polynomial fit, see Appendix A.

Note that, under the assumption of a sparse model, the estimation of m_1 in a nonparametric context will suffer from the curse of dimensionality. To moderate this issue, there are available, within the function, two techniques to reduce the dimension of $\mathbf{X}$ before applying the local polynomial fit. The first technique is the nonlinear SPC, as described previously, with fixed $\lambda = 0.3$. The second technique available for the user is the sliced inverse regression (SIR) method proposed by Li (1991). In SIR, it is assumed that a small number of linear combinations of the predictors contain all information about Y. More specifically, it is assumed that $y = m(\mathbf{x}^\top\beta_1, \ldots, \mathbf{x}^\top\beta_K, \epsilon)$, where β's are unknown row vectors and ϵ is independent of $\mathbf{x}$. This is an extension of the so called multiple index model. The estimation of these directions are based on the inverse regression of $\mathbf{x}^\top$ on y coordinate-wise. A brief description of this procedure is as follows: Let $\mathbf{x}_i, i = 1, \ldots, n$, be the p-dimensional vector of the observation i; standardize it to obtain $\tilde{\mathbf{x}}_i = \hat{\Sigma}^{-1/2}(\mathbf{x}_i - \bar{\mathbf{x}})$; divide the range of y in H slices and let $\hat{p}_h$ be the proportion of y_i that falls in slice $h \in H$; calculate $\hat{m}_h = \frac{1}{n\hat{p}_h}\sum_{\{y_i \text{ in slice } h\}} \tilde{\mathbf{x}}_i$; perform a weighted principal components analysis by getting the eigenvectors and eigenvalues of $\hat{V} = \sum_{h=1}^{H} \hat{p}_h \hat{m}_h \hat{m}_h^\top$; the result is $\hat{\beta}_k = \hat{\eta}_k \hat{\Sigma}^{-1/2} (k = 1, \ldots, K)$, where $\hat{\eta}_k$ are the K largest eigenvectors of $\hat{V}$. The number of eigenvectors is chosen by a sequential test by testing if the K first eigenvalues of V are significantly different from zero (see Li 1991 for details).

The call of the function, as described in the package manual, is:

```
npmodelcheck(X, Y, ind_test, p = 7, degree.pol = 0,
  kernel.type = "epanech", bandwidth = "CV", gridsize = 30,
  dim.red = c(1, 10))
```

The function `npmodelcheck()` returns an object of the class '`npmodelcheck`', containing a list with 3 items: the bandwidth used for the local polynomial fit, the predicted/smoothed values of the local polynomial fit and the p value of the hypothesis test. When simply called, it will only print the p value of the test, but a summary function is available, so that when the object of class '`npmodelcheck`' is inputted, it will display all items.

To illustrate the function `npmodelcheck()`, we will consider the following situation. Suppose we have 100 observations of a set $\mathbf{U}$ of 12 covariates and a response variable Y. Assume that the true, but unknown regression function is

$$\begin{aligned} m(\mathbf{U}) &= m_1(U_2, U_5) \\ &= \frac{\theta}{4}\left(\sin(2U_2) + 2\exp(-16U_2^2)\right) + \sqrt{|U_5(1-U_5)\sin(2\pi 1.05/(U_5+.05))|} + \epsilon, \end{aligned}$$

where $\epsilon \sim N(0, .5^2)$, and $U_i \overset{iid}{\sim} N(0,1), i = 1, \ldots, 12$, and θ a known constant. In this example, we will explore the test implemented by this function when testing $H_0 : m(\mathbf{U}) = m_1(\mathbf{U}_{-(2)})$, where $\mathbf{U}_{-(2)}$ represents all covariates except that of index 2. We will use $\theta = 0, 1$ and 2, for when $\theta = 0$ we have the null hypothesis, and when $\theta > 0$ we are under the alternative and expect the p value to be smaller as θ increases. The following code will run the test on simulated data.

```
R> set.seed(1)
R> runs <- 1000
R> p <- 9
R> n <- 100
R> d <- 12
R> result <- c(0, 0, 0)
R> for (index in 1:runs) {
+    for (theta in 0:2) {
+      U <- matrix(0, n, d)
+      for (i in 1:d) {
+        U[, i] <- rnorm(n)
+      }
+      Y <- theta/4 * (sin(2 * U[, 2]) + 2 * exp(- 16 * U[, 2]^2)) +
+        sqrt(abs(U[, 5] * (1 - U[, 5]))) *
+        sin(2 * pi * 1.05 / (U[, 5] + 0.05)) +  rnorm(n, 0, 0.5)
+      if (npmodelcheck(U, Y, 2)$p_value < 0.05) {
+        result[theta + 1] <- result[theta + 1] + 1
+      }
+    }
+  }
R> result/runs

[1] 0.051 0.405 0.954
```

The vector `result` contains the rejection rates, after 1000 simulation runs, of the test for $\theta = 0, 1, 2$ in its positions 1, 2 and 3 respectively. We see that the average rejection in the first position of `result`, which corresponds to the type I error, is 0.051, suggesting that the level of the test is very close to the nominal level 0.05. On the other hand, the rejections for $\theta = 1$ and 2 are higher, indicating an increase in power of the test under alternatives. For comparison purposes, the generalized likelihood ratio test (GLRT) of Fan and Jiang (2005) was also evaluated with the same settings, as it is one of the most recent tests that can deal with nonparametric hypothesis testing. The rejection rates of their test are 0.052, 0.33 and 0.90 for $\theta = 0, 1$ and 2 respectively. These results suggest that the test `npmodelcheck()` not only provides good level of the test, but also higher power than its competitors.

To demonstrate the importance of the dimension reduction techniques, and the improvement on the power of the test they can provide, we ran a similar simulation to the one above with the option `npmodelcheck(U, Y, 2, dim.red = 0)`. In this case, no dimension reduction is performed and the algorithm computes a nonparametric estimate of $m_1(\mathbf{X}_{-(2)})$, where $\mathbf{X}_{-(2)}$ represents all $\mathbf{X}$ covariates except that of index 2, with local polynomial regression, hence suffering from the curse of dimensionality. The output of the vector `result/runs` is

```
[1] 0.079 0.192 0.490
```

It is clear that the rejection rates from the test in this situation are completely different from those with dimension reduction, and the power for detecting departure from the null hypothesis is far from optimal.

The function `npmodelcheck()` also allows the user to input the size p of the window W_i, as defined in (3). This choice is crucial, as the larger p is the stronger the dependence between neighboring cells in the one-way ANOVA, causing decrease in power. On the other hand, we do not advise using p too small due to the instability of the test. For example, the code below tests the significance of the third covariate in $\mathbf{U}$ using $p = 3, 7$ and 15. Note that for $p = 15$ the test fails to detect the significance of the third covariate.

```
R> set.seed(1)
R> n <- 50
R> d <- 4
R> U <- matrix(0, n, d)
R> for (i in 1:d) {
+    U[, i] <- rnorm(n)
+  }
R> Y <- abs(U[, 3])^2 / 3 + rnorm(n, 0, 0.5)
R> npmodelcheck(U, Y, 3, p = 3)

Call:
npmodelcheck.default(X = U, Y = Y, ind_test = 3, p = 3)

P-value of the test: 0.008420023409187571417078

R> npmodelcheck(U, Y, 3, p = 7)

Call:
npmodelcheck.default(X = U, Y = Y, ind_test = 3, p = 7)
```

```
P-value of the test: 0.02166820662532253916055

R> npmodelcheck(U, Y, 3, p = 15)

Call:
npmodelcheck.default(X = U, Y = Y, ind_test = 3, p = 15)

P-value of the test: 0.3451234209169911082427
```

2.2. Hypothesis testing for multiple covariates with NonpModelCheck

It is not unusual to find applications where covariates act together as groups. Model checking in this case can be seen as testing for the significance of all the covariates in a group at once, i.e, testing if the group should be dropped from the model. An interesting example, which has been explored in several papers recently, is DNA microarray data. Such data usually consists of thousands of predictors (genes) but only a few hundred (or less) observations. Studies show that genes can be clustered together in their association with the response, avoiding collinearity among predictors and hence improving model adequacy.

The function `npmodelcheck()` allows the user to test a group of covariates in a single test, i.e., hypothesis tests as in (2). For that purpose, the user only needs to input a vector of indices corresponding to the covariates to be tested for the `ind_test` parameter of `npmodelcheck()`. As an example, consider the situation where 6 covariates are observed but they are clustered together in the following groups: group 1: (X_2, X_4, X_5), group 2: (X_1, X_3) group 3: (X_6). Assume that the covariates $\mathbf{X} = (X_1, \ldots, X_6)$ come from a normal distribution with mean $\mu = (0, \ldots, 0)$ and variance

$$\Sigma = \begin{pmatrix} 1 & 0 & 0.55 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0.6 & 0.7 & 0 \\ 0.55 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0.6 & 0 & 1 & 0.6 & 0 \\ 0 & 0.7 & 0 & 0.6 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Suppose that the response variable Y is associated with $\mathbf{X}$ through the following equation

$$Y = \frac{X_2 + X_4 + X_5}{4} + \epsilon,$$

where $\epsilon \overset{iid}{\sim} N(0, 1)$. The code below tests the significance of group 1.

```
R> library("MASS")
R> set.seed(1)
R> runs <- 1000
R> n <- 100
R> d <- 6
R> result <- 0
```

```
R> sigma <- matrix(c(1, 0, 0.55, 0, 0, 0, 0, 1, 0, 0.6, 0.7, 0, 0.55, 0, 1,
+    0, 0, 0, 0, 0.6, 0, 1, 0.6, 0, 0, 0.7, 0, 0.6, 1, 0, 0, 0, 0, 0, 0, 1),
+    6, 6)
R> for (index in 1:runs) {
+    X <- mvrnorm(n, rep(0, 6), sigma)
+  }
R> Y <- (X[, 2] + X[, 4] + X[, 5]) / 4 + rnorm(n)
R> if (npmodelcheck(X, Y, ind_test = c(2, 4, 5))$p_value < 0.05) {
+    result = result + 1
+  }
R> result/runs

[1] 0.978
```

For comparison purposes, we also evaluated the GLRT test for group 1. The rejection rate obtained was 0.976, demonstrating that even for additive models such as the one simulated, the hypothesis test `npmodelcheck()` achieves power as high as the GLRT.

3. Nonparametric variable/group selection

Variable selection has become an important topic in regression analysis. It uses the assumption of sparseness to build parsimonious models. Generally, when the initial regression model is introduced, we are presented with a large number of predictors. While using many predictors decreases bias, prediction may be poor if the model is built with insignificant ones. In order to have a more accurate model, with enhanced prediction power, procedures of variable selection such as backward elimination or forward selection aim at selecting only those predictors that contribute significantly to the regression model.

The function `npvarselec()` in package **NonpModelCheck**, uses either backward elimination or forward selection to build a model, exploring the connection between model checking and variable selection. Such a procedure was introduced by Zambom and Akritas (2014), where, in backward elimination for example, the algorithm eliminates, at each step, the least significant covariate, as long as its p value is larger than the cut off point in the false discovery rate correction for dependent tests (Benjamini and Yekutieli 2001). Next we briefly describe the algorithms for backward elimination or forward selection.

Assume that the model with d covariates is sparse, in the sense that there exits a subset of indices $I_0 = \{j_1, \ldots, j_{d_0}\} \subset \{1, \ldots, d\}$ such that only the covariates X_j with $j \in I_0$ influence the regression function. Moreover, assume the dimension reduction model of Li (1991), i.e., $m(\mathbf{x}) = g(\mathbf{B}\mathbf{x})$, for a K-dimensional function g, where $\mathbf{B}$ is a $K \times d$ matrix. The backward elimination algorithm, based on the Benjamini and Yekutieli (2001) method for controlling the false discovery rate, is as follows:

1. Obtain p values from testing each of the hypotheses

$$H_0^j : m(\mathbf{x}) = m_1(\mathbf{x}_{(-j)}),\ j = 1, \ldots, d, \tag{7}$$

where $\mathbf{x}_{(-j)} = (x_1, ..., x_{j-1}, x_{j+1}, ..., x_d)$ in the following way:

   (a) Compute the test statistic

$$z_j = \sqrt{n}(\mathrm{MST}_j - \mathrm{MSE}_j)/\sqrt{\frac{2p(2p-1)}{3(p-1)}\hat{\tau}_j^2}$$

   using residuals formed by a local polynomial regression on the variables $\widehat{\mathbf{B}}\mathbf{x}_{(-j)}$, where $\widehat{\mathbf{B}}$ is the $K \times (d-1)$ estimated matrix obtained by applying SIR;

   (b) Compute the p value for H_0^j as $\pi_j = 1 - \Phi(z_j)$.

2. Compute

$$k = \max\left\{ j : \pi_{(j)} \leq \frac{j}{d}\frac{\alpha}{\sum_{l=1}^{d} l^{-1}} \right\} \tag{8}$$

   for a choice of level α, where $\pi_{(1)}, \ldots, \pi_{(d)}$ are the ordered p values. If $k = d$ stop and retain all variables. If $k < d$

   (a) update $\mathbf{x}$ by eliminating the covariate corresponding to $\pi_{(d)}$;

   (b) update d to $d - 1$;

   (c) and proceed to the next step.

3. Repeat Steps 1 and 2, with the updated $\mathbf{x}$ and $\widehat{\mathbf{B}}$.

Note that if the user is trying to run the backward elimination on a very large number of covariates, computational cost may be high. Simulations show that a screening procedure can be applied before running the algorithm, not affecting the power. That is, by eliminating those covariates whose p values (obtained with univariate tests `npmodelcheck()`) are very high, say greater than 0.7 or 0.8, the algorithm does not improve performance, but reduces computational time.

With the same assumptions of sparseness and the dimension reduction model, a forward algorithm for variable selection with FDR corrections can be described as:

1. Obtain p values from testing each of the hypotheses

$$H_0^j : m(x_j) = c, \text{ where } c \text{ is a constant (w.l.o.g. } c = \bar{Y}) \tag{9}$$

   in the following way:

   (a) Compute the test statistic

$$z_j = \sqrt{n}(\mathrm{MST}_j - \mathrm{MSE}_j)/\sqrt{\frac{2p(2p-1)}{3(p-1)}\hat{\tau}_j^2}$$

   using $(\mathbf{Y} - \bar{Y})$ instead of residuals in the high-dimensional one-way ANOVA.

   (b) Compute the p value for H_0^j as $\pi_j = 1 - \Phi(z_j)$.

2. If $\pi_{(1)} > 0.05$, where $\pi_{(1)}, \ldots, \pi_{(d)}$ are the ordered p values, stop and retain no covariates. If $\pi_{(1)} < 0.05$, include in the model the covariate corresponding to $\pi_{(1)}$:

   (a) let $x^* = x_j$ and $q = 1$;
   (b) update $\mathbf{x}$ by eliminating the covariate corresponding to $\pi_{(1)}$;
   (c) and update d to $d - 1$.

3. Find the smallest p value to include: obtain p values from testing each of the hypotheses

$$H_0^j : m(\mathbf{x}^*, x_j) = m_1(\mathbf{x}^*),\ j = 1, \ldots, d \tag{10}$$

   in the following way:

   (a) Compute the test statistic

$$z_j = \sqrt{n}(\mathrm{MST}_j - \mathrm{MSE}_j) / \sqrt{\frac{2p(2p-1)}{3(p-1)} \hat{\tau}_j^2}$$

   using residuals formed by a local polynomial regression on the variables $\widehat{\mathbf{B}}\mathbf{x}^*$, where $\widehat{\mathbf{B}}$ is the $K \times q$ estimated matrix obtained by applying SIR.

   (b) Compute the p value for H_0^j as $\pi_j = 1 - \Phi(z_j)$.

4. Temporarily include in the model the covariate with smallest p value:
   (a) let $x^* = (\mathbf{x}^*, x_{(1)})$, where $x_{(1)}$ is the covariate corresponding to $\pi_{(1)}$; and let $q = q + 1$;
   (b) update $\mathbf{x}$ by eliminating the covariate corresponding to $\pi_{(1)}$;
   (c) update d to $d - 1$;
   (d) and proceed to the next step.

5. Check if all the covariates in the present model are significant according to FDR:

   (a) Obtain p values from testing each of the hypotheses

$$H_0^j : m(\mathbf{x}^*) = m_1(\mathbf{x}^*_{(-j)}),\ j = 1, \ldots, q, \tag{11}$$

   where $\mathbf{x}_{(-j)} = (x_1, ..., x_{j-1}, x_{j+1}, ..., x_q)$, in the following way:

      i. Compute the test statistic

$$z_j = \sqrt{n}(\mathrm{MST}_j - \mathrm{MSE}_j) / \sqrt{\frac{2p(2p-1)}{3(p-1)} \hat{\tau}_j^2}$$

      using residuals formed by a local polynomial regression on the variables $\widehat{\mathbf{B}}\mathbf{x}^*_{(-j)}$, where $\widehat{\mathbf{B}}$ is the $K \times (q-1)$ estimated matrix obtained by applying SIR.

      ii. Compute the p value for H_0^j as $\pi_j = 1 - \Phi(z_j)$.

   (b) Compute

$$k = \max\left\{ j : \pi_{(j)} \leq \frac{i}{q} \frac{\alpha}{\sum_{l=1}^{q} l^{-1}} \right\} \tag{12}$$

   for a choice of level α, where $\pi_{(1)}, \ldots, \pi_{(q)}$ are the ordered p values. If $k = q = d$ stop and retain all covariates. If $k = q < d$ go to Step 3. If $k < q$

   i. remove from $\mathbf{x}^*$ the covariate temporarily added in Step 4, and let $q = q - 1$;
   ii. stop and retain $\mathbf{x}^*$.

This forward procedure, at each step, first finds a candidate covariate to enter the model. The candidate is the covariate that yields the smallest p value from test `npmodelcheck()`, taking into consideration all covariates that are already in the model. Then, the candidate covariate is only retained if the new model, with the candidate covariate included, is significant, i.e., if the test of each covariate in the new model produces a significant p value according to FDR corrections.

Note that there may be a covariate which, when added to the model, would result in a new significant model. However, when tested as candidate, it is not the one that yields the smallest p value. Hence, a different forward procedure can be defined, adding to the model only the covariate that would result in the "most significant" new model.

1. Obtain p values from testing each of the hypotheses

$$H_0^j : m(x_j) = c, \text{ where } c \text{ is a constant (w.l.o.g. } c = \bar{Y}) \tag{13}$$

   in the following way:

   (a) Compute the test statistic

$$z_j = \sqrt{n}(\mathrm{MST}_j - \mathrm{MSE}_j) / \sqrt{\frac{2p(2p-1)}{3(p-1)}\hat{\tau}_j^2}$$

   using $(\mathbf{Y} - \bar{Y})$ instead of residuals in the high-dimensional one-way ANOVA.

   (b) Compute the p value for H_0^j as $\pi_j = 1 - \Phi(z_j)$.

2. If $\pi_{(1)} > 0.05$, where $\pi_{(1)}, \ldots, \pi_{(d)}$ are the ordered p values, stop and retain no covariates. If $\pi_{(1)} < 0.05$, include in the model the covariate corresponding to $\pi_{(1)}$:

   (a) let $x^* = x_j$ and $q = 1$;
   (b) update $\mathbf{x}$ by eliminating the covariate corresponding to $\pi_{(1)}$;
   (c) and update d to $d - 1$.

3. Add the covariate that produces the "most significant" new model:

   (a) For k in $1, \ldots, d$:

      i. Let $\mathbf{u}^* = (\mathbf{x}^*, x_j)$. Obtain p values from testing each of the hypotheses

$$H_0^j : m(\mathbf{u}^*) = m_1(\mathbf{u}^*_{(-j)}), \; j = 1, \ldots, q+1 \tag{14}$$

      where $\mathbf{u}^*_{(-j)}$ the vector $\mathbf{u}^*$ without covariate $\mathbf{u}^*_j$, in the following way:

      A. Compute the test statistic

$$z_j = \sqrt{n}(\mathrm{MST}_j - \mathrm{MSE}_j) / \sqrt{\frac{2p(2p-1)}{3(p-1)}\hat{\tau}_j^2}$$

      using residuals formed by a local polynomial regression on the variables $\widehat{\mathbf{B}}\mathbf{u}^*_{(-j)}$, where $\widehat{\mathbf{B}}$ is the $K \times q$ estimated matrix obtained by applying SIR.

B. Compute the p value for H_0^j as $\pi_j = 1 - \Phi(z_j)$.

ii. Compute

$$\ell = \max\left\{ j : \pi_{(j)} \leq \frac{i}{q} \frac{\alpha}{\sum_{m=1}^{q} m^{-1}} \right\} \tag{15}$$

for a choice of level α, where $\pi_{(1)}, \ldots, \pi_{(q)}$ are the ordered p values. If $\ell = q$ and if $\pi_{(q)}$ is less than the $\pi_{(q)}$ produced by the other candidates, set the covariate k as the new candidate.

(b) If no covariate was set as candidate, stop. Otherwise

i. include in x^* the candidate covariate and set $q = q + 1$;

ii. update $\mathbf{x}$ by eliminating the candidate covariate and set $d = d - 1$;

iii. return to Step 3a.

When it is believed that covariates come in groups, as for example clusters of genes in microarrays, one fundamental goal is to select those groups (clusters) that have a significant effect on the response variable Y. It is possible to use algorithms similar to those described above in order to run group selection, the only difference would be to use the test of the whole group in each step (see Zambom and Akritas 2015 for details).

3.1. Nonparametric variable selection with NonpModelCheck

The procedures for variable selection described in the previous section are implemented in function `npvarselec()` in package **NonpModelCheck**. The arguments for this function are basically the same as the ones needed for `npmodelcheck()`, except that the user is required to specify the type of the selection. The options are `"backward"`, `"forward"` and `"forward2"`, corresponding to the 3 algorithms described in Section 3, in the order they appear.

To demonstrate the use of this function, a study was performed in the following way. A data set with sample size $n = 110$ was generated from the model $Y = \mathbf{X}\boldsymbol{\beta} + \epsilon$, where $\epsilon \overset{iid}{\sim} N(0, 3^2)$ and $\mathbf{X} = (X_1, \ldots, X_{25}) \sim N(\mathbf{0}, \Sigma)$, with $\Sigma_{i,j} = I$ a diagonal matrix.

```
R> library("mnormt")
R> set.seed(1)
R> n <- 110
R> p <- 9
R> d <- 25
R> Cov_Matrix <- diag(d)
R> X <- rmnorm(n, rep(0, d), Cov_Matrix)
R> beta <- c(3, 1.5, 0, 0, 2, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0,
+    0, 0, 0, 0, 0)
R> Y <- as.numeric(X % * % beta  + rnorm(n, 0, 3))
R> npvarselec(X, Y, method = "backward", p = 9, degree.pol = 1,
+    kernel.type = "epanech", bandwidth = "CV")

-------------------------------------------------------------
Iter. | Variables in the model
1     | 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
```

```
2      | 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
3      | 1 2 3 4 5 6 7 8 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
4      | 1 2 3 4 5 6 7 8 10 11 12 13 14 15 16 17 18 19 20 21 22 23
5      | 1 2 4 5 6 7 8 10 11 12 13 14 15 16 17 18 19 20 21 22 23
6      | 1 2 5 6 7 8 10 11 12 13 14 15 16 17 18 19 20 21 22 23
7      | 1 2 5 6 7 8 10 11 12 13 15 16 17 18 19 20 21 22 23
8      | 1 2 5 6 7 8 10 11 12 13 15 16 17 18 19 20 21 22
9      | 1 2 5 6 7 10 11 12 13 15 16 17 18 19 20 21 22
10     | 1 2 5 6 7 10 11 12 13 15 16 17 18 19 20 21
11     | 1 2 5 6 7 10 11 12 13 15 17 18 19 20 21
12     | 1 2 5 6 7 10 11 12 13 15 17 19 20 21
13     | 1 2 5 7 10 11 12 13 15 17 19 20 21
14     | 1 2 5 7 10 12 13 15 17 19 20 21
15     | 1 2 5 7 12 13 15 17 19 20 21
16     | 1 2 5 7 12 13 15 17 19 20
17     | 1 2 5 7 13 15 17 19 20
18     | 1 2 5 7 13 15 17 19
19     | 1 2 5 7 13 15 17
20     | 1 2 5 7 15 17
21     | 1 2 5 7 17
-------------------------------------------------------------


Number of Covariates Selected: 5

Covariate(s) Selected:
---------------------------
Covariate Index  |  p-value
          1      |  0
          2      |  0.00082
          5      |  6.4e-11
          7      |  6e-11
         17      |  0
---------------------------
```

The output of `npvarselec()` is the step by step result at each iteration of the algorithm. We see that in this case, the selected predictors were exactly the ones that really have an effect on the response variable.

In order to assess the performance of the variable selection procedure using the three different algorithms, we simulated the above linear model 100 times and computed the average number of correctly selected predictors and the average number of incorrectly selected predictors. We also simulated this linear model when there is dependence between the covariates, that is when $\Sigma_{i,j} = 0.5^{|i-j|}$. The performance of these algorithms was compared to recent non-linear variable selection methods COSSO (R package **cosso**; Zhang and Lin 2013) and CISE (MATLAB package **cise**; accompanying material for Chen, Zou, and Cook 2010 available at `http://www.stat.nus.edu.sg/~stacx/cise.zip`). The results are shown in Table 1. Those for CISE were obtained using Octave (Eaton, Bateman, Hauberg, and Wehbring 2015). In

	$\Sigma = I$		$\Sigma = (0.5^{\lvert i-j \rvert})$	
Method	Correct	Incorrect	Correct	Incorrect
ANOVA backward	4.70	0.58	4.10	0.58
ANOVA forward	4.50	0.68	4.00	0.65
ANOVA forward2	4.50	0.75	4.10	0.74
COSSO	4.10	0.99	3.90	1.00
CISE	3.15	0.40	3.28	0.74

Table 1: Comparisons for a linear model with $d = 25$ and $n = 110$.

	g_1		g_2	
Method	Correct	Incorrect	Correct	Incorrect
ANOVA backward	1.00	0.13	1.90	0.09
ANOVA forward	1.00	0.19	2.00	0.10
ANOVA forward2	1.00	0.18	2.00	0.11
COSSO	0.55	1.60	0.91	1.10
CISE	0.27	1.76	0.37	1.65

Table 2: Comparisons using nonlinear models with $d = 8$ and $n = 40$.

Table 2, 100 data sets of size $n = 40$ were generated from the models $Y = g_\ell(\mathbf{X}) + \epsilon$, $\ell = 1, 2$, where $\epsilon \sim N(0, 0.3^2)$, the dimension of $\mathbf{X}$ is $d = 8$, and

$$g_1(\mathbf{x}) = \sin(\pi x_1), \quad g_2(\mathbf{x}) = \sin(3/4\pi x_1) - 3\Phi(-|x_5|^3).$$

In both linear and nonlinear models, the three proposed algorithms from `npvarselec()` had similar results. For the linear models, although CISE selected, in average, less incorrect predictors, it missed, on average, one more than the proposed algorithms. On the other hand, all three algorithms substantially outperformed CISE and COSSO in the case of nonlinear models. COSSO results were inferior in the simulated linear and nonlinear models compared to CISE and the proposed algorithms.

A demonstration of the nonparametric group selection algorithms using `npvarselec()` can be found in Section 4.

4. Group selection applied to a real data set

Microarray experiments generate large datasets which generally contain just a few samples but expression values for thousands of genes. In such studies, data analysts often consider studying clusters (or groups) of genes that act together, since their collective expression may have a stronger association with the response. Thus, it is crucial not only to find these clusters, but also to select those which significantly influence the outcome variable. Function `group.npvarselec()` in package **NonpModelCheck** performs group selection with algorithms similar to those in Section 3. All three types of algorithm are available: backward, forward and forward2. The difference is that at each step, all the covariates of a group are tested at the same time. Basically, the arguments of the function are the same as those for `npvarselec()`, except for two extra ones. First, the user needs to specify the indices of the variables in each group in an argument of the type '`list`'. Second, when testing group i, instead of computing the residuals based on $\hat{m}_1(\mathbf{X}_{(-I_i)})$, where I_j is the set of the indices of covariates in group i, the user may prefer to fit $\hat{m}_1(S_1, \ldots, S_{i-1}, S_{i+1}, \ldots, S_d)$, where $S_j, j = 1, \ldots, d$ is the first supervised principal component of group j, with d being the total number of groups. If the argument `fitSPC` is set to true, the latter is fit, otherwise the former computation is done.

A demonstration of the use of this function will be done using the prostate cancer dataset of Singh *et al.* (2002), available at `http://icos.cs.nott.ac.uk/datasets/microarray.html`. The data set is composed of gene expressions of 52 prostate tumor and 50 non-tumor prostate samples, obtained from the Affymetrix technology. Before applying the group selection algorithm, the clusters need to be formed. To do so, the procedure `wilma()` from package **supclust** was used. The following code performs a forward algorithm for group variable selection on the 60 clusters formed by `wilma()` (note that applying `wilma()` to a fixed dataset twice may result in different output clusters due to the randomness built in its algorithm).

```
R> library("supclust")
R> data <- read.table("prostate.txt")
R> Y <- as.numeric(data[, 1] == "normal")
R> X <- matrix(as.matrix(data[, (2:dim(data)[2])]),
+    ncol = ncol(data[, (2:dim(data)[2])]), dimnames = NULL)
R> n <- dim(X)[1]
R> d <- 60
R> set.seed(1)
R> fit <- wilma(X, Y, d, once.per.clust = TRUE)
R> groups <- vector("list", d)
R> for (i in 1:d) {
+    groups[[i]] <- fit$clist[[i]]
+  }
R> group.npvarselec(X, Y, groups, method = "forward2", fitSPC = TRUE)
```

```
-------------------------------------------------------------
Iter. | Variables in the model
1     | 1
2     | 1 14
3     | 1 14 22
4     | 1 14 22 25
5     | 1 14 22 25 26
6     | 1 14 22 25 26 31
7     | 1 14 22 25 26 31 41
8     | 1 14 22 25 26 31 41 54
9     | 1 14 22 25 26 31 41 54 58
10    | 1 14 22 25 26 31 41 54 58 15
11    | 1 14 22 25 26 31 41 54 58 15 52
12    | 1 14 22 25 26 31 41 54 58 15 52 59
13    | 1 14 22 25 26 31 41 54 58 15 52 59 56
14    | 1 14 22 25 26 31 41 54 58 15 52 59 56 57
15    | 1 14 22 25 26 31 41 54 58 15 52 59 56 57 8
16    | 1 14 22 25 26 31 41 54 58 15 52 59 56 57 8 29
17    | 1 14 22 25 26 31 41 54 58 15 52 59 56 57 8 29 5
-------------------------------------------------------------
```

```
Number of Groups Selected: 17

Groups Selected:
---------------------------
Group Index  |  p-value
         1   |  0.0032
        14   |  6.7e-09
        22   |  3.5e-07
        25   |  4.2e-07
        26   |  2.2e-07
        31   |  2.9e-07
        41   |  5.2e-07
        54   |  2.5e-09
        58   |  4.2e-11
        15   |  0
        52   |  4.3e-10
        59   |  2.4e-09
        56   |  4.3e-07
        57   |  1.9e-07
         8   |  0.0031
        29   |  9.3e-07
         5   |  3.7e-11
---------------------------
```

Function `group.npvarselec()` returns the indices of the selected groups and the corresponding p values. Note that all the groups in the final model have p values that are significant according to the false discovery rate corrections.

Acknowledgments

This paper was partially supported with grant 2012/10808-2 FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo).

References

Abramovich F, Benjamini Y, Donoho D, Johnstone I (2006). "Adapting to Unknown Sparsity by Controlling the False Discovery Rate." *The Annals of Statistics*, **34**(2), 584–653. doi:10.1214/009053606000000074.

Ait-Sahalia Y, Bickel PJ, Stoker TM (2001). "Goodness-of-Fit Tests for Kernel Regression with an Application to Option Implied Volatilities." *Journal of Econometrics*, **105**, 363–412. doi:10.1016/s0304-4076(01)00091-4.

Antoniadis A, Gijbels I, Lambert-Lacroix S (2014). "Penalized Estimation in Additive Varying Coefficient Models Using Grouped Regularization." *Statistical Papers*, **55**, 727–750. doi:10.1007/s00362-013-0522-1.

Bair E, Hastie T, Paul D, Tibshirani R (2006). "Prediction by Supervised Principal Components." *Journal of the American Statistical Association*, **101**, 119–137. doi:10.1198/016214505000000628.

Benjamini Y, Hochberg Y (1995). "Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing." *Journal of the Royal Statistical Society B*, **57**(1), 289–300.

Benjamini Y, Yekutieli D (2001). "The Control of the False Discovery Rate in Multiple Testing Under Dependency." *The Annals of Statistics*, **29**, 1165–1188. doi:10.1214/aos/1013699998.

Bowman AW, Azzalini A (2014). ***sm**: Nonparametric Smoothing Methods*. R package version 2.2-5.4, URL https://CRAN.R-project.org/package=sm.

Breheny P (2017). ***ncvreg**: Regularization Paths for SCAD and MCP Penalized Regression Models*. R package version 3.8-0, URL https://CRAN.R-project.org/package=ncvreg.

Breheny P, Huang J (2011). "Coordinate Descent Algorithms for Nonconvex Penalized Regression, with Applications to Biological Feature Selection." *The Annals of Applied Statistics*, **5**(1), 232–253.

Candes E, Tao T (2005). "The Dantzig Selector: Statistical Estimation When p Is Much Larger than n." *The Annals of Statistics*, **35**(6), 2313–2351. doi:10.1214/009053606000001523.

Chen X, Zou C, Cook RD (2010). "Coordinate-Independent Sparse Sufficient Dimension Reduction and Variable Selection." *The Annals of Statistics*, **38**(6), 3696–3723. doi:10.1214/10-AOS826.

Claeskens G (2004). "Restriced Likelihood Ratio Lack-of-Fit Test Using Mixed Spline Models." *Journal of the Royal Statistical Society B*, **66**, 909–926. doi:10.1111/j.1467-9868.2004.05421.x.

Deegan Jr J (1976). "The Consequences of Model Misspecification in Regression Analysis." *Multivariate Behavioral Research*, **11**(2), 237–248. doi:10.1207/s15327906mbr1102_9.

Delgado MA, Manteiga WG (2001). "Significance Testing in Nonparametric Regression Based on the Bootstrap." *The Annals of Statistics*, **29**(5), 1469–1507. doi:10.1214/aos/1013203462.

Eaton JW, Bateman D, Hauberg S, Wehbring R (2015). *GNU Octave Version 4.0.0 Manual: A High-Level Interactive Language for Numerical Computations*. URL https://www.gnu.org/software/octave/doc/interpreter.

Efron B, Hastie T, Johnstone I, Tibshirani R (2004). "Least Angle Regression." *The Annals of Statistics*, **32**, 407–499. doi:10.1214/009053604000000067.

Fan J (1992). "Design-Adaptive Nonparametric Regression." *Journal of the American Statistical Association*, **87**, 998–1004. doi:10.1080/01621459.1992.10476255.

Fan J (1993). "Local Linear Regression Smoothers and Their Minimax Efficiency." *The Annals of Statistics*, **21**, 196–216. doi:10.1214/aos/1176349022.

Fan J, Gijbels I (1992). "Variable Bandwidth and Local Linear Regression Smoothers." *The Annals of Statistics*, **20**, 2008–2036. doi:10.1214/aos/1176348900.

Fan J, Gijbels I (1995). "Data-Driven Bandwidth Selection in Local Polynomial Fitting." *Journal of the Royal Statistical Society B*, **72**(2), 371–394.

Fan J, Jiang J (2005). "Nonparametric Inferences for Additive Models." *Journal of the American Statistical Association*, **100**, 890–907. doi:10.1198/016214504000001439.

Fan J, Li R (2001). "Variable Selection via Nonconcave Penalized Likelihood and Its Oracle Properties." *Journal of the American Statistical Association*, **96**, 1348–1360. doi:10.1198/016214501753382273.

Fan J, Zhang CM, Zhang J (2001). "Generalized Likelihood Ratio Statistics and Wilks Phenomenon." *The Annals of Statistics*, **29**, 153–193. doi:10.1214/aos/996986505.

Fan Y, Li Q (1996). "Consistent Model Specification Tests: Omitted Variables and Semiparametric Functional Forms." *Econometrica*, **64**, 865–890. doi:10.2307/2171848.

Friedman J, Hastie T, Tibshirani R (2010). "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software*, **33**(1), 1–22. doi:10.18637/jss.v033.i01.

Gijbels I, Verhasselt A, Vrinssen I (2015). "Variable Selection Using P-Splines." *Wiley Interdisciplinary Reviews: Computational Statistics*, **7**, 1–20. doi:10.1002/wics.1327.

Hartford A, Davidian M (2000). "Consequences of Misspecifying Assumptions in Nonlinear Mixed Effects Models." *Computational Statistics & Data Analysis*, **34**, 139–164. doi:10.1016/s0167-9473(99)00076-6.

Hastie T (2016). ***gam**: Generalized Additive Models.* R package version 1.14, URL https://CRAN.R-project.org/package=gam.

Hastie T, Efron B (2013). ***lars**: Least Angle Regression, Lasso and Forward Stagewise.* R package version 1.2, URL https://CRAN.R-project.org/package=lars.

Huang J, Horowitz JL, Wei F (2010). "Variable Selection in Nonparametric Additive Models." URL http://faculty.wcas.northwestern.edu/~jlh951/papers/HHW-npam.pdf.

Larson HJ, Bancroft TA (1963). "Biases in Prediction by Regression for Certain Incompletely Specified Models." *Biometrika*, **50**(3/4), 391–402. doi:10.1093/biomet/50.3-4.391.

Lavergne P, Vuong Q (2000). "Nonparametric Significance Testing." *Econometric Theory*, **16**, 576–601. doi:10.1017/s0266466600164059.

Li KC (1991). "Sliced Inverse Regression for Dimension Reduction." *Journal of the American Statistical Association*, **86**, 316–327. doi:10.2307/2290563.

Li R, Liang H (2008). "Variable Selection in Semiparametric Regression Modeling." *The Annals of Statistics*, **36**, 261–286. doi:10.1214/009053607000000604.

Loader C (2013). ***locfit**: Local Regression, Likelihood and Density Estimation.* R package version 1.5-9.1, URL `https://CRAN.R-project.org/package=locfit`.

Masry E (1996). "Multivariate Local Polynomial Regression for Time Series: Uniform Strong Consistency and Rates." *Journal of Time Series Analysis*, **17**, 571–599. `doi:10.1111/j.1467-9892.1996.tb00294.x`.

Meinshausen N, Meier L, Bühlmann P (2009). "p-Values for High-Dimensional Regression." *Journal of the American Statistical Association*, **104**, 1671–1681. `doi:10.1198/jasa.2009.tm08647`.

Ojeda Cabrera JL (2012). ***locpol**: Kernel Local Polynomial Regression.* R package version 0.6-0, URL `https://CRAN.R-project.org/package=locpol`.

Ramsey J (1969). "Tests for Specification Errors in Classical Least Squares Regression Analysis." *Journal of the Royal Statistical Society B*, **31**(2), 350–371.

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL `https://www.R-project.org/`.

Ruppert D, Wand MP (1994). "Multivariate Locally Weighted Least Squares Regression." *The Annals of Statistics*, **22**, 1346–1370. `doi:10.1214/aos/1176325632`.

Singh D, Febbo P, Ross K, Jackson D, Manola J, Ladd C, Tamayo P, Renshaw A, D'Amico A, Richie J, Lander E, Loda M, Kantoff P, Golub T, Sellers W (2002). "Gene Expression Correlates of Clinical Prostate Cancer Behavior." *Cancer Cell*, **1**(2), 203–209. `doi:10.1016/s1535-6108(02)00030-2`.

Stone CJ (1977). "Consistent Nonparametric Regression." *The Annals of Statistics*, **5**, 595–645. `doi:10.1214/aos/1176343886`.

Storlie CB, Bondell HD, Reich BJ, Zhang HHH (2011). "Surface Estimation, Variable Selection, and the Nonparametric Oracle Property." *Statistica Sinica*, **21**(2), 679–705. `doi:10.5705/ss.2011.030a`.

Tibshirani R (1996). "Regression Shrinkage and Selection via the Lasso." *Journal of the Royal Statistical Society B*, **58**(1), 267–288.

Wand M (2015). ***KernSmooth**: Functions for Kernel Smoothing for Wand & Jones (1995).* R package version 2.23-15, URL `https://CRAN.R-project.org/package=KernSmooth`.

Wang H, Xia Y (2009). "Shrinkage Estimation of the Varying Coefficient Model." *Journal of the American Statistical Association*, **104**(486), 747–757. `doi:10.1198/jasa.2009.0138`.

White H (1981). "Consequences and Detection of Misspecified Nonlinear Regression Models." *Journal of the American Statistical Association*, **76**(374), 419–433. `doi:10.2307/2287845`.

Wood S (2017). ***mgcv**: Mixed GAM Computation Vehicle with GCV/AIC/REML Smoothness Estimation.* R package version 1.8-17, URL `https://CRAN.R-project.org/package=mgcv`.

Wood SN (2006). *Generalized Additive Models: An Introduction with R*. Chapman & Hall/CRC, Boca Raton.

Zambom AZ (2015). ***NonpModelCheck**: Model Checking and Variable Selection in Nonparametric Regression.* R package version 2.0, URL https://CRAN.R-project.org/package=NonpModelCheck.

Zambom AZ, Akritas MG (2014). "Nonparametric Lack-of-Fit Testing and Consistent Variable Selection." *Statistica Sinica*, **24**, 1837–1858. doi:10.5705/ss.2013.112.

Zambom AZ, Akritas MG (2015). "Nonparametric Significance Testing and Group Variable Selection." *Journal of Multivariate Analysis*, **133**, 51–60. doi:10.1016/j.jmva.2014.08.014.

Zhang HH, Lin CY (2013). ***cosso**: Fit Regularized Nonparametric Regression Models Using COSSO Penalty.* R package version 2.1-1, URL https://CRAN.R-project.org/package=cosso.

Zou H (2006). "The Adaptive Lasso and Its Oracle Properties." *Journal of the American Statistical Association*, **101**, 1418–1429. doi:10.1198/016214506000000735.

A. A new function for local polynomial fitting in R

Local polynomial regression is one of the most powerful nonparametric curve fitting approaches. It was introduced by Stone (1977) and has since been used in several applications. Its advantages over the simple Nadaraya-Watson estimator made it prominent in modern research when estimating surfaces without parametric assumptions. For example, it does not suffer from boundary effects, reduces bias, has the ability of design adaptation and is uniformly consistent over compact subsets (see Fan 1992, Fan 1993, Fan and Gijbels 1992, Ruppert and Wand 1994, Masry 1996, among others).

Due to its fundamental role in nonparametric estimation of regression curves, it is generally found in most mathematical/statistical software. Specifically in R, local polynomial estimation is implemented in functions `loess` (**stats**), `locpoly` (**KernSmooth**; Wand 2015), `locfit` (**locfit**; Loader 2013), `locpol` (**locpol**; Ojeda Cabrera 2012) and `sm.regression` (**sm**; Bowman and Azzalini 2014). However, not all the advantages of this procedure are explored in these functions: `locpoly()` and `locpol()` only admit the univariate case, `sm.regression()` works with 1 or 2 covariates, `loess()` fits up to 4 predictors and up to degree 2, and `locfit()` is complicated to use: It requires the user to input the predictors one by one and does not have a built-in procedure for automatically selecting the smoothing parameter.

Package **NonpModelCheck** provides a new function for local polynomial regression fitting, namely `localpoly.reg()`, which is extremely easy to use and implements clearly the classical approach. Moreover, it automatically identifies the number of predictors in the input matrix, provides several kernel type options for weights and different built-in procedures for bandwidth selection. Besides, it does not only estimate the regression function, but also its derivatives. The call of the function is

```
  localpoly.reg(X, Y, points = NULL, bandwidth = "CV", gridsize = 30,
    degree.pol = 0, kernel.type = "epanech", deriv = 0).
```

It is well known that the goodness of the fit highly depends on the type of kernel weight and degree of polynomial used. `localpoly.reg()` offers a variety of kernel options to the user, including `"box"`, `"trun.normal"`, `"gaussian"`, `"epanech"`, `"biweight"`, `"triweight"` and `"triangular"`. For the univariate case (when there is only one predictor), the choice of degree of the polynomial is unlimited, providing further bias reduction when desired. For the multivariate case where several predictors are available, the polynomial can be chosen up to the second degree. In a simulated example, we generated a uniform predictor $X \overset{iid}{\sim} U(-4, 4)$ and set the response $Y = \sin(\pi X)X^2 + \epsilon$, where $\epsilon \overset{iid}{\sim} N(0, 1)$ and fitted a nonparametric regression using `localpoly.reg()` and `loess()` with degree 2 polynomial. In another example, we generated $X \overset{iid}{\sim} U(-3, 10)$ and set $Y = \sqrt{|X(7-X)|}\cos(1/6\pi X) + \epsilon$, where $\epsilon \overset{iid}{\sim} N(0, .2)$, and fitted 2 different types of kernel in another simulation, comparing it to the fit of `sm.regression()`. The code is given below and the comparative plots in Figure 1.

```
R> set.seed(1)
R> X <- runif(180, - 4, 4)
R> Y <- sin(pi * X) * X^2 + rnorm(180, 0, 1)
R> fit <- localpoly.reg(X, Y, degree.pol = 2)
R> fit2 <- loess(Y ~ X)
```

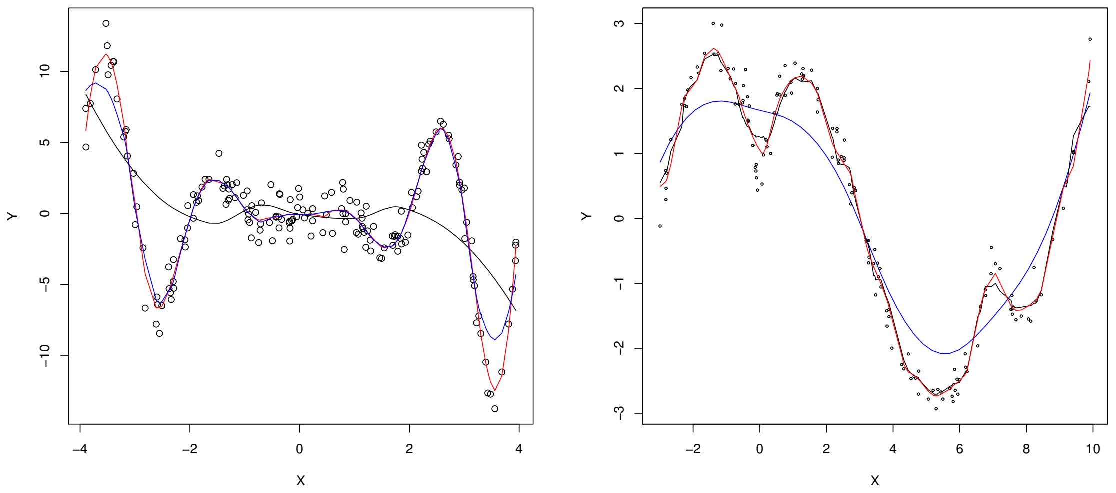

Figure 1: Left: Estimated curves using `localpoly.reg()` with `degree = 2` and cross validation to select the bandwidth (red), `loess()` by default (black) and `loess()` with span 0.2 (blue). Right: Estimated curves using `localpoly.reg()` with `kernel = "box"` (black) and `kernel = "epanech"` (red), and `sm.regression()` using bandwidth selection method `"df"` (blue).

```
R> fit3 <- loess(Y ~ X, span = 0.2)
R> plot(X, Y)
R> lines(X[order(X)], fit$predicted[order(X)], col = 2)
R> lines(X[order(X)], fit2$fitted[order(X)], col = 1)
R> lines(X[order(X)], fit3$fitted[order(X)], col = 4)
R> library("sm")
R> set.seed(11)
R> X <- runif(150, - 3, 10)
R> Y <- sqrt(abs(X * (7 - X))) * (cos(1/6 * pi * X)) + rnorm(150, 0, 0.2)
R> h <- h.select(X, Y)
R> sm.regression(X, Y, h = h, col = 4)
R> fit <- localpoly.reg(X, Y, kernel.type = "box")
R> lines(X[order(X)], fit$predicted[order(X)])
R> fit2 <- localpoly.reg(X, Y, kernel.type =  "epanech")
R> lines(X[order(X)], fit2$predicted[order(X)], col = 2)
```

Note from Figure 1 on the left that when the default is used, `loess()` does not capture all features of the regression, so the argument `span` has to be input manually or chosen adequately, visually or by another procedure, in order to produce a better fit. It can be seen from Figure 1 on the right that using different kernel types as weights yields different fits, specifically in this case, the epanech kernel produces a smoother curve compared to the box kernel.

A fundamental advantage of `localpoly.reg()` over the other functions is that it provides bandwidth selection procedures built-in, allowing the user to choose according to the situation, or even compare different methods if needed. In the univariate case, the available methods for bandwidth selection are leave-one-out cross validation, generalized cross validation and adaptive/variable bandwidth (binned) selection. The adaptive/variable bandwidth selection is a modified version of the one proposed by Fan and Gijbels (1995): first the interval of estimation (range of X) is divided into $[1.5n/(10\log(n))]$ subintervals; then, a leave-one-out cross validation is performed using only the observations in each subinterval, selecting its corresponding bandwidth; finally the resulting bandwidth step function is smoothed using a local polynomial fit with leave-one-out cross validation to select the pilot bandwidth.

The options for bandwidth selection are all condensed in the `bandwidth` argument of function `localpoly.reg()`. To request leave-one-out cross validation, just set the `bandwidth` argument to `"CV"`, while for generalized cross validation, set it to `"GCV"` and for adaptive bandwidth selection set it to `"Adp"`. The following code generates 3 random simulations, demonstrating the use of the different bandwidth selection methods in `localpoly.reg()`. Figures 2 and 3 are the output of the following code, where `localpoly.reg()` fits can be compared to the ones of `locpoly()`, `locfit()`, `locpol()` and `sm.regression()`.

```
R> library("KernSmooth")
R> library("locfit")
R> set.seed(123456789)
R> X <- runif(150, 2, 12)
R> Y <- exp(1/X) * (1 - 1/X) / sqrt(1 + abs(8 - X)) / 4.5 +
+    rnorm(150, 0, .01)
R> fit <- localpoly.reg(X, Y, bandwidth = "CV", degree.pol = 2,
+    kernel.type = "triweight")
R> fit2 <- locpoly(X, Y, bandwidth = fit$bandwidth, degree = 2)
R> fit3 <- locfit(Y ~ lp(X, deg = 2))
R> plot(fit3, get.data = TRUE)
R> lines(fit2$x, fit2$y, col = 4)
R> lines(X[order(X)], fit$predicted[order(X)], col = 2)
R> library("locpol")
R> set.seed(123)
R> X <- c(runif(150, 2, 12))
R> Y <- exp(1/X) / sqrt(2 + sqrt(abs(pi * (6 - X)))) + rnorm(150, 0, .01)
R> d <- data.frame(X)
R> d$Y <- Y
R> fit <- locpol(Y ~ X, d)
R> plot(X, Y)
R> lines(fit$xeval, fit$lpFit[, 2])
R> fit2 <- localpoly.reg(X, Y, bandwidth = "GCV", degree.pol = 1)
R> lines(X[order(X)], fit2$predicted[order(X)], col = 2)
R> set.seed(123)
R> X <- c(runif(30, 0, 15), runif(100, 15, 25), runif(50, 25, 55),
+    runif(300, 55, 230))
R> Y <- c(rnorm(30, 0, 0.1), sin(1/2 * pi * X[31:130]) + rnorm(100, 0, 0.3),
+    rnorm(50, 0, 0.15), cos(1/25 * pi * X[181:480]) + rnorm(300, 0, 0.2))
R> fit <- localpoly.reg(X, Y, bandwidth = "Adp", degree.pol = 1)
R> h <- h.select(X, Y, method = "cv")
R> fit2 <- sm.regression(X, Y, h = h)
```

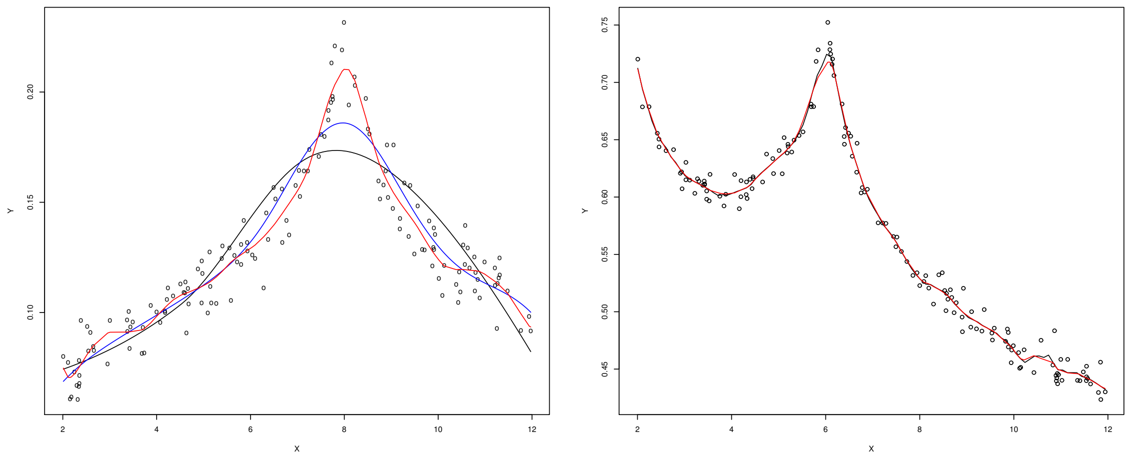

Figure 2: Left: Estimated curves using `localpoly.reg()` with leave-one-out cross validation to select the bandwidth (red), `locfit()` (black) and `locpoly()` (blue). Right: Estimated curves using `localpoly.reg()` with generalized cross validation (red) and `locpol()` (black), both with degree of polynomial equal to 1.

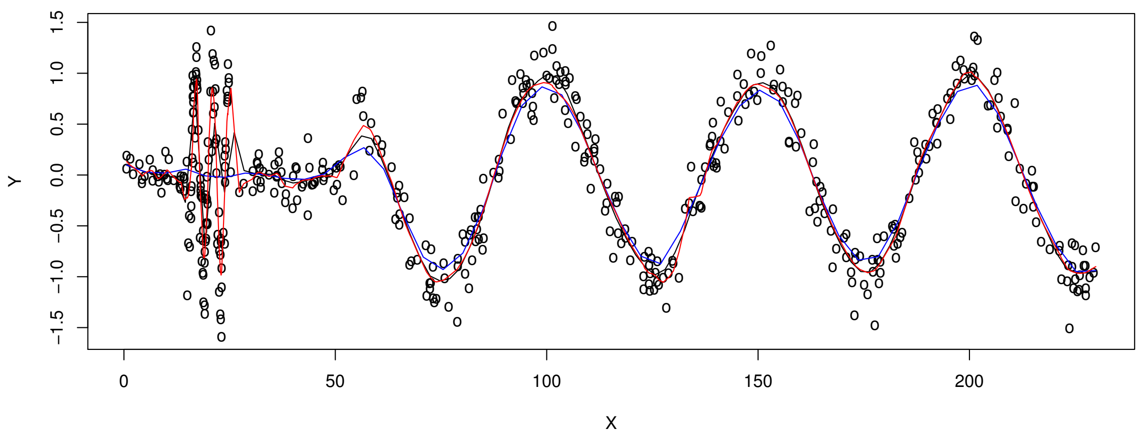

Figure 3: Estimated curves using `localpoly.reg()` with adaptive bandwidth selection (red), `locfit()` using 0.05 for nearest neighbor argument (black) and `sm.regression()` using cross validation to select bandwidth (blue).

```
R> fit3 <- locfit(Y ~ lp(X, deg = 1, nn = .05))
R> plot(fit3, get.data = TRUE)
R> lines(fit2$eval.points, fit2$estimate, col = 4)
R> lines(fit$points, fit$predicted, col = 2)
```

In Figure 2 on the left, the default settings of function `locfit()` were used, and in that case the fit produced was clearly oversmoothed. `locpoly()` may have the same characteristics as `localpoly.reg()`, as long as the user finds another code for selecting a useful bandwidth, for the user is required to input the bandwidth manually. Figure 2 on the right presents the curve estimated by `locpol()` and `localpoly.reg()`, where we can see that the latter has a slightly smoother estimate. In Figure 3, the simulated data has different patterns of relationship according to the predictor x, hence requiring a variable bandwidth. Clearly `sm.regression()` does not capture the first sine trend, and `locfit()` only does so when its nearest neighbor argument is manually set to be very small (0.05), while the adaptive algorithm built into `localpoly.reg()` captures all trends in the data.

An important feature that may be useful for the smoothing choice, is that a fixed bandwidth for each observed data point can be inputted to the function directly. When the observed data X is univariate, a vector of bandwidths of the same length as X may be used, corresponding to a bandwidth for each data point. For the case of multivariate $\mathbf{X}$, a vector or a matrix of bandwidths can be used: If a vector, it must be of the same dimension of the columns of $\mathbf{X}$, so that each entry of the bandwidth vector will be used for each covariate; if a matrix, it must be of the same dimensions of $\mathbf{X}$, where each entry of the matrix corresponds to each data point.

Modern research deals with an increasing amount of data, and often scientists face situations where linear models are not adequate. With the large number of observations available, it is possible to fit nonparametric models for multivariate predictors. Several models can be used in this case such as additive models, varying coefficient models, single index models, etc., with packages in R such as **gam** (Hastie 2016), **mgcv** (Wood 2017, 2006), etc. However, for a completely nonparametric model, the options for such analysis in R are restricted to a few functions, namely `loess()`, `locfit()` and `sm.regression()`. While `sm.regression()` can only deal with up to 2 predictors, `loess()` and `locfit()` work with a set of up to 4 covariates. For function `localpoly.reg()` in package **NonpModelCheck**, the user can input a matrix $\mathbf{X}$ with no limit on the number of predictors (columns) or on observations (rows). Obviously, caution must be taken with the curse of dimensionality, for an estimation in several dimensions for small sample size may be very poor. The call of the function is the same as described previously, so that if the argument $\mathbf{X}$ is a matrix, the function will identify it and run a multivariate local polynomial regression.

In the case of a bivariate situation, the function `plot3d.localpoly.reg()` available in package **NonpModelCheck**, creates a 3-dimensional plot of the surface estimated by function `localpol.reg()` using function `persp()` from the base package **graphics**. The call for this function is

```
R> plot3d.localpoly.reg(X, Y, bandwidth = "CV", gridsize = 30,
+    degree.pol = 0, kernel.type = "epanech", gridsurface = 30,
+    xlab = expression(X_1), ylab = expression(X_2), zlab = expression(Y),
+    theta = 30, phi = 30, expand = 0.5, col = "lightblue", ltheta = 120,
+    shade = 0.75, ticktype = "detailed", pch = 16, ...)
```

The user can input the parameters for function `localpoly.reg()` and also for `persp()`. The only extra parameter is `gridsurface`, which corresponds to the number of points on each axis at which to estimate the local polynomial surface. Figure 4 shows an example of an estimated bivariate curve plot from `plot3d.localpoly.reg()`. The code for such a plot is as follows.

```
R> set.seed(1111)
R> X <- matrix(0, 200, 2)
R> X[, 1] <- runif(200, - 3, 3)
R> X[, 2] <- runif(200, - 3, 3)
R> Y <- 4 * sin(pi * X[, 1]) + 4 * sin(pi * X[, 2]) + rnorm(200, 0, 0.3)
```

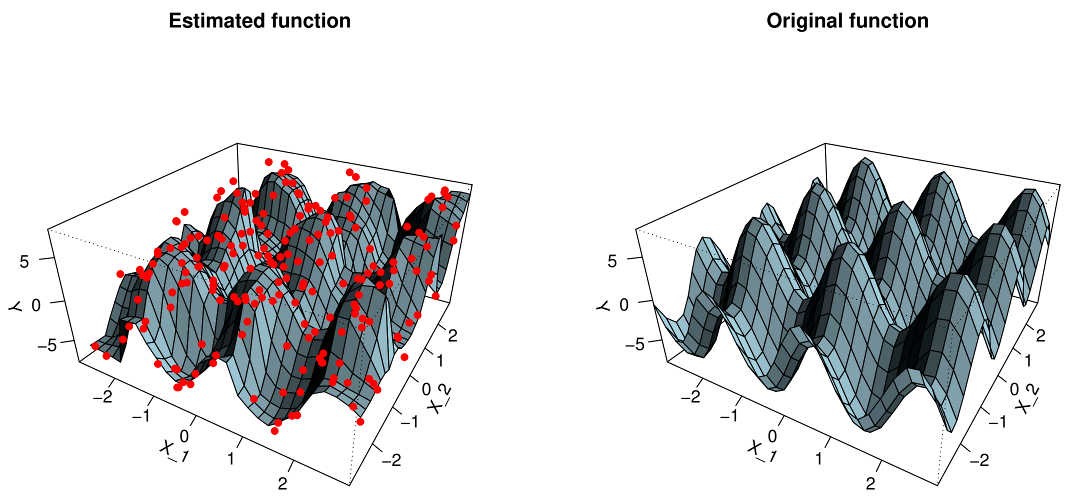

Figure 4: Estimated curve from a bivariate local polynomial regression with `localpoly.reg()` (left) and the original surface (right).

```
R> par(mfrow = c(1, 2))
R> plot3d.localpoly.reg(X, Y, bandwidth = "CV2", degree.pol = 0,
+    gridsurface = 30,  main = "Estimated function")
R> f <- function(x1, x2) { 4 * sin(pi * x1) + 4 * sin(pi * x2) }
R> persp(seq(min(X[, 1]), max(X[, 1]), length = 30), seq(min(X[, 2]),
+    max(X[, 2]), length = 30), outer(seq(min(X[, 1]), max(X[, 1]),
+    length = 30), seq(min(X[, 2]), max(X[, 2]), length = 30), f),
+    theta = 30, phi = 30, expand = 0.5, col = "lightblue", ltheta = 120,
+    shade = 0.75, ticktype = "detailed",  xlab = "X_1", ylab = "X_2",
+    zlab = "Y", main = "Original function")
```

Note that the choice of bandwidth was `"CV2"`. This indicates a request for leave-one-out cross validation for each dimension, as a search in a tensor product. If `bandwidth` is set to `"GCV2"`, the algorithm runs the tensor product search for generalized cross validation. The values `"CV"` and `"GCV"` used in the univariate case can also be used for multivariate predictors: The bandwidth for each dimension is chosen in an individual search, within the regression of each predictor on the response.

In conclusion, R package **NonpModelCheck** is user friendly and provides a very useful combination of tools for local polynomial estimation, which is not yet found in any other existing package in R. It performs the classical method, with unrestricted degree of polynomial for the univariate case, and up to second degree for the multivariate case. Moreover, three different bandwidth selection methods are available: cross-validation, generalized cross validation and adaptive (binned) bandwidth. Hence, **NonpModelCheck** provides resources, not found in other packages, that combined are useful for the user seeking to do nonparametric local polynomial estimation.

Affiliation:

Adriano Zanin Zambom
Department of Mathematics and Statistics
Loyola University Chicago
E-mail: adriano.zambom@gmail.com

Michael G. Akritas
Department of Statistics
Penn State University
E-mail: mga@stat.psu.edu

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
April 2017, Volume 77, Issue 10 *Submitted:* 2015-03-18
doi:10.18637/jss.v077.i10 *Accepted:* 2016-01-12