Using applications created by Rwui

A demonstration application created by Rwui can be accessed from the 'Help' page of Rwui.

Applications created by Rwui can include a login page. Access can be controlled either by a single password, or by username/password pairs.

If an application created by Rwui includes a Multiple/Replicate File upload page, then the application consists of two web pages on which the user enters information. On the first web page the user uploads multiple files one at a time. Once completed, a button takes the user to a second web page where singleton data files and values for all other variables are entered. The 'Analyse' button on this page submits the values of variables, uploads any data files and runs the R script. If a Multiple/Replicate File upload page is not included, the application consists of this second web page only.

Before running the R script the application first checks the validity of the values that the user has entered and returns an error message to the page if any are invalid. During the analysis, progress information can be displayed for the user. To enable this the R script must append information to a text file at stages during its execution. This text file is displayed for the user in a JavaScript pop-up window which refreshes at fixed intervals.

On completion of the analysis, a link to a Results page appears at the bottom of the web page. The user can change data files and/or the values of any of the variables and re-analyse, and the new results will appear as a second link at the bottom of the page, and so on. Clicking on a link brings up the Results page for the corresponding analysis. Figure 2 shows an example of a results page.

The user can download individual results files by clicking on the name of the appropriate file on a Results page. Alternatively, each Results page also contains a link which will download all the results files from the page and the html of the page itself. In this way the user can view offline saved Results pages with their associated results files. Any uploaded data files are also linked to on the Results page, giving the user the opportunity to check that the correct data has been submitted and has been uploaded correctly.

The applications include a SessionListener which detects when a user's session is about to expire. The SessionListener then removes all the working directories created during the session from the server, to prevent it from becoming clogged with data. By default a session expires 30 minutes after it was last accessed by the user, but the delay can be changed in the server configuration.

All the code of a complete demonstration application created by Rwui can be downloaded from a link on the 'Help' page of Rwui.

Acknowledgment

RN was funded as part of a Wellcome Trust Functional Genomics programme grant.

Bibliography

Gentleman,R.C., Carey,V.J., Bates,D.M., Bolstad,B., Dettling,M., Dudoit,S., Ellis,B., Gautier,L., Ge,Y., Gentry,J., Hornik,K., Hothorn,T., Huber,W., Iacus,S., Irizarry,R., Leisch,F., Li,C., Maechler,M., Rossini,A.J., Sawitzki,G., Smith,C., Smyth,G., Tierney,L., Yang,J.Y.H., Zhang,J. (2004) Bioconductor: Open software development for computational biology and bioinformatics, *Genome Biology*, **5**, R80.

Richard Newton
School of Crystallography
Birkbeck College, University of London, UK
`r.newton@mail.cryst.bbk.ac.uk`
Lorenz Wernisch
MRC Biostatistics Unit, Cambridge, UK
`lorenz.wernisch@mrc-bsu.cam.ac.uk`

The np Package: Kernel Methods for Categorical and Continuous Data

by Tristen Hayfield and Jeffrey S. Racine

Readers of R News are certainly aware of a number of nonparametric kernel[1] methods that exist in R base (e.g., `density`) and in certain R packages (e.g., `locpoly` in the **KernSmooth** package). Such functionality allows R users to nonparametrically model a density or to conduct nonparametric local polynomial regression, to name but two applications of kernel methods. Nonparametric kernel approaches are appealing to applied researchers because they often reveal features in the data that might be missed by classical parametric methods. However, traditional nonparametric kernel methods presume that the underlying data is continuous, which is frequently not the case.

[1] A 'kernel' is simply a weighting function.

Practitioners often encounter a mix of categorical and continuous data types, but may still wish to proceed in a nonparametric direction. The traditional nonparametric approach is called a 'frequency' approach, whereby data is broken up into subsets ('cells') corresponding to the values assumed by the categorical variables, and only then do you apply say `density` or `locpoly` to the continuous data remaining in each cell. Nonparametric frequency approaches are widely acknowledged to be unsatisfactory. Recent theoretical developments offer practitioners a variety of kernel-based methods for categorical (i.e., unordered and ordered factors) data only or for a mix of continuous and categorical data; see Li and Racine (2007) and the references therein for an in-depth treatment of these methods, and also see the articles listed in the bibliography below.

The **np** package implements recently developed kernel methods that seamlessly handle the mix of continuous, unordered, and ordered factor data types often found in applied settings. The package also allows users to create their own routines using high-level function calls rather than writing their own C or Fortran code.[2] The design philosophy underlying **np** is simply to provide an intuitive, flexible, and extensible environment for applied kernel estimation.

Currently, a range of methods can be found in the **np** package including unconditional (Li and Racine, 2003; Ouyang et al., 2006) and conditional (Hall et al., 2004; Racine et al., 2004) density estimation and bandwidth selection, conditional mean and gradient estimation (local constant (Racine and Li, 2004; Hall et al., forthcoming) and local polynomial (Li and Racine, 2004)), conditional quantile and gradient estimation (Li and Racine, forthcoming), model specification tests (regression (Hsiao et al., forthcoming), quantile, significance (Racine et al., forthcoming)), semiparametric regression (partially linear, index models, average derivative estimation, varying/smooth coefficient models), etc.

Before proceeding, we caution the reader that data-driven bandwidth selection methods can be numerically intensive, which is the reason underlying the development of an MPI-aware[3] version of the **np** package that uses some of the functionality of the **Rmpi** package, which we have tentatively called the **npRmpi** package. The functionality of **np** and **npRmpi** will be identical; however, using **npRmpi** you could take advantage of a cluster computing environment or a multi-core/multi-cpu desktop machine thereby alleviating the computational burden associated with the nonparametric analysis of large datasets. We ought also to point out that data-driven (i.e., automatic) bandwidth selection procedures are not guaranteed always to produce good results. For this reason, we advise the reader to interrogate their bandwidth objects with the `summary` command which produces a table of the bandwidths for the continuous variables scaled by an appropriate constant ($\sigma_x n^{\alpha}$ where α depends on the kernel order and number of continuous variables, e.g. $\alpha = -1/5$ for one continuous variable and a second order kernel), which some readers may find helpful. Also, the admissible range for the bandwidths for the categorical variables is provided when `summary` is used, which some readers may also find helpful.

We have tried to make **np** flexible enough to be of use to a wide range of users. All options can be tweaked by users (kernel function, kernel order, bandwidth type, estimator type and so forth). One function, `npksum`, allows you to create your own estimators, tests, etc. The function `npksum` is simply a call to highly optimized C code, so you get the benefits of compiled code along with the power and flexibility of the R language. We hope that incorporating the `npksum` function renders the package suitable for teaching and research alike.

There are two ways in which you can interact with functions in **np**: i) using data frames, or ii) using a formula interface, where appropriate.

To some, it may be natural to use the data frame interface. The R `data.frame` function preserves a variable's type once it has been cast (unlike `cbind`, which we avoid for this reason). If you find this most natural for your project, you first create a data frame casting data according to their type (i.e., one of continuous (default), `factor`, `ordered`), as in

```
data.object <- data.frame(x1=factor(x1),
  x2, x3=ordered(x3))
```

where `x1` is, say, a binary factor, `x2` continuous, and `x3` an ordered factor. Then you could pass this data frame to the appropriate **np** function, say

```
npudensbw(dat=data.object)
```

To others, however, it may be natural to use the formula interface that is used for the regression example outlined below. For nonparametric regression functions such as `npregbw`, you would proceed just as you would using `lm`, e.g.,

```
npregbw(y ~ factor(x1) + x2)
```

except that you would of course not need to specify, e.g., polynomials in variables or interaction terms. Every function in **np** supports both interfaces, where appropriate.

Before proceeding, a few words on the formula interface are in order. We use the standard formula interface as it provided capabilities for handling missing observations and so forth. This interface, however, is simply a convenient device for telling a routine which variable is, say, the outcome

[2] The high-level functions found in the package in turn call compiled C code, allowing users to focus on the application rather than the implementation details.

[3] MPI is a library specification for message-passing, and has emerged as a dominant paradigm for portable parallel programming.

and which are, say, the covariates. That is, just because one writes `x1 + x2` in no way means or is meant to imply that the model will be linear and additive (why use fully nonparametric methods to estimate such models in the first place?). It simply means that there are, say, two covariates in the model, the first being `x1` and the second `x2`, we are passing them to a routine with the formula interface, and nothing more is presumed or implied.

Regardless of whether you use the data frame or formula interface, workflow for nonparametric estimation in **np** typically proceeds as follows:

1. compute appropriate bandwidths;
2. estimate an object and extract fitted or predicted values, standard errors, etc.;
3. optionally, plot the object.

In order to streamline the creation of a set of complicated graphics objects, `plot` (which calls `npplot`) is dynamic; i.e., you can specify, say, bootstrapped error bounds and the appropriate routines will be called in real time.

Efficient nonparametric regression in the presence of qualitative data

We begin with a motivating example that demonstrates the potential benefits arising from the use of kernel smoothing methods that smooth both the qualitative and quantitative variables in a particular manner; see the subsequent section for more detailed information regarding the method itself. For what follows, we consider an application taken from Wooldridge (2003, pg. 226) that involves multiple regression analysis with qualitative information.

We consider modeling an hourly wage equation for which the dependent variable is log(`wage`) (`lwage`) while the explanatory variables include three continuous variables, namely `educ` (years of education), `exper` (the number of years of potential experience), and `tenure` (the number of years with their current employer) along with two qualitative variables, `female` ("Female"/"Male") and `married` ("Married"/"Notmarried"). For this example there are $n = 526$ observations.

The classical parametric approach towards estimating such relationships requires that one first specify the functional form of the underlying relationship. We start by first modelling this relationship using a simple parametric linear model. By way of example, Wooldridge (2003, pg. 222) presents the following model:[4]

```
> data("wage1")
>
> model.ols <- lm(lwage ~ factor(female) +
+  factor(married) + educ + exper + tenure,
+  data=wage1)
>
> summary(model.ols)
....
```

This model is, however, restrictive in a number of ways. First, the analyst must specify the functional form (in this case linear) for the continuous variables (`educ`, `exper`, and `tenure`). Second, the analyst must specify how the qualitative variables (`female` and `married`) enter the model (in this case they affect the model's intercepts only). Third, the analyst must specify the nature of any interactions among all variables, quantitative and qualitative (in this case, there are none). Should any of these assumptions be incorrect, then the estimated model will be biased and inconsistent, potentially leading to faulty inference.

One might next test the null hypothesis that this parametric linear model is correctly specified using the consistent model specification test found in Hsiao et al. (forthcoming) that admits both categorical and continuous data (this example will likely take a few minutes on a desktop computer as it uses bootstrapping and cross-validated bandwidth selection):

```
> data("wage1")
> attach(wage1)
>
> model.ols <- lm(lwage ~ factor(female) +
+  factor(married) + educ + exper + tenure,
+  x=TRUE, y=TRUE)
>
> X <- data.frame(factor(female),
+  factor(married), educ, exper, tenure)
>
> output <- npcmstest(model=model.ols,
+  xdat=X, ydat=lwage, tol=.1, ftol=.1)
> summary(output)

Consistent Model Specification Test
....
IID Bootstrap (399 replications)
Test Statistic 'Jn': 5.594745e+00
P Value: 0.000000e+00
....
```

This naïve linear model is rejected by the data (the P-value for the null of correct specification is < 0.001), hence one might proceed instead to model this relationship using kernel methods.

As noted, the traditional nonparametric approach towards modeling relationships in the presence of qualitative variables requires that you first split your data into subsets containing only the continuous

[4] We would like to thank Jeffrey Wooldridge for allowing us to use his data. Also, we would like to point out that Wooldridge starts out with this naïve linear model, but quickly moves on to a more realistic model involving nonlinearities in the continuous variables and so forth.

variables of interest (lwage, exper, and tenure). For instance, we would have four such subsets, a) $n = 132$ observations for married females, b) $n = 120$ observations for single females, c) $n = 86$ observations for single males, and d) $n = 188$ observations for married males. One would then construct smooth nonparametric regression models for each of these subsets and proceed with the analysis in this fashion. However, this may lead to a loss in efficiency due to a reduction in the sample size leading to overly variable estimates of the underlying relationship.

Instead, however, we could construct smooth nonparametric regression models by i) using a smoothing function that is appropriate for the qualitative variables such as that proposed by Aitchison and Aitken (1976), and ii) modifying the nonparametric regression model as was done by Li and Racine (2004). One can then conduct sound nonparametric estimation based on the $n = 526$ observations rather than resorting to sample splitting. The rationale for this lies in the fundamental concept that doing so may introduce potential bias, but it will always reduce variability, thus leading to potential finite-sample efficiency gains. Our experience has been that the potential benefits arising from this approach more than offset the potential costs in finite-sample settings.

Next, we consider using the local-linear nonparametric method described in Li and Racine (2004). For the reader's convenience we supply precomputed cross-validated bandwidths which are automatically loaded when one reads the wage1 dataset. The commented out code that generates the cross-validated bandwidths is also provided should the reader wish to fully replicate the example (recall being cautioned about the computational burden associated with multivariate data-driven bandwidth methods).

```
> data("wage1")
>
> #bw.all <- npregbw(lwage ~ factor(female) +
> # factor(married) + educ + exper + tenure,
> # regtype="ll", bwmethod="cv.aic",
> # data=wage1)
>
> model.np <- npreg(bws=bw.all)
>
> summary(model.np)

Regression Data: 526 training points,
                 in 5 variable(s)
....
Kernel Regression Estimator: Local-Linear
Bandwidth Type:Fixed
Residual standard error: 1.371530e-01
R-squared: 5.148139e-01
....
```

Note that the bandwidth object is the only thing you need to pass to npreg as it encapsulates the kernel types, regression method, and so forth. You could, of course, use npreg with a formula and perhaps manually specify the bandwidths using a bandwidth vector if you so chose. We have tried to make each function as flexible as possible to meet the needs of a variety of users.

The goodness of fit of the nonparametric model ($R^2 = 51.5\%$) is better than that for the parametric model ($R^2 = 40.4\%$). In order to investigate whether this apparent improvement reflects overfitting or simply that the nonparametric model is in fact more faithful to the underlying data generating process, we shuffled the data and created two independent samples, one of size $n_1 = 400$ and one of size $n_2 = 126$. We fit the models on the n_1 training observations then evaluate the models on the n_2 independent hold-out observations using the predicted square error criterion, namely $n_2^{-1}\sum_{i=1}^{n_2}(y_i - \hat{y}_i)^2$, where the y_is are the lwage values for the hold-out observations and the $\hat{y}_i$s are the predicted values. Finally, we compare the parametric model, the nonparametric model that smooths both the qualitative and quantitative variables, and the traditional frequency nonparametric model that breaks the data into subsets and smooths the quantitative data only.

```
> data("wage1")
> set.seed(123)
>
> # Shuffle the data and create two datasets...
>
> ii <- sample(seq(nrow(wage1)),replace=FALSE)
>
> wage1.train <- wage1[ii[1:400],]
> wage1.eval <- wage1[ii[401:nrow(wage1)],]
>
> # Compute the parametric model for the
> # training data...
>
> model.ols <- lm(lwage ~ factor(female) +
+  factor(married) + educ + exper + tenure,
+  data=wage1.train)
>
> # Obtain the out-of-sample predictions...
>
> fit.ols <- predict(model.ols,
+  data=wage1.train, newdata=wage1.eval)
>
> # Compute the predicted square error...
>
> pse.ols <- mean((wage1.eval$lwage -
+                  fit.ols)^2)
>
> #bw.subset <- npregbw(
> # lwage~factor(female) + factor(married) +
> # educ + exper + tenure, regtype="ll",
> # bwmethod="cv.aic", data=wage1.train)
>
```

```
> model.np <- npreg(bws=bw.subset)
>
> # Obtain the out-of-sample predictions...
>
> fit.np <- predict(model.np,
+  data=wage1.train, newdata=wage1.eval)
>
> # Compute the predicted square error...
>
> pse.np <- mean((wage1.eval$lwage -
+                 fit.np)^2)
>
> # Do the same for the frequency estimator...
> # We do this by setting lambda=0 for the
> # qualitative variables...
>
> bw.freq <- bw.subset
> bw.freq$bw[1] <- 0
> bw.freq$bw[2] <- 0
>
> model.np.freq <- npreg(bws=bw.freq)
>
> # Obtain the out-of-sample predictions...
>
> fit.np.freq <- predict(model.np.freq,
+  data=wage1.train, newdata=wage1.eval)
>
> # Compute the predicted square error...
>
> pse.np.freq <- mean((wage1.eval$lwage -
+                      fit.np.freq)^2)
>
> # Compare performance for the
> # hold-out data...
>
> pse.ols
[1] 0.1469691
> pse.np
[1] 0.1344028
> pse.np.freq
[1] 0.1450111
```

The predicted square error on the hold-out data was 0.147 for the parametric linear model, 0.145 for the traditional nonparametric estimator that splits the data into subsets, and 0.134 for the nonparametric estimator that uses the full sample but smooths both the qualitative and quantitative data. We therefore conclude that the nonparametric model that smooths both the continuous and qualitative variables appears to provide a better description of the underlying data generating process than either the nonparametric model that uses sample splitting or the naïve linear model.

Note that for this example we have only four cells. If one used all qualitative variables included in the dataset (16 in total), one would have 65,536 cells, many of which would be empty, and most having far too few observations to provide meaningful nonparametric estimates. As the number of qualitative variables increases, the difference between the estimator that smooths both continuous and discrete variables in a particular manner and the traditional estimator that relies upon sample splitting will become even more pronounced.

We now briefly outline the essence of kernel smoothing of categorical data for the interested reader.

Categorical data kernel methods

To introduce the R user to the basic concept of kernel smoothing of categorical random variables, consider a kernel estimator of a probability function, i.e. $p(x^d) = P(X^d = x^d)$, where X^d is an unordered (i.e., discrete) categorical variable assuming values in, say, $\{0, 1, 2, \ldots, c-1\}$ where c is the number of possible outcomes of the discrete random variable X^d. Aitchison and Aitken (1976) proposed a kernel estimator of $p(x^d)$ given by

$$\hat{p}(x^d) = \frac{1}{n}\sum_{i=1}^{n} L(X_i^d = x^d),$$

where $L(\cdot)$ is a kernel function defined by, say,

$$L(X_i^d = x^d) = \begin{cases} 1-\lambda & \text{if } X_i^d = x^d \\ \lambda/(c-1) & \text{otherwise,} \end{cases}$$

and where λ is a 'smoothing' parameter that can be selected via a variety of data-driven methods; see also Ouyang et al. (2006) for theoretical underpinnings of least-squares cross-validation in this context.

The smoothing parameter λ is restricted to lie in $[0, (c-1)/c]$ for the kernel given above. Note that when $\lambda = 0$, then $L(X_i^d = x^d)$ becomes an indicator function, and hence $\hat{p}(x^d)$ would be the standard frequency probability estimator (i.e., the sample proportion of $X_i^d = x^d$). If, on the other hand, $\lambda = (c-1)/c$, its upper bound, then $L(X_i^d = x^d)$ becomes a constant function and $\hat{p}(x^d) = 1/c$ for all x^d, which is the discrete uniform distribution. Data-driven methods for selecting λ are based on minimizing expected square error loss, among other criteria.

A number of issues surrounding this estimator are noteworthy: i) this approach extends trivially to a general multivariate setting, and in fact is best suited to such settings; ii) to deal with ordered variables you simply use an 'ordered kernel'; iii) there is no "curse of dimensionality" associated with smoothing discrete probability functions – the estimators are $\sqrt{n}$ consistent, just like their parametric counterparts; and iv) you can "mix-and-match" data types using "generalized product kernels" as we describe in the next section.

Why would anyone want to smooth a probability function in the first place? For finite-sample efficiency reasons of course. That is, smoothing a probability function may introduce some finite-sample

bias, but, it *always* reduces variability. In settings where you are modeling a mix of categorical and continuous data or where one has sparse multivariate categorical data, the efficiency gains can be substantial indeed.

Mixed data kernel methods

Estimating a joint density function defined over mixed data (i.e., both categorical and continuous) follows naturally using generalized product kernels. For example, for one discrete variable x^d and one continuous variable x^c, our kernel estimator of the joint PDF would be

$$\hat{f}(x^d, x^c) = \frac{1}{nh_x} \sum_{i=1}^{n} L(X_i^d = x^d) W\left(\frac{X_i^c - x^c}{h_{x^c}}\right),$$

where $L(X_i^d = x^d)$ is a categorical data kernel function, while $W[(X_i^c - x^c)/h_{x^c}]$ is a continuous data kernel function (e.g., Epanechnikov, Gaussian, or uniform) and h_{x^c} is the bandwidth for the continuous variable; see Li and Racine (2003) for further details.

Once we can consistently estimate a joint density function defined over mixed data, we can then proceed to estimate a range of statistical objects of interest to practitioners. Some mainstays of applied data analysis include estimation of regression functions and their derivatives, conditional density functions and their quantiles, conditional mode functions (i.e., count data models, probability models), and so forth, each of which is currently implemented.

Nonparametric estimation of binary outcome and count data models

For what follows, we adopt the conditional probability estimator proposed in Hall et al. (2004) to estimate a nonparametric model of a binary outcome when there exist a number of categorical covariates.

For this example, we use the `birthwt` data taken from the **MASS** package, and compute a parametric Logit model and a nonparametric conditional mode model. We then compare their confusion matrices and assess their classification ability. The outcome is an indicator of low infant birthweight (0/1).

```
> library("MASS")
> attach(birthwt)
>
> # First, we model this with a simple
> # parametric Logit model...
>
> model.logit <- glm(low ~ factor(smoke) +
+  factor(race) + factor(ht) + factor(ui)+
+  ordered(ftv) + age + lwt, family=binomial)
>
> # Now we model this with the nonparametric
> # conditional density estimator and compute
> # the conditional mode...
>
> bw <- npcdensbw(factor(low) ~
+  factor(smoke) + factor(race) +
+  factor(ht) + factor(ui)+ ordered(ftv) +
+  age + lwt, tol=.1, ftol=.1)
> model.np <- npconmode(bws=bw)
>
> # Finally, we compare confusion matrices
> # from the logit and nonparametric models...
>
> # Parametric...
>
> cm <- table(low,
+  ifelse(fitted(model.logit) > 0.5, 1, 0))
> cm

low   0   1
  0 119  11
  1  34  25
>
> # Nonparametric...
>
> summary(model.np)

Conditional Mode data: 189 training points,
                       in 8 variable(s)
....
Bandwidth Type: Fixed

Confusion Matrix
      Predicted
Actual   0   1
     0 127   3
     1  27  32

....
```

For this example the nonparametric model is better able to predict low birthweight infants than its parametric counterpart, correctly predicting 159/189 birthweights compared with 144/189 for the parametric model.

Visualizing and summarizing nonparametric results

Summarizing nonparametric results and comparing them with parametric ones is perhaps one of the main challenges faced by the practitioner. We have attempted to provide a range of summary measures and plotting methods to facilitate these tasks.

The `plot` function (which calls `npplot`) can be used on most nonparametric objects. In order to further facilitate comparison of parametric with nonparametric results, we also compute a number of summary measures of goodness of fit, normalized bandwidths ('scale factors') and so forth that are ac-

cessed by the summary command, while objects generated by **np** also contain various alternative measures of goodness of fit and the like. We again consider the approach detailed in Li and Racine (2004) which conducts local linear kernel regression with a mix of discrete and continuous data types on a popular macroeconomic dataset. The original study assessed the impact of Organisation for Economic Co-operation and Development (OECD) membership on a country's growth rate when controlling for a range of factors such as population growth rates, stock of capital (physical and human) and so forth. For this example we shall use the formula interface rather than the data frame interface. For this example, we have already computed the cross-validated bandwidths which are loaded when one reads the oecdpanel dataset.

```
> data("oecdpanel")
> attach(oecdpanel)
>
> #bw <- npregbw(growth ~ factor(oecd) +
> # ordered(year) + initgdp + popgro+ inv +
> # humancap, bwmethod="cv.aic",
> # regtype="ll")
>
> summary(bw)

Regression Data (616 observations,
                  6 variable(s)):

Regression Type: Local-Linear

....

>
> model <- npreg(bw)
>
> summary(model)

Regression Data: 616 training points,
                 in 6 variable(s)
....
>
> plot(bw,
+      plot.errors.method="bootstrap",
+      plot.errors.boot.num=25)
>
```

We display partial regression plots in Figure 1.[5] We also plot bootstrapped variability bounds, where the bootstrapping is done via the **boot** package thereby facilitating a variety of bootstrap methods.

If you prefer gradients rather than levels, you can simply use the argument gradients=TRUE in plot to indicate that you wish to plot partial derivatives rather than the conditional mean function itself.[6]

Many of the accessor functions such as predict, fitted, and residuals work with most nonparametric objects, where appropriate. As well, gradients is a generic function which extracts gradients from objects. The R function coef can be used for extracting the parametric coefficients from semiparametric models such as those created with npplreg.

Creating mixed data kernel objects via npksum

Rather than being limited to only those kernel methods that exist in **np**, you could instead create your own mixed data kernel objects. npksum is a function that computes kernel sums on evaluation data, given a set of training data, data to be weighted (optional), and a bandwidth specification (any bandwidth object). npksum uses highly-optimized C code that strives to minimize its memory footprint, while there is low overhead involved when using repeated calls to this function. The convolution kernel option would allow you to create, say, the least squares cross-validation function for kernel density estimation. You can choose powers to which the kernels constituting the sum are raised (default=1), whether or not to divide by bandwidths, whether or not to generate leave-one-out sums, and so on. Three classes of kernel methods for the continuous data types are available: fixed, adaptive nearest-neighbor, and generalized nearest-neighbor. Adaptive nearest-neighbor bandwidths change with each sample realization in the set, x_i, when estimating the kernel sum at the point x. Generalized nearest-neighbor bandwidths change with the point at which the sum is computed, x. Fixed bandwidths are constant over the support of x. A variety of kernels may be specified by users. Kernels implemented for continuous data types include the second, fourth, sixth, and eighth order Gaussian and Epanechnikov kernels, and the uniform kernel. We have tried to anticipate the needs of a variety of users when designing this function. A number of semiparametric estimators and nonparametric tests in the **np** package in fact make use of npksum.

In the following example, we conduct local-constant (i.e., Nadaraya-Watson) kernel regression. We shall use cross-validated bandwidths drawn from npregbw for this example. Note that we extract the kernel sum from npksum via the '$ksum' return value in both the numerator and denominator of the resulting object. For this example, we use the data frame interface rather than the formula interface, and output from this example is plotted in Figure 2.

[5] A "partial regression plot" is simply a 2D plot of the outcome y versus one covariate x_j when all other covariates are held constant at their respective medians (this can be changed by users).

[6] plot(bw, gradients=TRUE, plot.errors.method="bootstrap", plot.errors.boot.num=25, common.scale=FALSE) will work for this example.

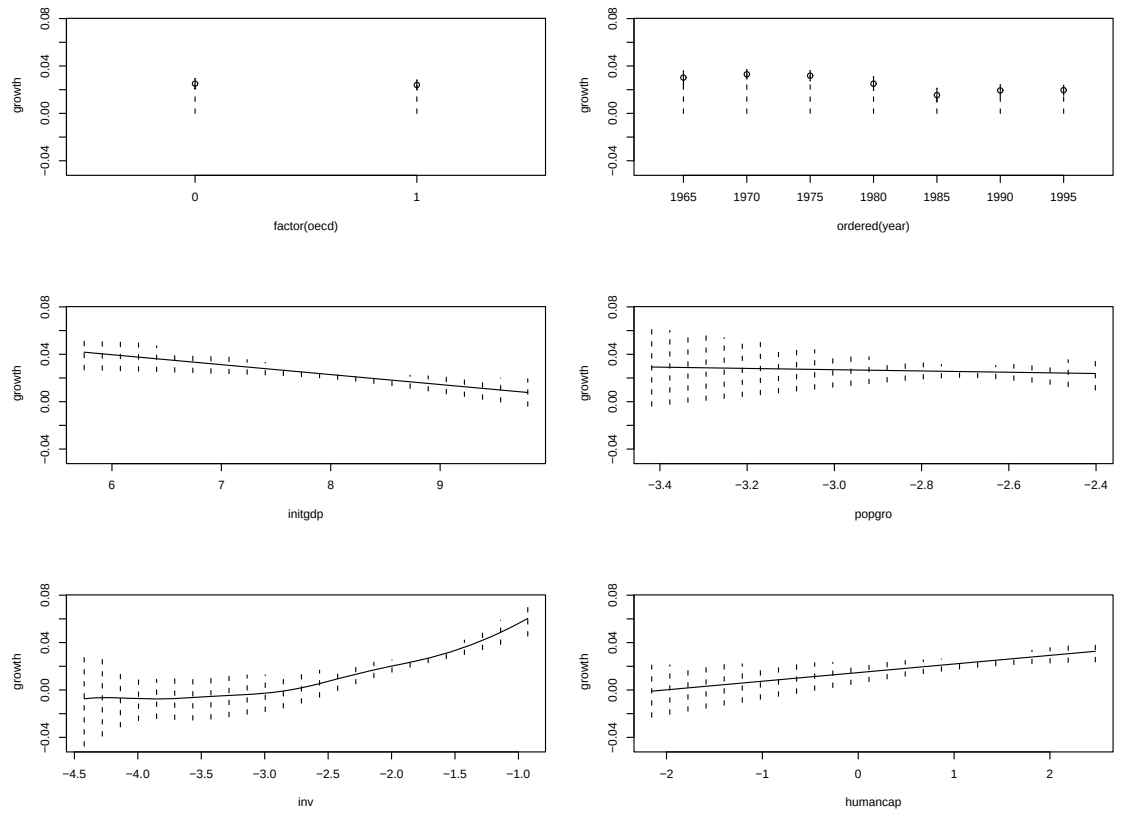

Figure 1: Partial regression plots with bootstrapped error bounds based on a cross-validated local linear kernel estimator.

```
> data("cps71")
> attach(cps71)
....
> fit.lc <- npksum(txdat=age,tydat=logwage,
+                  bws=1.892169)$ksum/
+         npksum(txdat=age,bws=1.892169)$ksum
>
```

Note that the arguments 'txdat' and 'tydat' refer to 'training' data for the regressors X and for the dependent variable y, respectively. One can also specify 'evaluation' data if you wish to evaluate the function on a set of points that differ from the training data. In such cases, you would also specify 'exdat' (and 'eydat' if so desired).

We direct the interested reader to see, by way of illustration, the example available via ?npksum that conducts leave-one-out cross-validation for a local constant regression estimator via calls to the R function nlm, and compares this to the npregbw function.

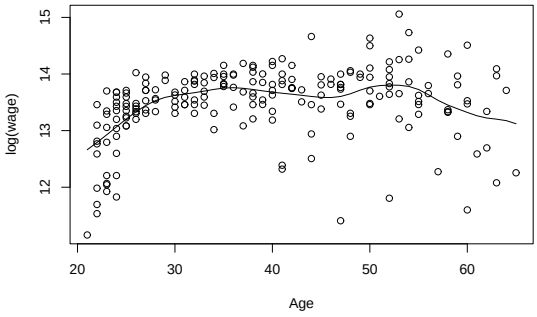

Figure 2: A local constant kernel estimator created with npksum using the Gaussian kernel (default) and a bandwidth of 1.89.

Acknowledgments

We would like to thank but not implicate Achim Zeileis and John Fox, whose many and varied suggestions led to a much improved version of the package and of this review.

Bibliography

J. Aitchison and C. G. G. Aitken. Multivariate binary discrimination by the kernel method. *Biometrika*, 63(3):413–420, 1976.

P. Hall, J. S. Racine, and Q. Li. Cross-validation and the estimation of conditional probability densities. *Journal of the American Statistical Association*, 99(468):1015–1026, 2004.

P. Hall, Q. Li, and J. S. Racine. Nonparametric estimation of regression functions in the presence of irrelevant regressors. *The Review of Economics and Statistics*, forthcoming.

C. Hsiao, Q. Li, and J. S. Racine. A consistent model specification test with mixed categorical and continuous data. *Journal of Econometrics*, 140(2):802–826, 2007.

Q. Li and J. Racine. *Nonparametric Econometrics: Theory and Practice*. Princeton University Press, 2007.

Q. Li and J. S. Racine. Nonparametric estimation of distributions with categorical and continuous data. *Journal of Multivariate Analysis*, 86:266–292, August 2003.

Q. Li and J. S. Racine. Cross-validated local linear nonparametric regression. *Statistica Sinica*, 14(2): 485–512, April 2004.

Q. Li and J. S. Racine. Nonparametric estimation of conditional CDF and quantile functions with mixed categorical and continuous data. *Journal of Business and Economic Statistics*, forthcoming.

D. Ouyang, Q. Li, and J. S. Racine. Cross-validation and the estimation of probability distributions with categorical data. *Journal of Nonparametric Statistics*, 18(1):69–100, 2006.

J. S. Racine and Q. Li. Nonparametric estimation of regression functions with both categorical and continuous data. *Journal of Econometrics*, 119(1):99–130, 2004.

J. S. Racine, Q. Li, and X. Zhu. Kernel estimation of multivariate conditional distributions. *Annals of Economics and Finance*, 5(2):211–235, 2004.

J. S. Racine, J. D. Hart, and Q. Li. Testing the significance of categorical predictor variables in nonparametric regression models. *Econometric Reviews*, 25:523–544, 2007.

J. M. Wooldridge. *Introductory Econometrics*. Thompson South-Western, 2003.

Tristen Hayfield and Jeffrey S. Racine
McMaster University
Hamilton, Ontario, Canada
hayfietj@mcmaster.ca
racinej@mcmaster.ca

eiPack: R × C Ecological Inference and Higher-Dimension Data Management

by Olivia Lau, Ryan T. Moore, and Michael Kellermann

Introduction

Ecological inference (EI) models allow researchers to infer individual-level behavior from aggregate data when individual-level data is unavailable. Table 1 shows a typical unit of ecological analysis: a contingency table with observed row and column marginals and unobserved interior cells.

	col_1	col_2	...	col_C	
row_1	N_{11i}	N_{12i}	...	N_{1Ci}	$N_{1\cdot i}$
row_2	N_{21i}	N_{22i}	...	N_{2Ci}	$N_{2\cdot i}$
...	...	...		...	...
row_R	N_{R1i}	N_{R2i}	...	N_{RCi}	$N_{R\cdot i}$
	$N_{\cdot 1i}$	$N_{\cdot 2i}$	...	$N_{\cdot Ci}$	N_i

Table 1: A typical $R \times C$ unit in ecological inference; red quantities are typically unobserved.

In ecological inference, challenges arise because information is lost when aggregating across individuals, a problem that cannot be solved by collecting more aggregate-level data. Thus, EI models are unusually sensitive to modeling assumptions. Testing these assumptions is difficult without access to individual-level data, and recent years have witnessed a lively discussion of the relative merits of various models (Wakefield, 2004).

Nevertheless, there are many applied problems in which ecological inferences are necessary, either because individual-level data is unavailable or because the aggregate-level data is considered more authoritative. The latter is true in the voting rights context in the United States, where federal courts often base decisions on evidence derived from one or more EI models (Cho and Yoon, 2001). While packages such as **MCMCpack** (Martin and Quinn, 2006) and **eco** (Imai and Lu, 2005), provide tools for 2×2 inference, this is insufficient in many applications. In **eiPack**,