PACKAGE `np` FAQ

JEFFREY S. RACINE

This set of frequently asked questions is intended to help users who are encountering unexpected or undesired behavior when trying to use the `np` package.

Many of the underlying C routines have been extensively tested over the past decade. However, the `R` 'hooks' needed to call these routines along with processing of the data required for a seamless user experience may produce unexpected or undesired results in some settings.

Kindly report any such issues to me, and please include your code and data so that I can help track down any such issues (racinej@mcmaster.ca). And, of course, if you encounter an issue that you think might be of interest to others, kindly email me the relevant information and I will incorporate it into this FAQ.

1. Version Covered

This FAQ refers to the most recent version, which as of this writing is 0.20-2. Kindly update your version should you not be using the most current (from within R, `update.packages()` ought to do it, though also see 3 below.). See the appendix in this file for changes between versions 0.20-2, 0.20-1, 0.20-0, 0.14-3, 0.14-2, 0.14-1, 0.13-1, and 0.12-1.

2. Frequently Asked Questions

(1) *How do I cite the* `np` *package?*

If you load the `np` package and type citation("np") you will be presented with the following information.

```
> citation("np")

To cite np in publications use:

  Tristen Hayfield and Jeffrey S. Racine (2008). Nonparametric
  Econometrics: The np Package. Journal of Statistical Software
  27(5). URL http://www.jstatsoft.org/v27/i05/.
```

Date: November 1, 2008.

```
A BibTeX entry for LaTeX users is

  @Article{,
    title = {Nonparametric Econometrics: The np Package},
    author = {Tristen Hayfield and Jeffrey S. Racine},
    journal = {Journal of Statistical Software},
    year = {2008},
    volume = {27},
    number = {5},
    url = {http://www.jstatsoft.org/v27/i05/},
  }
```

(2) *I have never used* `R`*. Can you direct me to some introductory material that will guide me through the basics...*

There are many excellent introductions to the `R` environment with more on the way. First, I would recommend going directly to the `R` website (`http://www.r-project.org`) and looking under Documentation/Manuals (`http://cran.r-project.org/manuals.html`) where you will discover a wealth of documentation for `R` users of all levels. See also the `R` taskviews summary page (`http://cran.nedmirror.nl/web/views/index.html`) for information grouped under field of interest. A few documents that I mention to my students that are tailored to econometricians include Cribari-Neto & Zarkos (1999) [1], Racine & Hyndman (2002) [5] and Farnsworth (2006) [2], to name but a few.

Often the best resource is right down the hall. Ask a colleague whether they use or know anyone who uses R, then offer to buy that person a coffee and along the way drop something like "I keep hearing about the `R` project...I feel like such a Luddite..."

(3) *How do I keep all* `R` *packages on my system current?*

Run the command `update.packages(checkBuilt=TRUE,ask=FALSE)`, which will not only update all packages that are no longer current, but will also update all packages built under outdated installed versions of R, if appropriate.

(4) *Is there a 'gentle guide' to the* `np` *package that contains some easy to follow examples?*

Perhaps the most gentle introduction is contained in the `np` package itself in the form of a 'vignette'. To view the vignette run R, install the `np` package (`install.packages("np")`), then type `vignette("np",package="np")` to view or print the vignette.

(5) *What is the difference between the* `np` *package and the previous stand-alone programs N ©, NPREG ©, and NPDEN ©?*

The `np` package is built from the same C library that underlies its predecessors. In fact, `R` calls the compiled C code that underlies its predecessors (one of the beauties of `R` is that you can get the benefits of compiled code yet have access to the rich superset of `R` routines and `R` packages built by others). Therefore, there is no penalty in run-time unless the compiler switches differ from those used to build its predecessors.

(6) *How can I read my Stata/SAS/Minitab... data into the R program?*

Install the foreign library via `install.packages("foreign")` then do something like

```
mydat <- read.dta("datafile.dta"),
```

where `datafile.dta` is a Stata data file.

(7) *I want to use so-and-so's semiparametric/nonparametric method, however, the* `np` *package does not include this particular method...*

This is why we have included the function `npksum()`, which exists so that you can create your own kernel objects and take advantage of the general set of methods implemented in the np package without having to write, say, C or Fortran code.

With the options available, you could create new nonparametric tests or even new kernel estimators. For instance, the convolution kernel option would allow you to replicate, say, the least squares cross-validation function for kernel density estimation found in `npudensbw()`. The function `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.

See, by way of illustration, the example in the `npksum()` help file 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.

If you wish to have a method incorporated into a future version of the `np` package, the best way to achieve this is to successfully code up the method using `npksum()`, briefly document it and write up an example, then send it to us. We will then, at our discretion, do our best to adapt and incorporate this into a future version and of course give credit where credit is due.

(8) *Cross-validation takes forever, and I can't wait that long...*

This is the most common complaint for frustrated users coming to terms with numerically demanding statistical methods. I am fond of saying 'if you want the wrong answer, I can give it to you right away', but this wears thin quickly.

(a) Some background may be in order. Cross-validation methods have run times that are exponential in the number of observations. The solution I favor is to run the code in a parallel computing environment. The underlying C code for `np` is MPI-aware (MPI=message passing interface, a popular parallel programming library), and we intend to develop the `R` `np` package in a parallel environment via the `Rmpi` wrapper.

(b) Alternatively, you can use the method outlined in Racine, J.S. (1993) "An Efficient Cross-Validation Algorithm For Window Width Selection for Nonparametric Kernel Regression," *Communications in Statistics*, October, Volume 22, Issue 4, pp 1107-1114. The method is based on the fact that the unknown constant c_j (the 'scale factor') in the formula $c_j\sigma_j n^{-1/(2p+r)}$ is independent of the sample size, so one can conduct bandwidth selection on random subsets and do this for a large number of subsets then take the mean/median over these subsets and feed the scale factor into the final routine for the entire sample. Here is some simple `R` code that replicates the method.

```
## Generate a large dataset

library(np)
set.seed(12345)
n <- 10000
x1 <- runif(n)
x2 <- runif(n)

y <- 1 + x1 + sin(pi*x2) + rnorm(n,sd=.1)

## Set the number of resamples and the subsample size

num.res <- 99
n.sub <- 100

## Create a storage matrix
```

```
bw.mat <- matrix(NA,nrow=num.res,ncol=2)

## Get the scale factors for resamples from the full sample of size n.sub

for(i in 1:num.res)  bw.mat[i,] <- npregbw(y~x1+x2,regtype="ll",
                       subset=sample(n,n.sub,replace=TRUE))$sfactor$x

## A function to compute the median of the columns of a matrix

colMedians <- function(data) {
  colmed <- numeric(ncol(data))
  for(i in 1:ncol(data)) {
    colmed[i] <- median(data[,i])
  }
  return(colmed)
}

## Take the median scale factors

bw <- colMedians(bw.mat)

## The final model for the full dataset

model.res <- npreg(y~x1+x2,bws=bw,regtype="ll",bwscaling=TRUE)
```

(c) Barring this, you can set the search tolerances to be a bit less terse (at the expense of potential accuracy, i.e., becoming trapped in local minima) by setting, say, `tol=0.1` and `ftol=0.1` in the respective bandwidth routine (see the docs for examples). Also, you can set `nmulti=1` which overrides the number of times the search procedure restarts from different random starting values (the default is to restart k times where k is the number of variables). Be warned, however, that this is *definitely not recommended* and should *be avoided at all costs* for all but the most casual examination of a relationship. One ought to use multistarting for any final results and never override default search tolerances *unless increasing multistarts beyond the default.* Results based upon exhaustive search *often differ dramatically* from that based on overriding search tolerances.

(d) For those who like to tinker and who work on a *NIX system with the gcc compiler suite, you can change the default compiler switches used for building `R` packages which may generate some improvements in run time. The default compiler switches are

```
-g -O2
```

and are set in the R-*.*.*/etc/Makeconf file (where *.*.* refers to your `R` version number, e.g. R-2.5.1). You can edit this file and change these defaults to

```
-O3 -ffast-math -fexpensive-optimizations -fomit-frame-pointer
```

then reinstall the `np` package and you may experience some improvements in run time. Note that the -g flag turns on production of debugging information which can carry involve some overhead, so we are disabling this feature. This is not a feature used by the typical applied researcher but if you envision requiring this it is clearly trivial to re-enable debugging. I typically experience in the neighborhood of a 0-5% reduction in run time for data-driven bandwidth selection on a variety of systems depending on the method being used, though mileage will of course vary.

(9) *I wrote a program using* `np` *and it does not work as I expected. . .*

There exist a rather extensive set of examples contained in the docs. You can run these examples by typing `example(npfunctionname)` where npfunctionname is, say, `w`, as in `example(w)`. These examples all pass quality control and produce the expected results, so first see whether your problem fits into an existing example, and if not, carefully follow the examples listed in a given function for syntax issues etc.

If you are convinced that the problem lies with `np` (there certainly will be undiscovered 'features', i.e., bugs), the kindly send me your code and data so that I can replicate and help resolve the issue.

(10) *Under Mac OS X, when I run a command no progress is displayed. . .*

This is a limitation of console input/output (I/O) under Mac OS X. However, if you run `R` in a terminal (install X11 support and export the display) rather than `Rgui` you will get the full *NIX[1] experience.

(11) *When some routines are running under MS Windows,* `R` *appears to be 'not responding.' It appears that the program is not 'hung', rather is simply computing. The previous stand-alone program (N ©) always displayed useful information. . .*

This is a 'feature' of R, and is not specific to the `np` package.

[1]*NIX is often used to describe UNIX and other UNIX-like platforms (i.e., UNIX, BSD, and GNU/Linux distributions). I harbor strong preferences for *NIX computing platforms.

From the `R` Windows FAQ...

"When using Rgui the output to the console seems to be delayed. This is deliberate: the console output is buffered and re-written in chunks to be faster and less distracting. You can turn buffering off or on from the 'Misc' menu or the right-click menu: <Ctrl-W> toggles the setting."

(12) *Some results take a while, and my MS Windows computer is sluggish while* `R` *is running...*

You can easily change the priority of your `R` job on the fly, just as you might under *NIX. Pull up the task manager (<Ctrl>-<Alt>-<Del>), go to the process list, and find the process `Rgui.exe` (or `R.exe` if you are running `Rterm`), select this process by left clicking on it, then right clicking will bring up a menu, select `Set priority`, then change priority to `low` and hit <`ok`>. For lengthy jobs this will make your life much better, and you can, say, run multiple jobs in low priority with no sluggishness whatsoever for your other applications (useful for queuing a number of long jobs).

(13) *A variable must be cast as, say, a factor in order for* `np` *to recognize this as an unordered factor. How can I determine whether my data is already a factor?*

Use the `class()` function. For example, define `x <- factor(c("male","female"))`, then type `class(x)`.

(14) *When I use* `plot()` *or* `npplot()` *existing graphics windows are overwritten. How can I display multiple graphics plots in separate graphics windows?*

Use the `dev.new()` command. This will leave the existing graphics window open and start a new one. The command `dev.list()` will list all graphics windows, and the command `dev.set(integer.foo)` will allow you to switch from one to another and overwrite existing graphics windows should you so choose.

(15) *Sometimes* `plot()` *fails to use my variable names...*

This should not occur unless you are using the data frame method and not naming your variables (e.g., you are doing something like data.frame(ordered(year))). To correct this, name your variables in the respective data frame, as in

```
data <- data.frame(year=ordered(year),gdp)
```

so that the ordered factor appears as 'year' and not 'ordered.year'

(16) *Sometimes* `plot()` *appends my variable names with .ordered or .factor...*

See also 15 above.

(17) *I specify a variable as* `factor()` *or* `ordered()` *in a data frame, then call this when I conduct bandwidth selection. However, when I try to plot the resulting object, it complains*

`Error in eval(expr, envir, enclos) : object "variable" not found...`

This arises because `plot()` (`npplot()`) tries to retrieve the variable from the environment but you have changed the definition when you called the bandwidth selection routine (e.g., `npregbw(y~x,data=dataframe)`).

To correct this, simply call `plot()` with the argument `data=dataframe` where `dataframe` is the name of your dataframe.

(18) *My* `np` *code gives errors when I attempt to run it...*

First, it is good practice to name all arguments (see the docs for examples) as in `npregbw(formula=y~x)` (i.e., explicitly call formula). This will help the code return a potentially helpful error message.

Next, follow the examples listed at the end of each function man page closely. See also 9 above.

(19) *I have 0/1 dummy variables in my parametric model. Can I just pass them to your* `np` *functions as is?*

In general, definitely not. Suppose you have created dummy variables for year, for example, dummy06 which equals 1 for 2006, 0 otherwise, dummy07 which equals 1 for 2007, 0 otherwise etc. We create these for the parametric model. But, the underlying variable is simply year, which equals 2006, 2007, and so forth.

In `np`, you get to economize by just telling the function that the variable 'year' is ordered, as in ordered(year), where year is a vector containing elements 2006, 2007 etc. Of course, seasoned R users would appreciate that this is in fact the simple way to do it with a parametric model as well, but that is another story.

You would *never*, therefore, just pass dummy variables to an `np` function as you would for linear models. The *only* exception is where you have only one 0/1 dummy for one variable, say 'sex', and in this case you still would have to enter this as factor(sex) so that the `np` function recognizes this as a factor.

(20) *Can I skip creating a bandwidth object and enter a bandwidth directly?*

Certainly. For example, attach a dataset via

```
data(cps71)
attach(cps71)
```

then enter, say,

```
plot(age,logwage)
lines(age,fitted(npreg(logwage~age,bws=1)),col="blue")
lines(age,fitted(npreg(logwage~age,bws=2)),col="red")
lines(age,fitted(npreg(logwage~age,bws=3)),col="green")
```

to plot the local constant estimator with a bandwidth of 2 years. Note that the age variable is already sorted. If your data is not sorted you will need to do so prior to plotting so that your lines command works properly. Or see 21 below for a multivariate example.

(21) *When I estimate my gradients when there are two or more variables and extract them with the gradients() function, they are not 'smooth', though if I plot a model with the gradients=TRUE option, they are. The gradients() function must be broken...*

The function `plot()` (`npplot()`) plots 'partial' means and gradients. In other words, it plots x_1 versus $\hat{g}(x_1, \bar{x}_2)$ for the partial mean, where $\bar{x}_2$ is, say, the median/modal value of x_2. It also plots x_1 versus $\partial\hat{g}(x_1, \bar{x}_2)/\partial x_2$ for the gradient. Note that we are controlling for the values of the other covariate(s). This is in effect what people expect when they play with linear parametric models of the form $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \epsilon$ since, given the additive nature of the model, $\partial y/\partial x_1 = \beta_1$ (i.e., does not vary with x_2).

The example below shows how you could manually generate the partial gradients (and means) for your data where the sample realizations form the evaluation data for x_1 (unlike `npplot()` which uses an evenly spaced grid). Note we use the function `uocquantile()` to generate a vector that holds x_2 constant at is median/modal value (i.e., the 0.5 quantile) in the evaluation data. The function uocquantile() can compute quantiles for ordered, unordered, and continuous data (see `?uocquantile` for details).

```
library(np)

n <- 100

x1 <- runif(n)
x2 <- runif(n)
y <- x1^2+x2^2 + rnorm(n,sd=.1)

train <- data.frame(x1,x2,y)

bw <- npregbw(y~x1+x2,
              data=train,
              regtype="ll",
              bwmethod="cv.aic")


eval <-  data.frame(x1 = x1,
                    x2 = rep(uocquantile(x2,.5),n))
```

```
model <- npreg(bws=bw,
               data=train,
               newdata=eval,
               gradients=TRUE)

plot(eval[,1],model$grad[,1],xlab="X1",ylab="Gradient")
```

(22) *I use plot to plot, say, a density and the resulting plot looks like an inverted density rather than a density...*

This can occur when the data-driven bandwidth is dramatically undersmoothed. By default, `npplot()` takes the two extremes of the data (minimum, maximum i.e., actual data points) then creates an equally spaced grid of evaluation data (i.e., not actual data points in general) and computes the density for these points. Since the bandwidth is extremely small, the density estimate at these evaluation points is correctly zero, while those for the sample realizations (in this case only two, the min and max) are non-zero, hence we get two peaks at the edges of the plot and a flat bowl equal to zero everywhere else.

This can happen when your data is heavily discretized and you treat it as continuous. In such cases, treating the data as ordered may result in more sensible estimates.

(23) *Can* `npksum()` *compute analytical derivatives with respect to a continuous variable?*

As of version 0.20-0 and up, yes it can, using the `operator =` argument, which is put to its paces in the following code snippet.

```
Z <- seq(-2.23,2.23,length=100)
par(mfrow=c(2,2))

plot(Z,main="Kernel",ylab="K()",npksum(txdat=0,exdat=Z,bws=1,
         ckertype="epanechnikov",ckerorder=2,operator="normal")$ksum,
         col="blue",type="l")

plot(Z,main="Kernel Derivative",ylab="K()",npksum(txdat=0,exdat=Z,bws=1,
         ckertype="epanechnikov",ckerorder=2,operator="derivative")$ksum,
         col="blue",type="l")

plot(Z,main="Kernel Integral",ylab="K()",npksum(txdat=0,exdat=Z,bws=1,
         ckertype="epanechnikov",ckerorder=2,operator="integral")$ksum,
```

```
         col="blue",type="l")

plot(Z,main="Kernel Convolution",ylab="K()",npksum(txdat=0,exdat=Z,bws=1,
         ckertype="epanechnikov",ckerorder=2,operator="convolution")$ksum,
         col="blue",type="l")
```

An alternative to computing analytical derivatives is to compute them numerically using a tried and true method, namely finite-differences. One simply computes the kernel sum evaluating the sum with variable j set at $x_j - h_j/2$ and calls this, say, $ksum_{j1}$, then again set at $x_j + h_j/2$ and call this $ksum_j2$, then compute $\nabla = (ksum_{j2} - ksum_{j1})/h_j$. This method has been used for both theoretical and applied work and produces consistent estimates of the derivatives, as of course do the analytical derivatives. The following example provides a simple demonstration, and it is clear that the two methods are in agreement. See 21 above for multivariate partial regression when using this method.

```
## In this example we consider the local constant estimator computed
## using npksum, and then use npksum to compute numerical derivatives
## using finite-difference methods, then finally compare them with the
## analytical ones.

library(np)
data(cps71)
attach(cps71)

## Grab the cross-validated bandwidth

bw <- npregbw(logwage~age)
h <- bw$bw[1]

## Evaluate the local constant regression at x-h/2, x+h/2...

ksum.1 <- npksum(txdat=age, exdat=age-h/2,tydat=logwage,bws=bw)$ksum/
  npksum(txdat=age,exdat=age-h/2,bws=bw)$ksum

ksum.2 <- npksum(txdat=age, exdat=age+h/2,tydat=logwage,bws=bw)$ksum/
  npksum(txdat=age,exdat=age+h/2,bws=bw)$ksum

## Compute the numerical gradient...

grad.numerical <- (ksum.2-ksum.1)/h
```

```
## Compare with the analytical gradient...

grad.analytical <- gradients(npreg(bws=bw,gradient=TRUE))

## Plot the resulting estimates...

plot(age,grad.numerical,type="l",col="blue",lty=1,ylab="gradient")
lines(age,grad.analytical,type="l",col="red",lty=2)
```

(24) *Can I use the* `npcmstest()` *function that implements the consistent test for correct specification of parametric regression models as described in Hsiao, Li, & Racine (2007)* [3] *to test for correct specification of the semiparametric partially linear model?*

As Brennan Thompson points out, yes, you can.

To test a parametric linear specification against a semiparametric partially linear alternative, i.e.,

$$H_0 : y = X'\beta + Z'\gamma + u$$
$$H_1 : y = X'\beta + g(Z) + u,$$

you could use `npcmstest()` as follows:

```
lmodel <- lm(y~X+Z,y=TRUE,x=TRUE)
uhat <- resid(lmodel)
npcmstest(xdat=Z,ydat=uhat,model=lmodel)
```

A slightly better way (as discussed in Li & Wang (1998) [4]) would be to use a 'mixed' residual, i.e., $\hat{u}_i = y_i - X_i'\tilde{\beta} - Z_i'\hat{\gamma}$ in the test, where $\tilde{\beta}$ is the semiparametric estimator of β (based on the semiparametric partially linear model), and $\hat{\gamma}$ is the OLS estimator of γ based on the linear model. This could lead to potential power gains due to the improved efficiency of $\hat{\beta}$ under the alternative.

(25) *I want to plot the kernel function itself. How can I do this?*

Use the `npksum()` function and switch the evaluation and training roles as in the following example that plots the 2nd, 4th, 6th and 8th order Epanechnikov kernels.

```
Z <- seq(-sqrt(5),sqrt(5),length=100)
par(mfrow=c(2,2))
plot(Z,ylab="kernel",npksum(txdat=0,exdat=Z,bws=1,ckertype="epanechnikov",
ckerorder=2)$ksum,type="l")
plot(Z,ylab="kernel",npksum(txdat=0,exdat=Z,bws=1,ckertype="epanechnikov",
ckerorder=4)$ksum,type="l")
```

```
plot(Z,ylab="kernel",npksum(txdat=0,exdat=Z,bws=1,ckertype="epanechnikov",
ckerorder=6)$ksum,type="l")
plot(Z,ylab="kernel",npksum(txdat=0,exdat=Z,bws=1,ckertype="epanechnikov",
ckerorder=8)$ksum,type="l")
```

(26) *In version 0.20-0 and up I can 'combine' steps such as bandwidth selection and estimation. But when I do* `summary(modelname)` *I don't get the same summary that I would get from, say,* `summary(bw)` *and then* `summary(modelname)`*. How do I get scale factor summaries when combining steps?*

Don't worry, the bandwidth object exists when you do the combined steps and is easily accessed via `summary(modelname$bws)` or extracted via `bw <- modelname$bws`.

(27) *I estimated a semiparametric index model via* `model <- npindex(y x1+x2)` *but* `se(model)` *returns NULL.*

Single-index models do not have asymptotic standard errors implemented at the moment. You can, however, get robust bootstrapped standard errors by adding the argument `errors = TRUE` to your call to npindex. See the documentation of npindex for further details.

(28) *How do I interpret gradients from the conditional density estimator?*

If you plot a conditional density $f(y|x)$ when x is a scalar, with gradients, by default you will get the following:

(a) A plot of $\partial f(y = \text{median}|x)/\partial x$ (admittedly not the most useful plot). (If y is discrete the only difference is that you get a plot of $\partial f(y = \text{(unconditional) mode}|x)/\partial x$).
(b) A plot of $\partial f(y|x = \text{median})/\partial x$.

If x is multivariate (for example, 2D) you get:

(a) A plot of $\partial f(y = \text{median}|x1, x2 = \text{median})/\partial x_1$
(b) A plot of $\partial f(y = \text{median}|x1, x2 = \text{median})/\partial x_2$
(c) A plot of $\partial f(y = \text{median}|x1 = \text{median}, x2)/\partial x_1$
(d) A plot of $\partial f(y = \text{median}|x1 = \text{median}, x2)/\partial x_2$
(e) A plot of $\partial f(y|x1 = \text{median}, x2 = \text{median})/\partial x_1$
(f) A plot of $\partial f(y|x1 = \text{median}, x2 = \text{median})/\partial x_2$

References

[1] Francisco Cribari-Neto and Spyros G Zarkos. R: Yet another econometric programming environment. *Journal of Applied Econometrics*, 14(3):319–29, May-June 1999. Available at http://ideas.repec.org/a/jae/japmet/v14y1999i3p319-29.html.

[2] Grant V. Farnsworth. Econometrics in R. Technical report, June 2006. Available at http://cran.r-project.org/doc/contrib/Farnsworth-EconometricsInR.pdf.

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

[4] Q. Li and S. Wang. A simple consistent bootstrap test for a parametric regression functional form. *Journal of Econometrics*, 87:145–165, 1998.

[5] J. S. Racine and R. Hyndman. Using R to teach econometrics. *Journal of Applied Econometrics*, 17(2):175–189, 2002.

Changes from Version 0.20-2 to 0.20-2

- Allow for evaluation outside of discrete support of factors in npksum and fixed a warning in jksum
- Fixed a bug which lead to unpredictable behavior when there were more categorical values for the training data than realisations

Changes from Version 0.20-0 to 0.20-1

- Work-around for scale-factor issues during npregbw cv when changing the training data

Changes from Version 0.14-3 to 0.20-0

- npksum now supports an expanded set of kernels (including convolution, derivative and integral), which can be selected via the 'operator' argument
- Automatic bandwidth searches are now performed when attempting to evaluate on data without bandwidths. This allows users to combine bandwidth selection and estimation in one step
- The npsigtest interface is brought in line with other functions (S3)
- Significance tests can now be performed on npreg outputs, so npsigtest(modelname) is now supported
- Added a vignette and faq. To see the vignette try `vignette("np",package="np")`
- summary on npconmode now properly retrieves names from bandwidth objects
- Fixed the 6th and 8th order epanechnikov kernels
- Fixed some quietness issues

- npplot now returns data upon request for conditional densities
- npreg and npcdens now take the appropriate limits in some pathological cases
- User supplied bandwidths now operate seamlessly with the formula interface

Changes from Version 0.14-2 to 0.14-3

- Fixed a glitch that only arose when using the 'liracine' unordered kernel in the presence of irrelevant variables. The upper bound for numerical search was constrained to be (c-1)/c [that for the aitchisonaitken unordered kernel] but ought to have been 1. The summary output would therefore show a value of lambda hitting the (smaller) upper bound (c-1)/1 when it may have hit the (larger) upper bound 1

Changes from Version 0.14-1 to 0.14-2

- Relaxed checking tolerances slightly to prevent spurious 'invalid bandwidth' errors
- Empty sections were removed from help files
- example(foobar) now works again. This was disabled in 0.14-1 at the request of the `R` maintainers in order to shorten the duration of `R CMD check`. All examples remained in the help files but due to the presence of 'dontrun' they were not run when examples(foobar) is requested. Now a limited subset are run while the full set of examples remain in the documents

Changes from Version 0.13-1 to 0.14-1

- Now use optim for minimisation in single index and smooth coefficient models
- Fixed bug in klein-spady objective function
- Standard errors are now available in the case of no continuous variables
- Summary should look prettier, print additional information
- Tidied up lingering issues with out-of-sample data and conditional modes
- Fixed error when plotting asymptotic errors with conditional densities
- Fixed a bug in npplot with partially linear regressions and plot.behavior='data' or 'plot-data'
- Maximum default number of multistarts is 5
- Least-squares cross-validation of conditional densities uses a new, much faster algorithm
- New, faster algorithm for least-squares cross-validation for both local-constant and local linear regressions

NB: The estimator has changed somewhat: both cross-validation and the estimator itself use a method of shrinking towards the local constant estimator when singularity would otherwise lead to the breakdown of the estimator. This arises in sparse data settings in conjunction with small bandwidths for one or more regressor

- Optimised smooth coefficient code, added ridging
- Fixed bug in uniform CDF kernel
- Fixed bug where npindexbw would ignore bandwidth.compute = FALSE and compute bandwidths when supplied with a preexisting bw object
- Now can handle estimation out of discrete support
- Summary would misreport the values of discrete scale factors which were computed with bwscaling = TRUE

Changes from Version 0.12-1 to 0.13-1

- Bandwidths are now checked for validity based on their variable and kernel types
- np now does a better job of preserving names of some 'y' data
- Names of coefficients returned from coef() now match variable names
- Fixed some corner cases in npksum involving the dimensionality of outputs
- Fixed deprecation warnings in R 2.5.0 caused by use of $ on atomic objects
- Various and sundry bug fixes in npscoef
- npscoef now handles discrete 'z' data
- Predict now accepts the argument 'se.fit', like predict.lm
- Fixed bug where incorrect asymptotic standard errors of gradients for regression objects were being displayed in npplot()
- Fixed bug where errors of gradients of regression objects were not being returned in matrix form
- vcov() now works with partially linear regression objects
- Fixed detection of evaluation responses when using the formula interface
- Pre-computed bandwidth objects are now provided for some of the more computationally burdensome examples
- Added Jeffrey Wooldridge's WAGE1 dataset with qualitative variables (married, female, nonwhite)
- Predictions outside of discrete support for regressions and conditional densities are now allowed
- Fixed sign issue with scaling of standard errors in the single index model
- Fixed error when calculating some bandwidths/scale factors for display purposes

- Bug in passing certain arguments to npcdensbw fixed
- Added predict method for qregression objects
- Proper normalisation for liracine kernel shown in summary
- Fixed output bug (^H) in summary method for sigtest objects
- Fixed regression with plotting of bootstrapped errors in perspective plots
- npcdist no longer incorrectly calls npcdens
- Fixed spacing between var name and p-value in significance test summaries