Nonparametric Kernel Smoothing Methods. The `sm` library in *Xlisp-Stat*.

Luca Scrucca
Department of Statistics
Università degli Studi di Perugia, Italy
`luca@stat.unipg.it`

June 14, 2001

Abstract

In this paper we describe the *Xlisp-Stat* version of the `sm` library, a software for applying nonparametric kernel smoothing methods. The original version of the `sm` library was written by Bowman and Azzalini in *S-Plus*, and it is documented in their book *Applied Smoothing Techniques for Data Analysis* (1997). This is also the main reference for a complete description of the statistical methods implemented.

The `sm` library provides kernel smoothing methods for obtaining nonparametric estimates of density functions and regression curves for different data structures. Smoothing techniques may be employed as a descriptive graphical tool for exploratory data analysis. Furthermore, they can also serve for inferential purposes as, for instance, when a nonparametric estimate is used for checking a proposed parametric model. The *Xlisp-Stat* version includes some extensions to the original `sm` library, mainly in the area of local likelihood estimation for generalized linear models.

The *Xlisp-Stat* version of the `sm` library has been written following an object-oriented approach. This should allow experienced *Xlisp-Stat* users to implement easily their own methods and new research ideas into the built-in prototypes.

KEY WORDS: Kernel smoothing methods; nonparametric density estimation; nonparametric regression; graphics; local likelihood for generalized linear models; *Xlisp-Stat*; *Arc*.

Contents

1 Introduction

This document describes the *Xlisp-Stat* version of the `sm` library for applying nonparametric kernel smoothing methods. The original version of the `sm` library was written by Bowman and Azzalini in *S-Plus*, and it is documented in their book *Applied Smoothing Techniques for Data Analysis* (1997). Algorithmic aspects of the software are discussed by Bowman and Azzalini (2001). All the statistical methods discussed by the two authors have been translated following an object-oriented approach. In addition, further nonparametric methods have been implemented, mainly in the area of local likelihood. The `sm` library have been developed and checked using *Xlisp-Stat* 3.52.17 and *Arc* 1.03, both on Windows and Unix systems.

The computer code that implements the `sm` library is available at the Web address `http://www.stat.unipg/luca/xlispstat/sm.zip`. The library contains the following files:

- `sm-density.lsp` nonparametric density estimation
- `sm-regression.lsp` nonparametric regression estimation
- `sm-glm.lsp` local likelihood estimation for generalized linear models
- `sm-survival.lsp` nonparametric regression with survival data
- `sm-functions.lsp` set of functions used for binning, bandwidth selection, etc.
- `sm-miscfun.lsp` set of miscellaneous functions of general utility
- `sm-addson.lsp` set of statistical functions and methods added to the `sm` library from other sources

Furthermore, the file `sm-readme.txt` provides information on how to install the library, whereas the file `sm-library.lsp` contains code for loading all the needed files, compiling the library to speed-up computations, and provides an help through a windows user interface.

Scripts for running the examples from the book by Bowman and Azzalini are contained in the file `sm-script.lsp`, and datasets discussed by the two authors are also available in *Arc* format (see Section 9). Moreover, scripts for reproducing the examples used in the present document are available at the bottom of the file `sm-script.lsp`.

The following sections will briefly describe the main functions and provide some examples. Further details on each function are provided through the on-line help discussed in Section 1.2. As a general philosophy, during the translation from *S-Plus* to *Xlisp-Stat* we tried to keep as much as possible things similar to the original `sm` library. From a user perspective there should be no great differences between the two libraries. Neverthless, the way the functions are called and arguments are passed necessarily differ due to the differences in the two statistical environments. For a full understand of the usage of the library we suggest to repeat the examples from the book of Bowman and Azzalini (1997) using the scripts provided in the file `sm-script.lsp`.

A basic knowledge of the *lisp* language is assumed. A readable introduction is provided by Tierney (1990), and further documents on *Xlisp-Stat* can be obtained at the address `http://www.stat.umn.edu/~luke/xls/xlsinfo/xlsinfo.html`. *Arc*, a computer program written in *Xlisp-Stat* for the analysis of regression data as described in Cook and Weisberg (1999), can be obtained from the Web at `http://www.stat.umn.edu/arc`. An archive of *Xlisp-Stat* programs may be found on the Web at `http://www.xlispstat.org`, whereas a set of useful links is available at the addess `http://www.visualstats.org`. The book *Common Lisp* by Guy L. Steele (1990) is a comprehensive source for the *lisp* language, and it is available on-line at `http://www.cs.cmu.edu/Groups/AI/html/cltl/cltl2.html`.

1.1 Installing and loading the `sm` library into *Xlisp-Stat*

Assuming that a version of *Xlisp-Stat* or *Arc* is already installed on your computer, the following steps are needed for installing the `sm` library:

1. get the `sm.zip` file;
2. for inflating the archive in Windows OS you may use Winzip, while in Unix/Linux you need to type

   ```
   > unzip -a sm.zip
   ```

   Stuffit Expander may be used in Macintosh OS.
3. copy the files extracted from the zipped archive in a directory you have created, for example one called `sm`;
4. copy the file `sm-library.lsp` in the directory where you run *Xlisp-Stat* or *Arc*. In the latter case, you may want to copy the file in the `Extras` sub-directory so it will be automatically loaded at the start-up;
5. if needed modify the `*sm-home-directory*` at the beginning of the file `sm-library.lsp`. This variable should contain the directory to be added to your search paths. By default, `*sm-home-directory*` is set to the directory `sm` relative to your working directory.

   For example, suppose you are working on Unix/Linux system, and you run *Xlisp-Stat* or *Arc* from the directory `xls`. Then, by default the variable `*sm-home-directory*` is set to `"sm/"`. However, if the source `sm` lisp files are located in a different directory, say `"~user/xls/library/sm"`, then you must edit the file `sm-library.lsp` as follows:

   ```
   (defparameter *sm-home-directory* #+macintosh "Extras:sm:"
                                     #+unix      "library/sm/"
                                     #+msdos     "Extras\\sm\\")
   ```

   So, we simply changed the second row from `"sm/"` to `"library/sm/"`. For other systems the procedure is analogous, provided that the right row is modified depending on your system.

Once you have installed the `sm` library, you may simply run *Xlisp-Stat* or *Arc* and load the library as follows:

```
> (load "sm-library")
```

If you are using *Arc* the `sm` library is automatically loaded at start-up if the file `sm-library.lsp` is located on the:
Windows: `Extras` sub-directory of your `Arc` directory;
Macintosh: `Extras` sub-folder of your `Arc` folder;
Unix/Linux: `Extras` sub-directory of the directory from which you start *Arc*.

Computational efficiency may be speed-up compiling the library. The function

```
> (sm-library-compile)
```

will do the job for you, and the resulting `.fsl` files will be saved on the directory specified by `*sm-home-directory*`.

1.2 Getting on-line help

A very basic on-line help system can be invoked through the command

```
> (sm-help)
```

This function opens a window like that on Figure 1. From this window you may select an item and click on the **Print** button (or double-clicking on the item itself) to see the corresponding documentation on the listener.

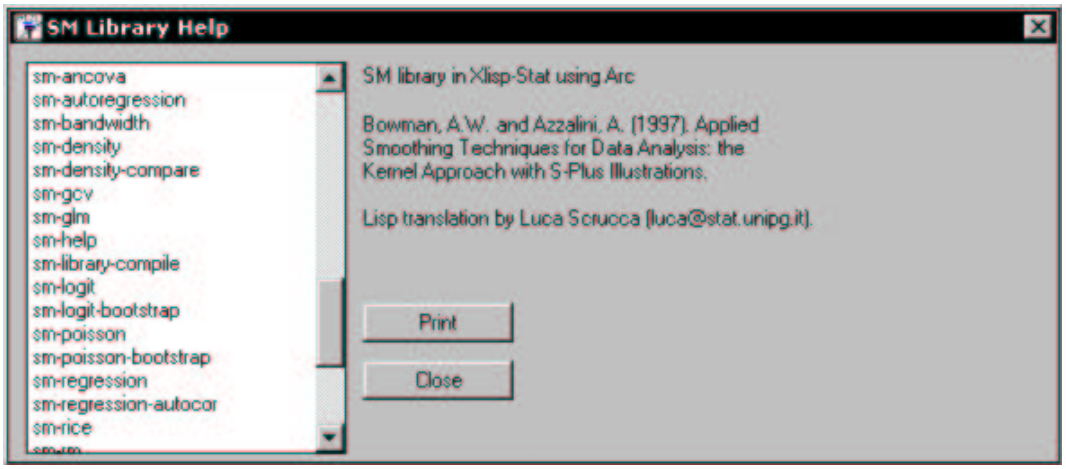

Figure 1: Help dialog for the `sm` library.

The standard *lisp* procedure can also be used to obtain the documentation associated with a function. For example, suppose you want to display the help for the `hnorm` function. You need to type the following in the text windows command line following the prompt:

```
> (help 'hnorm)
HNORM                                                          [function-doc]
-------------------------------------------------------------------------------
    Normal optimal choice of smoothing parameter in density estimation
-------------------------------------------------------------------------------
This function evaluates the smoothing parameter which is asymptotically optimal
for estimating a density function when the underlying distribution is Normal.

Args: (x &optional weights)

...
```

2 Nonparametric density estimation

A univariate kernel density estimator for a continuous variable based on a sample $\{y_1, \ldots, y_n\}$ at the evaluation point y can be expressed as

$$\hat{f}(y) = \frac{1}{n} \sum_{i=1}^{n} w(y - y_i; h)$$

where $w(.)$ is called kernel function, a symmetric weighting function, and h is the smoothing parameter or bandwidth. The `sm` library uses a normal kernel function centered at the evaluation point with mean 0 and standard deviation h, so that

$$w(y - y_i; h) = \phi(y - y_i; h)$$

where $\phi(z; h)$ denotes the normal density function in z with mean 0 and standard deviation h.

A two-dimensional density estimate can be constructed using a two-dimensional kernel function in the form

$$\hat{f}(y_1, y_2) = \frac{1}{n} \sum_{i=1}^{n} w(y_1 - y_{1i}; h_1) w(y_2 - y_{2i}; h_2)$$

where (h_1, h_2) are the joint smoothing parameters, and the kernel function is obtained as the product of two univariate kernels.

2.1 The `sm-density` function

The function `sm-density` allows nonparametric density estimation in one or two dimensions:

```
                                    sm-density
Args: (x &key h (hmult 1)
              (h-weights nil) (weights nil)
              (binning nil)  (nbins nil)
              eval-points positive delta
              (model "none")
              (display "estimate") if 1D or (display "persp") if 2D
              (add nil)
              (props (list 75 50 25))
              xlab ylab xlim ylim yht
              (ngrid *sm-ngrid*)
              (points-on-plot T)
              (color 'black)
              (symbol 'cross) if 1D or (symbol 'disk) if 2D
              (type 'solid) (width 1))
```

The only required argument is the data `x`, which can be a list in the one-dimensional case, or a list of lists or a matrix with two columns in the two-dimensional case. The keyword argument `:h` provides the smoothing parameter (or two smoothing parameters in the two-dimensional case, one for each dimension), and if this parameter is omitted a normal optimal smoothing parameter is used (in the two-dimensional case the default smoothing parameter is computed separately for each dimension).

In large datasets a procedure called *binning* can be used for increasing the computational speed. Setting the keyword argument `:binning` to `T`, the kernel density estimate is computed on the binned data with number of bins provided by `:nbins`, or if not provided it is computed by the `nbins` function. For further details on binning see Section 4.4.

Variable bandwidths may be used throught the keyword argument `:h-weights`, a list of weights which multiply the smoothing parameter used in the kernel function at each observation (for details on the methodology see Bowman and Azzalini, 1997, p. 17, and the example on script 1.11).

A list of weights may be assigned to each observation using the keyword argument `:weights`. Weights are used, for example, in the computation of density estimate from binned data.

The points at which the density estimator should be evaluated can be set using the keyword argument `:eval-points`. If not provided, a regular grid is used for computing the estimates with number of points given by `:ngrid`. By default `:ngrid` provided by the global variable `*sm-ngrid*`: for one-dimensional density estimate this gives the number of points in the regular grid used, whereas for two-dimensional data, approximately `*sm-ngrid*`/2 number of points for each axis are used.

Data with positive bounded support can be handled setting `:positive` to `T`. In this case, a log transformation is applied to the data before construction of a density estimate. The result is then transformed back to the original scale (the method is described in Bowman and Azzalini, 1997, pp. 14-16).

The graphical output is controlled by the keyword argument `:display`. The setting `"none"` or `NIL` will prevent any graphical output from being produced. In one dimension, the default setting `"estimate"` will produce the density estimate, while the setting `"se"` will in addition produce a variability band, showing the variability, but not the bias, of the estimate. In two dimensions, the default setting `"persp"` will produce a perspective plot of the density estimate, while the setting `"slice"` will produce a slice or contour plot. Observed points are added to the plot if `:points-on-plot` is set to T, unless a perspective plot is drawn.

In one dimension a reference band, indicating where a density estimate is likely to lie when the data are normally distributed, may be superimposed on the plot setting the keyword argument `:model` to `"Normal"`. The default `"none"` assumes no reference model.

By default any plot is drawn on a new window, but a plot may be added to an existing one if you provide a graph object to the keyword argument `:add`.

The function `sm-density` returns a `sm-density-proto` object as its result. Hence, the user may assign the returned object to a variable and then send messages to it.

Example 2.1 (aircraft span data) Consider the data on aircraft technology described in Section 1.2 of Bowman and Azzalini (1997). This dataset can be analyzed in *Arc* loading the file `sm-aircraft.lsp`. Suppose we want to obtain a density estimate for the aircraft span on a log scale for the period 1956–1984. The following expressions must be typed in the listener window:

```
> (load "sm-aircraft")
> (def y3 (log (select span (which (= period 3)))))
> (def sm (sm-density y3 :xlab "Log Span"))
```

After loading the data, a variable containing the log-span for the third period is defined. Then, the last line produces a graph of the density estimate and returns a `sm-density-proto` object which is assigned to the variable `sm` (but you may want to use a different name as well). Slot values can be retrieved from this object by sending the messages summarized in Table 1. The results from the density estimation are stored in the slots shown in Table 2.

Information from the resultant object can be obtained by sending to it the `:help` message:

```
> (send sm :help)
SM-DENSITY-PROTO
Nonparametric Kernel Density Estimation.
Translation by Luca Scrucca from the S-Plus library "sm" by Bowman, A.W. and
Azzalini, A. (1997). Applied Smoothing Techniques for Data Analysis: the Kernel
```

```
Approach with S-Plus Illustrations. Oxford University Press, Oxford.
Help is available on the following:

ADD ADD-METHOD ADD-SLOT BAND COLOR COMPUTE DELETE-DOCUMENTATION DELETE-METHOD
DELETE-SLOT DELTA DENSITY-EVAL-1D DENSITY-EVAL-2D DENSITY-POSITIVE-1D
DENSITY-POSITIVE-2D DIAGNOSTIC-PLOTS DISPLAY DOC-TOPICS DOCUMENTATION EST-CDF
ESTIMATE EVAL-POINTS GET-METHOD GRAPH H H-WEIGHTS HAS-METHOD HAS-SLOT HELP
HMULT INTERNAL-DOC ISNEW LOWER MAKE-CLONE METHOD-SELECTORS MODEL NBINS NDIM
NEW NGRID NORMDENS-BAND NORMDENS-BAND-LOWER NORMDENS-BAND-UPPER OWN-METHODS
OWN-SLOTS PARENTS PLOT PLOT-DENSITY-2D-PERSP PLOT-DENSITY-2D-SLICE
POINTS-ON-PLOT POSITIVE PRECEDENCE-LIST PRINT PROPS PROTO REPARENT RETYPE
SE SHOW SLOT-NAMES SLOT-VALUE SYMBOL TYPE UPPER WEIGHTS X X-OBS XLAB XLIM
YHT YLAB YLIM
```

Help on a specific topic can be obtained as follows:

```
> (send sm :help :plot)
PLOT
Args: ()
Plots the density estimate unless the display slot is set to "none" or NIL.
```

Table 1: Methods for accessing slot values in a `sm-density-proto` object

Method	Description
`:x`	data used in density estimation, i.e. the observed data or the binned data if binning is used
`:x-obs`	observed data
`:h`	value of the smoothing parameter
`:hmult`	multiplier for the smoothing parameter (by default equal to 1)
`:h-weights`	weigths applied to the smoothing parameter
`:nbins`	number of bins used if any, `NIL` otherwise
`:ndim`	number of dimensions
`:ngrid`	number of points used in a regular grid for density estimation
`:eval-points`	points where the kernel density estimation has been evaluated
`:positive`	logical flag for data with positive bounded support
`:weights`	weights associated to each observation
`:model`	reference model
`:band`	if a band for normality is computed (`T`) or not (`NIL`)
`:display`	controls the graphical output
`:points-on-plot`	logical flag which controls whether or not observed points are drawn
`:props`	list defining the proportions of the data to be included within each contour in a slice plot for two-dimensional data
`:add`	object address which contains the graphical output if required, `NIL` otherwise

As an example of their use, suppose we want to add to the previous plot a density estimate for the other two periods. This is very easy, since we need only to get the graph address from the sm-density object created, and then call the function `sm-density` specifying that we want to add the density estimate to an existing graph.

```
> (def y1 (log (select span (which (= period 1)))))
> (def y2 (log (select span (which (= period 2)))))
```

Table 2: Methods for accessing results from a `sm-density-proto` object

Method	Description
`:estimate`	density estimate at each evaluation point
`:se`	standard error at each evaluation point
`:upper`	variability band upper limit at each evaluation point, computed as the density estimate plus 2 times the standard errors
`:lower`	variability band lower limit at each evaluation point, computed as the density estimate minus 2 times the standard errors
`:normdens-band-upper`	upper limit for the normality band
`:normdens-band-lower`	lower limit for the normality band
`:graph`	address of the last graph drawn

```
> (sm-density y2 :add (send sm :graph) :color 'blue :type 'dashed)
> (sm-density y1 :add (send sm :graph) :color 'cyan)
> (send (send sm :graph) :legend 4 1 '("Period = 1" "Period = 2" "Period = 3")
                         :color '(black blue cyan)
                         :type '(solid dashed solid)
                         :width '(1 1 1)
                         :frame T)
```

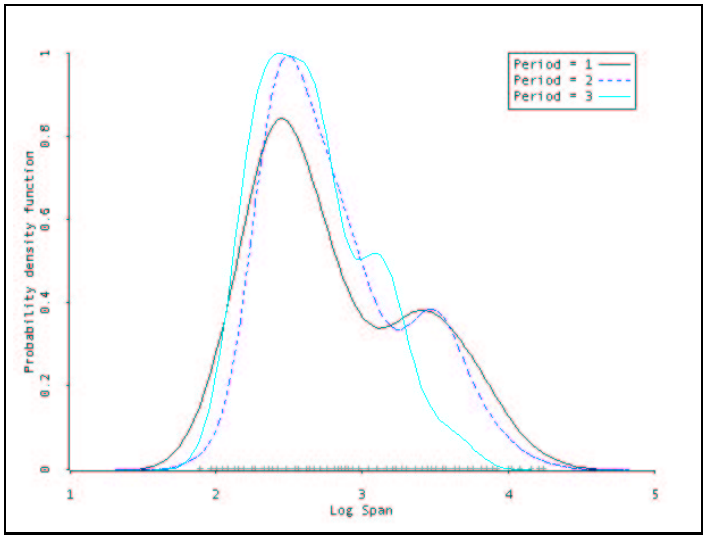

Figure 2: Density estimation for the log span data within each period from the `sm-aircraft.lsp` dataset.

The final plot is shown in Figure 2. ■

Example 2.2 (tephra data) As a further example, script 2.6 contained in the file `sm-script.lsp` shows how to obtain reference bands for normality using the tephra data. These data record the percentages of aluminium oxide found in samples from a tephra layer resulting from a volcanic eruption in Iceland around 3500 years ago. Since the variable `Al2O3` is a percentage, before computing the density estimate a logit transformation is applied.

```
> (load "tephra")
> (def logit (log (/ Al2O3 (- 100 Al2O3))))
```

```
> (sm-density logit :model "Normal" :xlab "logit(Al203)" :points-on-plot nil)
```

The density estimate is shown on the left panel of Figure 3, together with reference bands for normality. The argument `:points-on-plot` sets to `NIL` implies that observed points are not drawn at the bottom of the plot.

In the previous estimate we did not provide a bandwidth, so by default a normal optimal bandwidth has been used (see 4.2, and Bowman and Azzalini, 1997, pp. 31-32). It is known that bandwidths chosen using this criterion tend to oversmooth if the underlying density is not normal. Perhaps a better approach would be to select the bandwidth based on some other more general criterion, as for example by cross-validation (see Section 4.2, and Bowman and Azzalini, 1997, pp. 32-34). The following instrunctions allow to use the cross-validation criterion to select the bandwidth, and then compute the density estimate.

```
> (hnorm logit)
0.0263566
> (setf hcv (hcv logit))
0.017678
> (sm-density logit :h hcv :model "Normal" :xlab "logit(Al203)")
```

The `hnorm` function returns the normal optimal smoothing parameter, which is larger than the bandwidth choosen by cross-validation using the `hcv` function. The density estimate based on the latter bandwidth is shown on the right panel of Figure 3. Bowman and Azzalini (1997, p. 40, script 2.6) uses the Sheather-Jones plug-in criterion to select the bandwidth, and this provides a value of h very close to that selected by cross-validation. ■

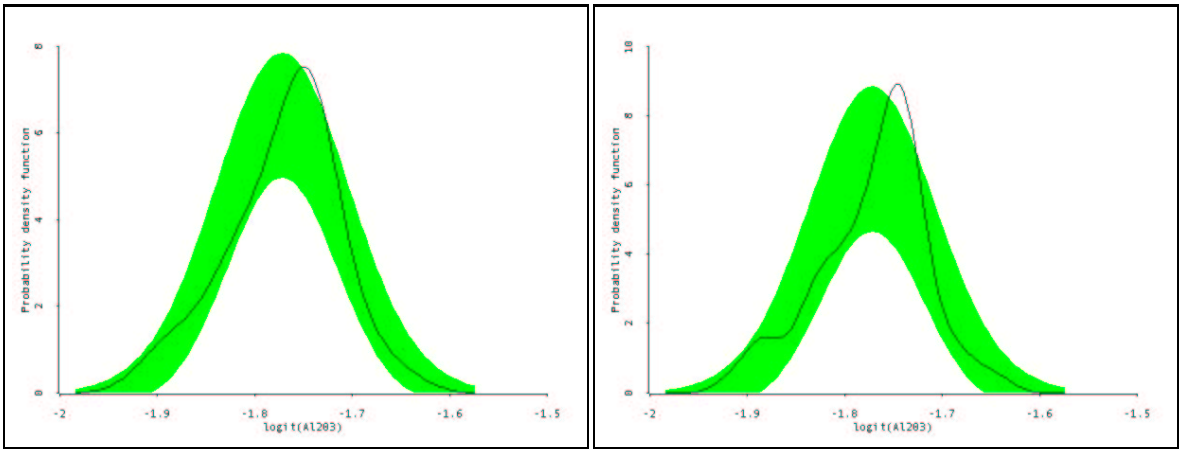

Figure 3: Density estimates of the `tephra.lsp` data with reference bands for normality. On the left panel a normal optimal smoothing parameter has been used, whereas a smoothing parameter selected by cross-validation on the right panel.

Table 3: Methods for computing and graphing density estimates

Method	Description
`:compute`	starts computing density estimates based on the information stored in the slots of the current object
`:plot`	draws a plot of the current density estimates

Table 3 may be of interest to experienced *Xlisp-Stat* users. The two methods shown are responsible for computing and plotting the density estimates. This may be helpful if we want to re-compute or re-

graph the density estimate based on the same data but changing one or more parameters. For instance, in some circumstances we would like to investigate the sensitivity of our nonparametric estimate to bandwidth size changing. Thus, we may define different bandwidths and plotting the results on the same graph to allow comparisons. Two approaches may be adopted. The first approach defines the list of values for the bandwidth, computes the first density estimate, and then adds the successive estimates to the existing plot. Taking a different approach, we may define only one `sm-density-proto` object, and then repeatedly modify the parameter `hmult` to control the amount of smoothing applied.

```
> (def sm (sm-density logit :xlab "logit(Al203)"))
> (send sm :h)                                    get the default bandwidth used
0.0263566
> (send sm :hmult)                                get the default multiplier of h
1
> (def hmult (list 0.5 0.75 1.25 1.5))            define a list of values for hmult
> (send sm :add (send sm :graph))                 set the plot for adding the density estimates
> (dotimes (i (length hmult))                     starts the loop
           (send sm :hmult (select hmult i))      ... re-defines the value of the slot hmult
           (send sm :compute)                     ... re-computes the estimates
           (send sm :plot))                       ... graphs the results
```

Example 2.3 (aircraft span data) Data on the aircraft span based on the first two principal components can be used as an example of density estimation in two dimensions (see also script 1.4). After loading the dataset, a perspective plot and a contour plot using the default normal bandwidth can be obtained as follows:

```
> (load "sm-aircraft-pc")
> (def pc3 (list (select Comp1 (which (= period 3)))
                 (select Comp2 (which (= period 3)))))
> (sm-density pc3 :xlab "Comp1" :ylab "Comp2")
> (sm-density pc3 :display "slice" :xlab "Comp1" :ylab "Comp2")
```

■

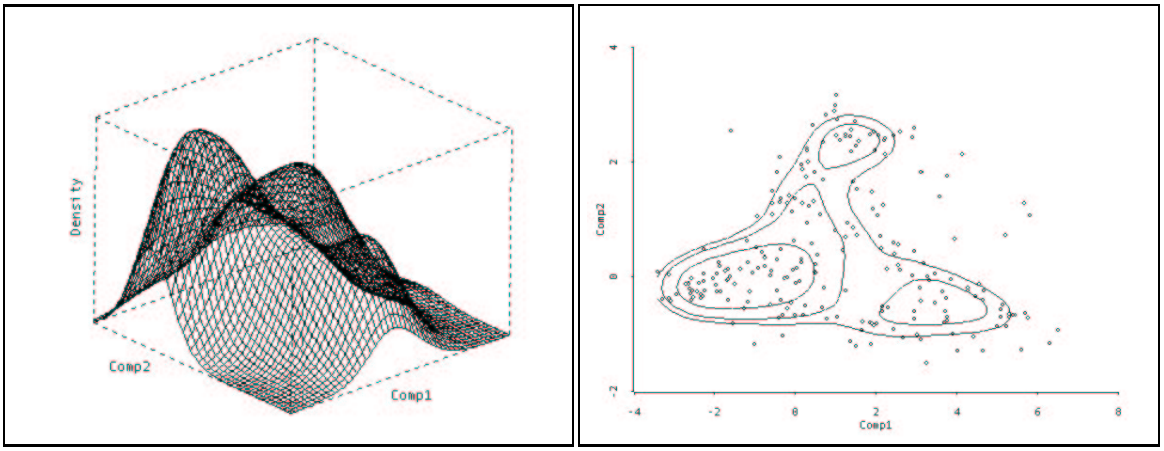

Figure 4: Bivariate density estimates of the Aircraft span data in the file `sm-aircraft-pc.lsp` The left panel shows a perspective plot, whereas the right panel a contour plot with default contour at 75, 50 and 25 percent levels.

2.2 Comparison of univariate density estimates

`sm-density-compare`

```
Args: (x group &key h
                    (display "lines")
                    (model "none")
                    (test nil test-set)
                    (band T)
                    (ngrid *sm-ngrid*)
                    (nsim 100)
                    (verbose T)
                    xlab ylab )
```

This function allows a set of univariate density estimates to be compared, both graphically and formally in a bootstrap hypothesis test of equality (see Bowman and Azzalini, 1997, Section 6.2). The data `x` are grouped based on the argument `group`, and the same smoothing parameter `h` is used for each group. If `:model` is `"none"` comparison is restricted to plotting only. The alternative value `"equal"` produces a permutation hypothesis test of equality. For two groups, an appropriate reference band is displayed, unless `:band` is set to `NIL`. The number of permutation simulations is controlled by the argument `:nsim`.

2.3 Nonparametric density estimation of stationary time series data

`sm-ts-pdf`

```
Args: (x &key h lags
             (display "persp")
             (ngrid *sm-ngrid* set-ngrid)
             (points-on-plot T)
             (varname "x"))
```

This function estimates the density function of a time series `x`, assumed to be stationary (see Bowman and Azzalini, 1997, Section 7.2). The univariate marginal density is estimated in all cases. Bivariate densities of pairs of lagged values are estimated depending on the keyword argument `:lags`, a list of values for the lags to consider in the joint distribution estimation. The bandwidth can be set by the argument `:h`, and if missing a normal optimal bandwidth is used. The `:display` argument controls the bivariate density plot, with possible values `"persp"` (default) and `"slice"`.

3 Nonparametric regression estimation

A general formulation for a nonparametric regression model takes the form

$$y = m(x) + \epsilon$$

where y denotes the response variable, x the explanatory variable and ϵ an independent error term with zero mean and variance equal to σ^2. The aim of a nonparametric regression is to provide an estimate of the smooth function $m(.)$.

A simple kernel estimator of $m(.)$ is the *local mean estimator* (see Bowman and Azzalini, 1997, p. 49)

$$\hat{m}(x) = \frac{\sum_{i=1}^{n} w(x_i - x; h) y_i}{\sum_{i=1}^{n} w(x_i - x; h)}$$

where the kernel function $w(z; h)$ is a smooth positive function, wich gives weights that decrease monotonically as $|z|$ increases in size. As for density estimation, a normal kernel function centered at the evaluation point with mean 0 and standard deviation h is used. Therefore, h acts as smoothing parameter.

The local mean estimator provides an estimate $\hat{m}(x)$ by computing a local average in the neighbour of the evaluation point x. A better estimator, in particular near the edges of the region over which the data have been collected, is the *local linear estimator*. This involves solving the following least squares problem

$$\min_{\alpha,\beta} \sum_{i=1}^{n} \{y_i - \alpha - \beta(x_i - x)\}^2 w(x_i - x; h)$$

and taking as the estimate at the evaluation point x the value $\hat{\alpha}$ (see Bowman and Azzalini, 1997, p. 50). Extension to two dimensions is straightforward (see Bowman and Azzalini, 1997, p. 52-53).

3.1 The `sm-regression` function

The function `sm-regression` can be used for nonparametric regression estimation in one or two dimensions:

```
                                    sm-regression
Args: (x y &key (h nil) (hmult 1)
                (h-weights nil) (weights nil)
                (binning nil) (nbins nil)
                (model "none")
                (display "lines") if 1D (display "persp") if 2D
                (print T)
                (add nil) levels
                (test nil) (poly-index 1)
                (design-mat nil)
                (ngrid nil) eval-points
                 xlab ylab zlab levels
                (points-on-plot T)
                (color 'black) (symbol 'disk)
                (type 'solid) (width 1))
```

The required arguments are the covariate(s) `x` (list of values in the one covariate case, or a list of lists or a matrix with two columns in the two covariates case), the response variable `y` (a list of values), and the smoothing parameter `h` (a list of one or two values, for the one- and two-dimensional case respectively).

The keyword argument `:poly-index` controls whether local mean estimator (0) or a local linear estimator (1) is used.

The graphical output is controlled by the keyword argument `:display`. The setting `"none"` or `NIL` will prevent any graphical output from being produced. With one covariate, the default setting `"lines"` will produce the regression estimate, while the setting `"se"` will in addition produce a variability band, showing the variability, but not the bias, of the estimate (see Bowman and Azzalini, 1997, pp. 75-77). With two covariates, the default setting `"persp"` will produce a perspective plot of the regression estimate, while the setting `"contour"` will produce a contour plot with levels specified by the argument `:levels`. This must be list of heights of contour lines, and by default 5 levels are used to drawn contours which cover the range of the smooth estimate of y.

The keyword argument `:model` defines the reference model. The values `"none"`, `"no effect"` and `"linear"` are possible. When a model is provided, a reference band for the reference model is plotted on the graph. Moreover, if `:test` is `T` a formal test is produced using the reference model as the null hypothesis (see Bowman and Azzalini, 1997, Sections 5.2, 5.3).

Most of the remaining arguments have the same meaning we have already discussed for the `sm-density` function. Details can be obtained throught the on-line help.

The function `sm-regression` returns a `sm-regression-proto` object as its result. Then, the user may assign the returned object to a variable and send messages to it.

The slot values can be retrieved sending the messages summarized on Table 4, and the results from the nonparametric regression are stored in the slots reported on Table 5. As we saw for the density estimation case, slot values can be modified and the needed computations invoked sending the message `:compute`, while a new plot may be obtained through the `:plot` method (see Table 3.1).

Table 4: Methods for accessing slot values in a `sm-regression-proto` object

Method	Description
`:x`	data for the covariate(s), i.e. the observed data or the binned data if binning is used
`:x-obs`	observed data for the covariate(s)
`:y`	data on the response variable, i.e. the observed data or the binned data if binning is used
`:y-obs`	observed data on the response variable
`:h`	value of the smoothing parameter
`:hmult`	multiplier for the smoothing parameter (by default equal to 1)
`:h-weights`	weigths applied to the smoothing parameter
`:nbins`	number of bins used if any, `NIL` otherwise
`:ndim`	number of dimensions
`:ngrid`	number of points used in a regular grid for estimation
`:eval-points`	points where the nonparametric smoothing regression have been evaluated
`:weights`	weights associated to each observation
`:poly-index`	the estimator used, 0 for local mean estimator and 1 for local linear estimator
`:model`	reference model
`:band`	when a reference model is provided, this slot is set to `T` and a band will be plotted on the graph (only for one covariate); setting this slot to `NIL` will prevent any band to be plotted
`:display`	controls the graphical output
`:points-on-plot`	logical flag which controls whether or not observed points are drawn
`:add`	object address which contains the graphical output if required, `NIL` otherwise

Example 3.1 (brain weight data) The following example considers the data on brain weight (`BrainWt`) in grams and body weight `BodyWt` in kilograms for sixty-two species of mammals. This dataset is described by Cook and Weisberg (1999, Section 5.1) and can be analyzed in *Arc* by loading the file `brains.lsp`. A linear regression model for `BrainWt` on `BodyWt`, both expressed on a log scale, with constant variance function seems reasonable. We may further investigate this linear relationship comparing the estimated parametric model with a nonparametric regression model (see Bowman and Azzalini, 1997, Section 5.3). The cross-validation criterion may be used to select the smoothing parameter, then a plot of the estimated smooth function together with reference bands for linearity allows to perform a graphical check. This can be obtained typing the following commands:

```
> (load "brains")
> (def log[BodyWt] (log BodyWt))
> (def log[BrainWt] (log BrainWt))
> (setf hcv (hcv log[BodyWt] :y log[BrainWt] :display "lines"
```

Table 5: Methods for accessing results from a `sm-regression-proto` object

Method	Description
`:estimate`	regression estimates at each evaluation point
`:estimate-grid`	regression estimates over a grid formed by the evaluation points
`:se`	standard errors at each evaluation point
`:upper`	variability band upper limits at each evaluation point, computed as the regression estimates plus 2 times the standard errors
`:lower`	variability band lower limits at each evaluation point, computed as the regression estimate minus 2 times the standard errors
`:sigma`	estimated standard deviation of the error
`:graph`	address of the last graph drawn from the object
`:regression-test`	computes p-value for a hypothesis test that checks a reference parametric regression model with the estimated nonparametric model.

Table 6: Methods for computing and graphing nonparametric regression estimates

Method	Description
`:compute`	starts computing regression estimates based on the information stored in the slots of the current object
`:plot`	draws a plot of the estimated nonparametric regression model

```
                :hstart 0.3 :hend 1.5 :ngrid 20))
(0.49632 #<Object: f1bcd0, prototype = SCATTERPLOT-PROTO>)
> (def sm (sm-regression log[BodyWt] log[BrainWt] :h (select hcv 0) :model "linear"
                         :xlab "log[BodyWt]" :ylab "log[BrainWt]"))
Test of linear model: significance = 0.1243
```

The function `hcv` returns a list giving the value of h which minimizes the cross-validatory criterion and a plot showing the criterion function plotted over the search grid of smoothing parameters (see top-left panel of Figure 5).

The nonparametric regression analysis is then performed through the function `sm-regression`, which returns an object assigned to the variable `sm` (again, you may want to use a different name as well). The graphical output shown in the bottom panel of Figure 5 is also produced. From this plot we can see that the estimated smooth function lies inside the reference band over almost all the range of the predictor. A formal hypothesis test suggest that there is no strong evidence against the reference linear model. Since this test depends on the smoothing parameter selected, it is possible to obtain the observed significance for a range of smoothing parameters as follows:

```
> (sig-trace sm (rseq 0.3 1.5 10))
((0.3 0.433333 0.566667 0.7 0.833333 0.966667 1.1 1.23333 1.36667 1.5)
 (0.204511 0.129438 0.128668 0.148171 0.164595 0.175607 0.189515 0.211389
  0.241197 0.276249)
 #<Object: c8641c, prototype = SCATTERPLOT-PROTO>)
```

The significance trace plot, shown on the top-right panel of Figure 5, suggests no strong evidence against the null hypothesis (for further details on the methodology see Bowman and Azzalini, 1997, p. 93). ■

Example 3.2 (ethanol data) Consider the dataset contained in the file `ethanol.lsp`, where the goal is to model nitric oxide concentration (`NOx`) from the emissions of a single cylinder engine as a function of

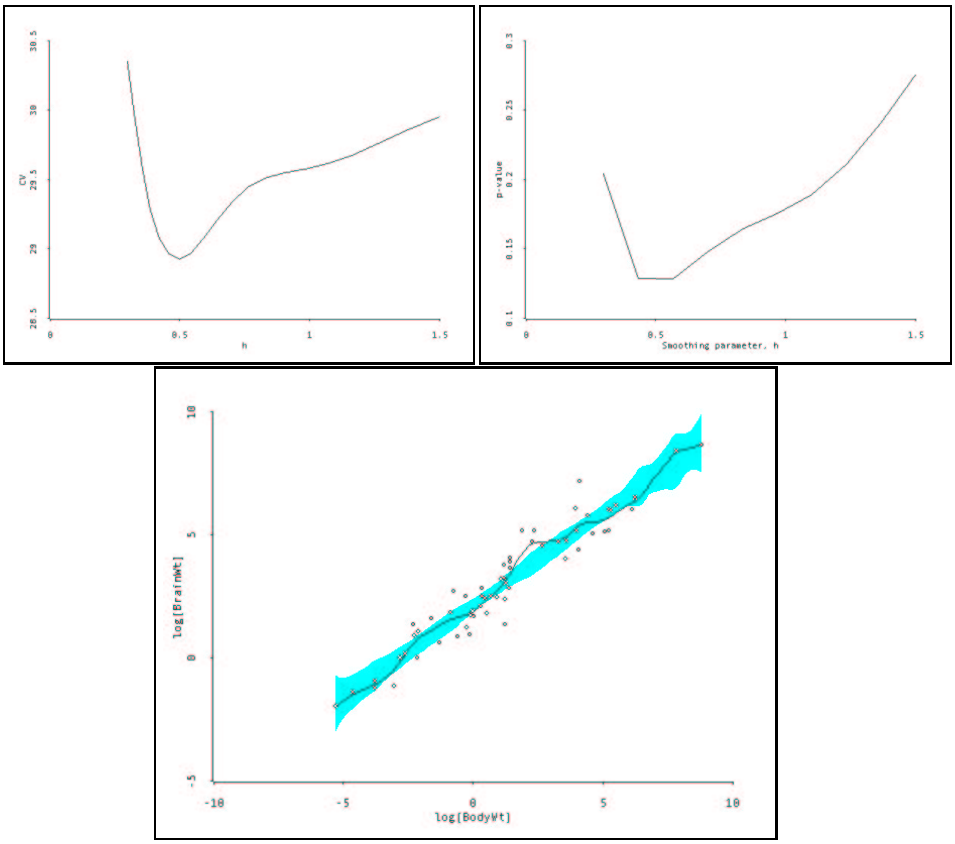

Figure 5: Nonparametric regression analysis of the `brains.lsp` dataset. The top-left panel shows the CV criterion evaluated over a grid of h values, while the top-right panel shows the significance trace plot. The graph on the bottom is the plot of the estimated smooth function with reference bands for linearity.

equivalence ratio (`E`), a measure of the richness of the air and fuel mix in the engine, and of compression ratio of the engine (`C`). This dataset has been studied extensively, among others, by Cleveland (1993). We start by loading the data and using the function `hcp` for selecting the bandwidths (see Section 5.4), both separately and jointly for the two predictors. We also provide the corresponding approximate span computed as defined in Section 4.1.

```
> (load "ethanol")
> (hcp E NOx :display T)
(0.0205786 #<Object: eff744, prototype = SCATTERPLOT-PROTO>)
> (sm-span E 0.025786)
0.221974
> (hcp C NOx :hstart 1 :hend 15 :ngrid 15 :display T)
hCP: boundary of search area reached.
Try readjusting hstart and hend.
hstart: 1
hend  : 15
Values of h and CP:
1  4.77802
1.21341  4.2412
1.47236  3.49723
1.78657  2.76979
2.16783  2.14982
```

```
2.63047  1.61278
3.19183  1.13083
3.87298  0.716118
4.69951  0.397845
5.70241  0.173718
6.91935  0.0206948
8.39599  -0.0830884
10.1877  -0.153441
12.3619  -0.201155
15  -0.233531
Warning: Dumped
> (hcp (list E C) NOx :hstart 1 :hend 15 :ngrid 15 :display T)
((0.434613 8.39571) #<Object: f29008, prototype = SCATTERPLOT-PROTO>)
> (sm-span E 0.434613)
3.74129
> (sm-span C 8.39571)
4.79755
```

The selected bandwidth for `E` is equal to 0.0205786, which corresponds to a span of approximately 0.22. The same function is then applied to `C`, but the criterion does not converge even after re-adjusting the seach grid. The problem is due to the fact that `NOx` is almost constant as `C` varies, so the criterion tends to select the largest bandwidth value available in the search grid. This difficulty also emerges if we use both predictors: the bandwidths selected are extremely large. Adopting an heuristic approach for bandwidth selection, we computed the values corresponding approximately to a span of 0.5, and then we used them for smoothing data in two dimensions.

```
> (sm-bandwidth E 0.5)
0.0580833
> (sm-bandwidth C 0.5)
0.875
> (sm-regression (list E C) NOx :h (list 0.058 0.875)
                                 :xlab "E" :zlab "C" :ylab "NOx")
> (def sm (sm-regression (list E C) NOx :h (list 0.058 0.875)
                                        :display "contour" :levels '(3 2 1 0)
                                        :xlab "E" :zlab "C" :ylab "NOx"))
> (send sm :summary)
Nonparametric regression
Local linear estimator        Kernel function: Normal(0,h)
Response = NOx                                 h    = (0.058 0.875)
Terms    = E, C                                span = (0.499283 0.5)

Sigma:                0.219
Number of cases:         88
Fitted df:           28.788
Residual df:         59.212
RSS:                  2.837
```

The first fitting produces the perspective plot shown in the left panel of Figure 6, while the latter gives the contour plot in the right panel of Figure 6, where levels are computed at selected value of `NOx` estimated. From such graphs the underlying relationship between the response and the two predictors can be easily assessed. ■

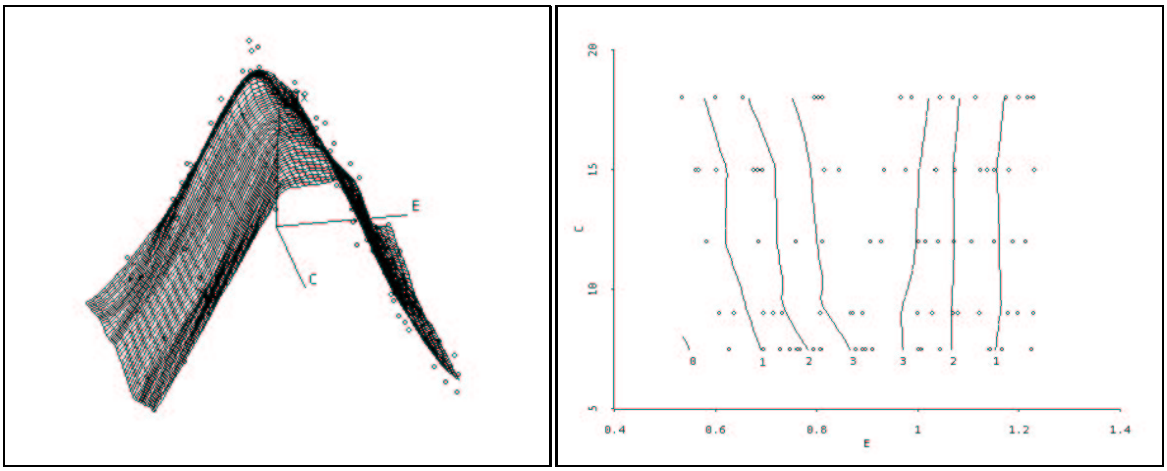

Figure 6: Nonparametric regression analysis of the `ethanol.lsp` dataset. The left panel shows the estimated smooth function through a perspective plot, whereas the right panel through a contour plot.

3.2 Nonparametric analysis of covariance

A nonparametric analysis of covariance model (see Bowman and Azzalini, 1997, Sections 6.4, 6.5) may be written as

$$y_{ij} = m_i(x_{ij}) + \epsilon_{ij}$$

where i is used to identify groups ($i = 1, \ldots, g$), and j indexes the n_i observations within each group. Furthermore, each group has its own smoothing function $m_i(.)$. This general nonparametric analysis of covariance model may be adapted to reflect the assumption of parallel nonparametric regression functions, that is we may write

$$y_{ij} = \alpha_i + m(x_{ij}) + \epsilon_{ij}$$

where there is a single smooth function $m(.)$, but different intercepts α_i for each group.

The `sm` library can be used to perform a nonparametric analysis of covariance using the function

`sm-ancova`

```
Args: (x y &key group h (display "lines")
               (model "none")
               (band T)
               (test T)
               (h.alpha nil)
               weights
               (binning nil) (nbins nil)
               (add nil)
               (ngrid *sm-ngrid*)
               eval-points
               xlab ylab
               (points-on-plot T)
               color symbol type width)
```

The required arguments are the covariate `x`, the response variable `y`, the grouping variable `group`, and the smoothing parameter `h`.

The reference model can be set using the keyword argument `:model`, which can take the values `"none"`, `"equal"` and `"parallel"`. Setting `:band` to `T` a reference band for the reference model will be produced (only in the case of two groups). A formal test can also be obtained setting `:test` to `T`.

3.3 Nonparametric estimation of the autoregression function

The dependence structure of a process $\{y_t\}$ can be studied nonparametrically analyzing the conditional mean function or autoregression function. This may be written as (see Bowman and Azzalini, 1997, Section 7.3)

$$m_k(y) = \mathrm{E}(Y_t|Y_{t-k} = y) \qquad \text{for } k \geq 1$$

or

$$m_{j,k}(y,z) = \mathrm{E}(Y_t|Y_{t-j} = y, Y_{t-k} = z) \qquad \text{for } 1 \leq j \leq k$$

The function in the `sm` library for fitting this kind of models is the following:

`sm-autoregression`

```
Args: (x &key h (d 1)
              (lags (iseq 1 d))
              (se nil)
              (ngrid *sm-ngrid*)
              (add nil)
              display
              (varname "x")
              (color 'black)
              (type 'solid) )
```

As usual, the argument `x` provide the data, in this case a list of time series values, while `h` the bandwidth used for kernel smoothing.

The keyword argument `:d` provides the number of past observations used for conditioning (1, the default, or 2), whereas `:lags` needs to be a list containing the lags considered for conditioning. If `:se` is set to `T` pointwise confidence bands are computed of approximate 95% level.

3.4 Nonparametric analysis of repeated measurements data

A repeated measurements model can be formulated in a nonparametric context as follows (see Bowman and Azzalini, 1997, Section 7.4)

$$y_{it} = m(t) + \epsilon_{ij}$$

where i is used to identify individuals ($i = 1, \ldots, N$), and t indexes the time ($t = 1, \ldots, n$). The smooth function $m(.)$ is a function of time, and the errors ϵ_{ij} are correlated within each individual and independent between individuals (a further assumption often made is that of stationarity).

Such a model can be fitted using the `sm` library through the function

`sm-rm`

```
Args: (y &key Time
              (minh 0.1) (maxh 2) (hngrid 100)
              (display "lines")
              (poly-index 1)
              (add nil) (ngrid *sm-ngrid*)
              (display-rice nil) (print t)
              (varname "y")
              (color 'black) (type 'solid) (symbol 'disk))
```

This function estimates nonparametrically the mean profile for repeated measurements (i.e. longitudinal data) from a matrix `y`, with each rows associated to individuals and columns associated to observation times. The smoothing parameter is computed using the modified Rice criterion, which selects the bandwidth taking into account the estimated autocorrelation function (see Bowman and Azzalini, 1997, p. 139).

Further details are provided in the on-line help. Scripts 7.3 and 7.4 show some statistical analyses on repeated measurements data.

3.5 Nonparametric regression with autocorrelated errors

A nonparametric regression model with autocorrelated errors can be stated as follows

$$y_t = m(x_t) + \epsilon_t$$

where the $m(.)$ is a smooth function which depends on single covariate observed over time ($t = 1, \ldots, n$), and the errors ϵ_t are generated by a stationary stochastic process with mean 0 (see Bowman and Azzalini, 1997, Section 7.5).

The function in the `sm` library which fits a nonparametric regression model with autocorrelated errors is the following

```
sm-regression-autocor
```

```
Args: (y hfirst &key (x nil)
                     (minh 0.5) (maxh 10) (hngrid 50)
                     (method "direct")
                     (display "plot")
                     (add nil) (ngrid *sm-ngrid*)
                     (display-gcv nil) (display-acf nil)
                     (print T)
                     (xlab "x") (ylab "y")
                     (color 'black) (type 'solid))
```

This function estimates nonparametrically the regression function of `y` on `x` when the error terms are serially correlated. The keyword `:method` specifies the optimality criterium adopted: possible values are `"no.cor"` `"direct"` (default), and `"indirect"`.

4 Other functions in the `sm` library

The file `sm-functions.lsp` contains the definition of global variables used by the `sm` library (see Table 7), and several functions used in conjunction with methods for nonparametric density and regression estimation.

4.1 Controlling the span and the bandwidth of the smoothing parameter

The amount of smoothing applied to the data is controlled by the smoothing parameter h, which in the `sm` library corresponds to the standard deviation of a gaussian kernel function with zero mean. Often, since the *bandwidth* is expressed in absolute value, a more easily interpretable measure is computed, namely the *span*, which is the proportion of sample data providing positive weights. Since we are using a normal kernel with standard deviation h, we may reasonably assume that data points outside a range of ± 3 times the standard deviation receive virtually zero weight. Hence, there exists the following approximate relations

$$span \approx \frac{6 * h}{\max(x) - \min(x)}$$

and

$$h \approx \frac{span * (\max(x) - \min(x))}{6}$$

Table 7: Global variables for the sm library

Variable	Default value	Description
sm-band	T	If T reference bands produced by sm-density and sm-regression are plotted as a shaded area, otherwise as lines.
sm-ngrid	100	The number of points in a regular grid used to plot the kernel estimate. For two-dimensional data approximately, *sm-ngrid*/2 number of points are used along each dimension.
sm-glm-ngrid	25	The number of points in a regular grid used for fitting local likelihood models.
sm-glm-aic	NIL	If T the corrected version of AIC is used in local likelihood model (see Section 6.8)
sm-plot-size	'(600 450)	Sets the size of the graphs produced through the sm library.

Note that the above approximations turns out to be exact, except near the boundaries, when data points along the x-axis are evenly spaced. Hence, the approximate value of the span is really the proportion of the observed range of x which receives positive weights for a given evaluation point. In this respect, it provides a simple descriptive statistic of the amount of smoothing applied to the data.

Two functions are available for computing the span for a given bandwidth and viceversa.

sm-bandwidth

```
Args: (x span &key (nnbr nil))
```

This function calculates the bandwidth for the data x (a list, a list of lists or two-columns matrix) corresponding to the span provided by the argument span. If nnbr is T a list of weights based on nearest neighbour distances is calculated; this allows to use variable bandwidths. In two dimensions, the same calculations are applied to each dimension. If nnbr is NIL only the bandwidth is returned, otherwise a list with the bandwidth and a list of weights.

These weights are computed as

$$w_k(x_i) = d_k(x_i)/\bar{d}$$

where $d_k(x_i)$ is the kth nearest neighbour distance of the covariate value x_i (computed with the function (nnbr x k) for k=round(n*span)) and $\bar{d}$ is the geometric mean of the $d_k(x_i)$. Variable bandwidths can then be defined as $h_i = h\ w_k(x_i)$ for an overall bandwidth h (see Bowman and Azzalini, 1997, pp. 17-19). Weights based on nearest neighbour distances can be provided as argument to :h-weigths in both the sm-density and sm-regression functions.

sm-span

```
Args: (x h)
```

This function returns the span for the data x (a list, a list of lists or two-column matrix) corresponding to the bandwidth h.

Example 4.1 (follicle data) The follicle.lsp dataset can be used to show how variable bandwidths may be fitted with the sm library. The data concern the number of ovarian follicles counted from sectioned ovaries of women of various ages. The scatterplot of log[Count] vs Age shows that points are sparse

along the x-axis, with data denser at older ages. Using a span of 0.5 we compute a nonparametric regression curve with fixed bandwidth and one with variable bandwidth. For comparison purposes also a *lowess* estimate is computed and then added to the plot. The last command provides a legend.

```
> (load "follicle")
> (def h (sm-bandwidth Age 0.5 :nnbr t))
> h
(4.25 (3.62644 3.62644 ... 0.997271 1.08793))
> (def sm (sm-regression Age log[Count] :h (first h)
                                        :xlab "Age" :ylab "log[Count]"))
> (sm-regression Age log[Count] :h (first h) :h-weights (second h)
                                :add (send sm :graph) :type 'dashed)
> (def lw (lowess Age log[Count] :f 0.5))
> (send (send sm :graph) :add-lines (first lw) (second lw) :color 'red :width 2)
> (send (send sm :graph) :legend 5 6 '("sm fit with fixed bandwidth"
                                       "sm fit with var. bandwidths"
                                       "lowess")
                         :color '(black black red)
                         :type '(solid dashed solid)
                         :width '(1 1 2)
                         :frame T)
```

■

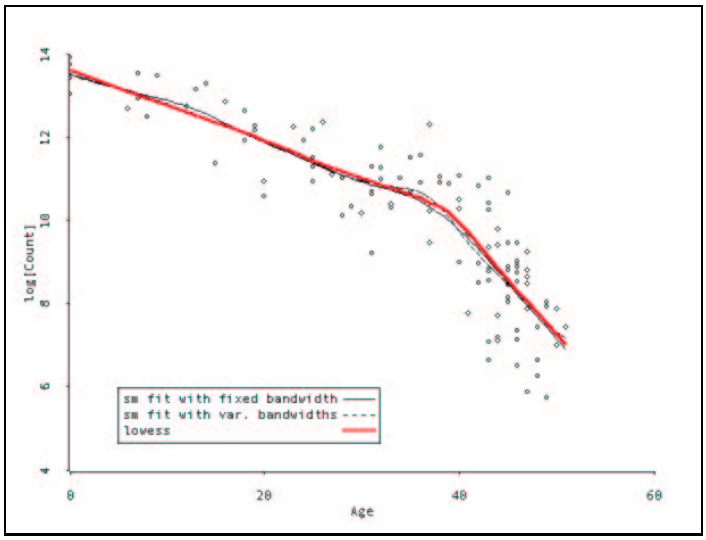

Figure 7: Nonparametric regression analysis of the `follicle.lsp` dataset. The solid curve is based on a nonparametric regression with span approximately equal to 0.5 and fixed bandwidth, the dashed line uses variable bandwidths, and the thick line is obtained with the *lowess* smoother.

4.2 Functions related to density estimation

Nearest neighbour distances from data in one or two dimensions

`nnbr`

`Args: (x k)`

This function calculates the k nearest neighbour distance from each value in x to the remainder of the data. In two dimensions, Euclidean distance is used after standardizing the data to have unit variance in each component.

Mean integrated squared error for density estimation with normal data

```
nmise
```

```
Args: (sd n h)
```

This function evaluates the mean integrated squared error of a density estimate which is constructed from data which follow a normal distribution (see Bowman and Azzalini, 1997, pp. 26-17).

Normal optimal choice of smoothing parameter in density estimation

```
hnorm
```

```
Args: (x &optional weights)
```

This function evaluates the smoothing parameter which is asymptotically optimal for estimating a density function when the underlying distribution is normal (see Bowman and Azzalini, 1997, pp. 31-32).

Cross-validatory choice of smoothing parameter

```
hcv
```

```
Args: (x &key (y nil) (h-weights nil) (ngrid 8)
              (hstart nil) (hend nil) (display "none") (add nil))
```

This function uses the technique of cross-validation to select a smoothing parameter suitable for constructing a density estimate or nonparametric regression curve in one or two dimensions (see Bowman and Azzalini, 1997, pp. 32-34, 77-79).

Cross-validation criterion for nonparametric density estimation

```
cv
```

```
Args: (x h &optional (h-weights nil))
```

This function computes a cross-validatory criterion, based on integrated squared error, for use in selecting a smoothing parameter in nonparametric density estimation.

Sheather-Jones choice of smoothing parameter for density estimation

```
hsj
```

```
Args: (x &key (step-tol 0.1))
```

This function uses the Sheather-Jones plug-in method for selecting a bandwidth which is suitable for constructing a density estimate in the one-dimensional case (see Bowman and Azzalini, 1997, pp. 34-36).

Sheather-Jones criterion for nonparametric density estimation

```
sj
```

```
Args: (x h)
```

This function computes a criterion associated with the Sheather-Jones plug-in method of selecting a bandwidth in nonparametric density estimation.

Integrated squared error between a density estimate and a normal density

```
nise
```

```
Args: (y &optional (hmult 1) (nbins (nbins y)))
```

This function evaluates the integrated squared error between a density estimate constructed from a standardized version of the univariate data `y` and a standard normal density function (see Bowman and Azzalini, 1997, pp. 38-40).

4.3 Functions related to nonparametric regression

Weighted mean and variance

```
wmean
```

```
Args (x w)
```

```
wvar
```

```
Args: (x w)
```

The two functions return, respectively, the weighted mean and the weighted variance of `x` with weights given by `w`, where `x` and `w` can be a list or a vector.

A significance trace for a hypothesis test comparing a reference parametric model with a nonparametric one

```
sig-trace
```

```
Args: (smreg hvec &key (plot T))
```

This functions creates a significance trace for a hypothesis test based on a nonparametric smoothing procedure. The argument `smreg` must be a `sm-regression` object and `hvec` a list of smoothing parameters for which the test will be performed. It :plot is `T` the p-value of the test is plotted against a range of smoothing parameters (see Bowman and Azzalini, 1997, pp. 89, 93).

Probability that a quadratic form is greater than zero

```
p-quad-moment
```

```
Args: (A Sigma &optional (tobs 0) (ndevs 0))
```

This function calculates the probability that a quadratic form $y^\top \mathbf{A} y$ is greater than zero, where $y \sim \mathrm{N}(0, \mathbf{\Sigma})$. The arguments which must be provided are the $n \times n$ matrix of coefficients for the quadratic form, and the $n \times n$ covariance matrix. Optional arguments are required if binning is used and they are, respectively, the observed statistic and the number of observations minus the number of bins.

Estimation of the smoothing matrix in nonparametric regression

```
sm-weight
```

```
Args: (x h &key (eval-points x) (hmult 1)
                (h-weights (repeat 1 (length x)))
                (poly-index 1) (cross nil)
                (weights (repeat 1 (length x))))
```

For the one-covariate case, this function computes a `length(eval-points)` by `length(x)` smoother matrix $\mathbf{S}$ such that the local polynomial smooth of `y` on `x` with bandwidth `h` and weights `h-weights` is given by $\mathbf{S}y$.

An analogue function called `sm-weight2` allows to obtain the smoothing matrix in nonparametric regressions with two covariates.

Estimation of the error standard deviation in nonparametric regression

`sm-sigma`

```
Args: (x y &key (diff-ord 2) (devs nil)
                (weights (repeat 1 (length y))))
```

This function uses ideas of local differencing to estimate the standard deviation of the errors in a nonparametric regression model with one covariate. Simple first-order differencing of pairs of neighbouring observations, or a method based on pseudo-residuals constructed from three neighbouring observations, may be used (see Bowman and Azzalini, 1997, pp. 73-75).

4.4 Binning

When the number of data points is high, the method of *binning* (see Wand and Jones, 1995, Appendix D) may be used to avoid lengthy computations and use of very large matrices. If binning is used in density estimation, the data are tabulated with the use of the function `binning`, and the outcome is passed with the frequencies computed as weights.

`binning`

```
Args: (x &key (y nil) (breaks nil) (nbins nil))
```

Given the data in `x`, this function constructs a frequency table associated to appropriate intervals covering the range of `x`. This can be a list, a vector, or a matrix with one column in the one dimensional case, whereas a list of lists, or a matrix with two columns in the two dimensional case. If you want to use binning in nonparametric regression you must provide a list or a one-column matrix of values for the response variable with `:y`.

The argument `:breaks`, either a list or a matrix with two columns, assigns the division points of the axis, or the axes in the matrix case. If `:breaks` is not given, it is computed by dividing the range of `x` into `:nbins` intervals for each of the axes. The argument `:nbins` gives the number of intervals on the `x` axis (in the univariate case), or a list of two elements with the number of intervals on each axes of `x` (in the bivariate case). If `:nbins` is not given, a value is computed as `(round (+ (log (length x) 2) 1))` or using a similar expression in the bivariate case.

The function returns a list of several elements, and for details see the on-line help.

A sort of "optimal" number of bins is returned by the following function:

`nbins`

```
Args: (x &optional (k 500))
```

For the list or list of lists `x`, given the cutoff value `k`, returns the number of bins to use according to the following rule:

$$\texttt{nbins} = \begin{cases} \texttt{round}(8 * \log(\texttt{n}))/\texttt{ndim} & \text{if } \texttt{n} > \texttt{k} \\ \texttt{NIL} & \text{otherwise} \end{cases}$$

where `n` is the number of observations and `ndim` is the dimension of `x`.

5 Extra functions for nonparametric density and regression

The file `sm-addson` contains code not provided in the original `sm` library and implements methods discussed mainly by Loader (1999).

5.1 Diagnostics for density estimation

Once we have obtained a density estimate for the data at hand, we may ask whether such estimate fit the data or not. This question is not easily answered since there is no natural definition of residuals as for regression problems, and no reference model for comparison, such as a saturated or null model. Loader (1999, pp. 87-90) suggests two graphical checks:

1. compare the integral of the density estimate

$$\hat{F}(x) = \int_{-\infty}^{x} \hat{f}(u)du$$

   with the empirical distribution function

$$\hat{F}_{emp}(x) = \frac{1}{n}\sum_{i=1}^{n} I(x_i \leq x)$$

   Thus, plot $\hat{F}(x)$ vs $\hat{F}_{emp}(x)$ and use the latter (which is the MLE for the true cumulative distribution function) as a reference for comparing the distribution function based on kernel estimation.

2. use PP-plot and QQ-plot for $\hat{F}(x)$ and $\hat{F}_{emp}(x)$ respectively. The probability (PP) plot exploits the result that any cdf is distributed uniformly on the range $[0, 1]$. Hence, if $X_{(i)}$ is the ith order statistic, then $\mathrm{E}(F(X_{(i)})) = i/(n+1)$, and a plot of $\hat{F}(X_{(i)})$ vs $i/(n+1)$ should be close to a straight line with zero intercept and unit slope. The quantile (QQ) plot transforms back to the observation scale, plotting $X_{(i)}$ vs $\hat{F}^{-1}(i/(n+1))$. Also in this case, any departure from a straight line indicates lack of fit.

The library allows, only for the one-dimensional case, to obtain a plot of the estimated cdf vs the emprical distribution function, and the PP-plot for the estimated cdf.

Example 5.1 (aircraft span data) Recalling the `aircraft.lsp` dataset, we may get a density estimate for the log span on the third period and these dignostic plots as follows:

```
> (load "sm-aircraft")
> (def y3 (log (select span (which (= period 3)))))
> (def sm (sm-density y3 :xlab "Log Span"))
> (send sm :diagnostic-plots)
```

The method `:diagnostic-plots` produces the plots shown in Figure 8. The estimated CDF is computed by the `:est-cdf` method. ■

5.2 Smoothing matrix, approximate df, residuals and diagnostic plots for nonparametric regression

A method called `:smooth-matrix` has been added to the `sm-regression-proto` in order to obtain the *smoothing matrix*. This method returns a $(n \times n)$ smoother matrix $\mathbf{S}$ evaluated at the observed points x_i $(i = 1, \ldots, n)$ for the current nonparametric regression model.

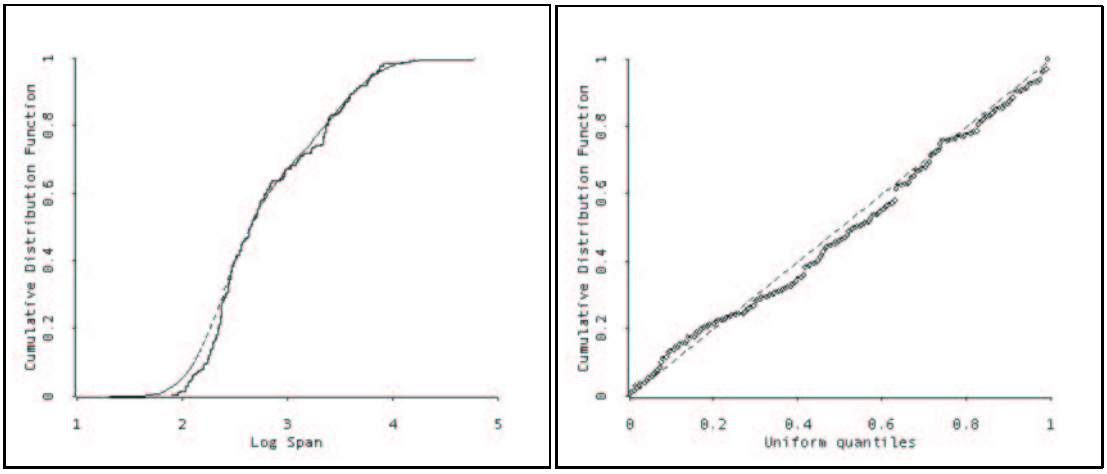

Figure 8: Diagnostic plots for nonparametric density estimation.

Fitted values for the current `sm-regression` model can be retrieved with the message `:fit-values`. These are defined as $\hat{\mathbf{m}} = \mathbf{S}\mathbf{y}$, where $\mathbf{S}$ is the smoothing matrix evaluated at the observed points and $\mathbf{y}$ is the vector of observed values for the response variable.

Residuals, defined as $e_i = y_i - \hat{m}_i$, can be retrieved sending the message `:residuals`. A plot of the residuals vs the explanatory variable (one-predictor case) or the fitted values (two-predictors case) is produced by the method `:plot-residuals`.

The approximate "number of parameters" used by the smoothing procedure is given by the *degrees of freedom* of a local fit. In analogy with the definitions used in parametric linear models (see Hastie and Tibshirani, 1990, Sec. 3.5), two possible definitions are:

$$\nu_1 = \mathrm{tr}(\mathbf{S})$$
$$\nu_2 = \mathrm{tr}(2\mathbf{S} - \mathbf{S}^\top\mathbf{S})$$

where $\mathbf{S}$ is the smoothing matrix evaluated at the observed values. The method `:fitted-df` returns the approximate degrees of freedom for the current fit. The computing formula is controlled by the optional parameter `formula`, which may take the value `1` (default) or `2`, for ν_1 and ν_2 respectively.

In the context of model comparison, approximate degrees of freedom for the error may be defined as

$$df_1 = \mathrm{tr}(\mathbf{I} - \mathbf{S}) = n - \mathrm{tr}(\mathbf{S}) = n - \nu_1$$
$$df_2 = \mathrm{tr}[(\mathbf{I} - \mathbf{S})^\top(\mathbf{I} - \mathbf{S})] = n - \mathrm{tr}(2\mathbf{S} - \mathbf{S}^\top\mathbf{S}) = n - \nu_2$$

where $\mathbf{I}$ is an identity matrix of suitable dimensions. To avoid an expensive calculation of df_2, Hastie and Tibshirani (1990, Appendix B) suggest to use the following approximation

$$2\mathrm{tr}(\mathbf{S}) - \mathrm{tr}(\mathbf{S}^\top\mathbf{S}) \approx 1.25\mathrm{tr}(\mathbf{S}) - 0.5$$

The method `:df` returns the approximate degrees of freedom for the error. It also takes the optional argument `formula` for controlling the computational formula: 1 for df_1, 2 for df_2 and 3 for the approximated version of df_2. It may be worthwhile to note that the above definitions of degrees of freedom coincide when $\mathbf{S}$ is a projection matrix.

For any evaluation point x, the estimated value computed with a nonparametric regression model can be written as

$$\hat{m}(x) = \sum_{i=1}^{n} s(x_i - x)y_i$$

where $\mathbf{s}(x) = \{s(x_i - x)\}_{i=1}^n$ is the row of the smoothing matrix $\mathbf{S}$ corresponding to the evaluation point x. This shows that nonparametric regression estimators are a linear function of the response variable. Thus, it is easy to see that

$$\mathrm{E}[\hat{m}(x)] = \sum_{i=1}^{n} s(x_i - x) m(x_i) = \mathbf{s}(x)^\top \mathbf{m}(x)$$

and

$$\mathrm{Var}[\hat{m}(x)] = \sigma^2 \sum_{i=1}^{n} s(x_i - x)^2 = \sigma^2 ||\mathbf{s}(x)||^2$$

These results form the basis for some diagnostic procedures (see Loader, 1999, pp. 27–29). Assume the evaluation points are the actual observed values of the explanatory variables, so $\mathbf{S}$ is a $(n \times n)$ smoothing matrix. The diagonal elements of $\mathbf{S}$, denoted as $s_i(x_i)$, may be used as a measure of the sensitivity of the estimated curve to the individual data points. Another useful quantity is the variance reducing factor $||\mathbf{s}(x)||^2$ which measures the reduction in variance due to the local regression. It can be shown that

$$\frac{1}{n} \leq ||\mathbf{s}(x)||^2 \leq s_i(x_i) \leq 1$$

The two extreme cases correspond, respectively, to a smoothed curve which is equal to the sample average of the response variable, and a curve which interpolates exactly the observed points.

Diagnostic plots can be obtained through the method `:diagnostic-plots`. As a result a graph with the following functions is drawn:

• **Influence function**: a plot of diagonal elements of $\mathbf{S}$ vs the explanatory variable (one-predictor case) or the fitted values (two-predictors case);

• **Variance function**: a plot of variance reducing factors vs the explanatory variable (one-predictor case) or the fitted values (two-predictors case).

Finally, a printed *summary* for the fitted nonparametric model is provided when the evaluation points are set equal to the observed points. In any case, this summary may be obtained sending the message `:summary` to a `sm-regression` object.

5.3 Generalized cross-validatory choice of smoothing parameter for nonparametric regression

The generalized cross-validation criterion can be used to select a smoothing parameter for a nonparametric regression curve in one or two dimensions. The GCV statistic is an estimate of the predictive mean squared error (PMSE), and it is given by

$$\mathrm{GCV}(h) = n \, \frac{\sum_{i=1}^{n} (y_i - \hat{m}(x_i))^2}{n - \nu_1}$$

where ν_1 is equal to the trace of the smoothing matrix, so $(n - \nu_1)$ is an estimate of the degrees of freedom already discussed.

In the `sm` library the GCV criterion can be invoked using the following function:

`hgcv`

```
Args: (x y &key (h-weights nil) (ngrid 10) (hstart nil) (hend nil)
                (nnbr nil) (display "none") (add nil)
                (color 'black) (type 'solid) )
```

The `x` argument provides the explanatory variable (a list, a vector, or a single column matrix) in the one-dimensional case, or the explanatory variables (a list of lists or a two-column matrix) in the two-dimensional

case. `y` is a list of response values. If the argument `:nnbr` is set to `T` a nearest neighbour bandwidth is used. The argument `:display` controls the graphical output: any character setting other than `"none"` will cause the criterion function to be plotted over the search grid of smoothing parameters. The particular value `"log"` will use a log scale for the grid values, whereas `"df"` will use the fitted degrees of freedom on the x-axis. By default fitted df are computed using $\nu 1$ in Section 5.2, whereas ν_2 is used if `:display` is set to `"df2"`. The remaining arguments were already discussed in the `hcv` function (see Section 4.2).

5.4 Mallows' CP criterion for selecting the smoothing parameter for nonparametric regression

Mallows' CP criterion can be used to select a smoothing parameter for nonparametric regression in one or two dimensions. The CP criterion is an estimate of the squared estimation error, and is given by

$$\mathrm{CP}(h) = \frac{1}{\sigma^2}\sum_{i=1}^{n}(y_i - \hat{m}(x_i))^2 - n + 2\nu_1$$

An estimate of σ^2 is required. Cleveland and Devlin (1988) suggested to use the normalized residual sum of squares evaluated at the smallest bandwidth considered (i.e. where the bias is the smallest):

$$\hat{\sigma}^2 = \frac{1}{n - 2\nu_1 + \nu_2}\sum_{i=1}^{n}(y_i - \hat{m}(x_i))^2$$

The CP criterion can be applied using the following function:

`hcp`

```
Args: (x y &key (h-weights nil) (ngrid 10) (hstart nil) (hend nil)
                (nnbr nil) (display "none") (add nil)
                (color 'black) (type 'solid) )
```

All the arguments have the same meaning already discussed in the previous section for the `hgcv` function.

Example 5.2 (motorcycle data) Considers the `mcycle.lsp` dataset (Silverman, 1985), which records data from an experiment to test crash helmets. A series of measurements of head acceleration after a simulated motorcycle crash were recorded (`accel`), together with the time in milliseconds since impact (`times`). Since observations are sparse for large values of the covariate, we might think to use a variable bandwidths based on nearest neighbour distances.

```
> (load "mcycle.lsp")
> (def hgcv (hgcv times accel :nnbr T :ngrid 20 :display T))
(1.53183 #<Object: ca4160, prototype = SCATTERPLOT-PROTO>)
> (def hcp (hcp times accel :nnbr T :ngrid 20 :display T))
(1.57266 #<Object: c9dfa0, prototype = SCATTERPLOT-PROTO>)
> (sm-span times 1.53)
0.166304
```

The smoothing parameter selected by the GCV criterion and the CP criterion are almost the same (see also Figure 9), and corresponds to a span of about 0.166. Since we are using variable bandwidths, we first need to get the weights for any observation at the selected span. Then, we fit a local linear regression model and we produce some diagnostic plots.

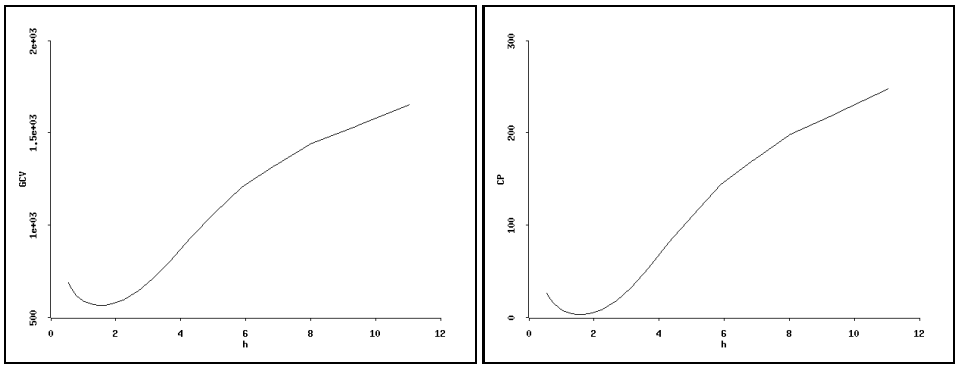

Figure 9: Generalized cross-validation plot (left panel) and CP plot (right panel) for bandwidth selection in the `mcycle.lsp` dataset.

```
> (def h (sm-bandwidth times (sm-span times 1.53) :nnbr T))
(1.53 (3.29206 3.2381 ... 4.91111))
> (def smreg (sm-regression times accel :h (first h) :h-weights (second h)
                                        :eval-points times
                                        :xlab "times" :ylab "accel"))
Nonparametric regression
Local linear estimator        Kernel function: Normal(0,h)
Variable bandwidths
Response = accel                               h    =      1.530
Term     = times                               span =      0.166

Sigma:              22.825
Number of cases:       133
Fitted df:           13.856
Residual df:        119.144
RSS:              60600.097

> (send smreg :plot-residuals)
> (send smreg :diagnostic-plots)
```

The top panel of Figure 10 shows both the smooth regression function obtained using variable bandwidths (solid line) and the smooth function obtained with fixed bandwidth as follows:

```
> (sm-regression times accel :h (first h)
                             :eval-points times
                             :add (send smreg :graph) :type 'dashed
                             :xlab "times" :ylab "accel")

Nonparametric regression
Local linear estimator        Kernel function: Normal(0,h)
Response = accel                               h    =      1.530
Term     = times                               span =      0.166

Sigma:              22.825
Number of cases:       133
Fitted df:           16.072
Residual df:        116.928
RSS:              61660.119
```

■

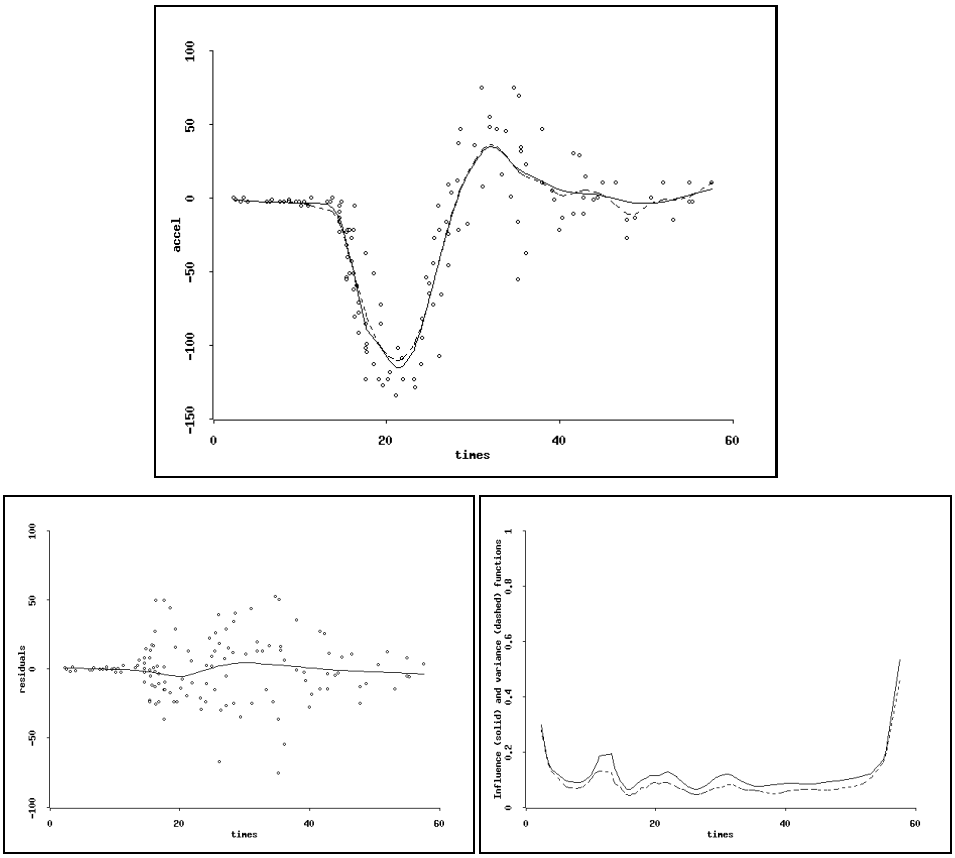

Figure 10: Nonparametric regression model for the `mcycle.lsp` dataset. The top panel shows the plot of the response variable vs the explanatory variable together with the estimated smooth functions, solid for variable bandwidths and dashed for fixed bandwidth. The bottom-left panel shows the residuals plot, while the bottom-right panel shows the influence and variance functions.

6 Local likelihood estimation for generalized linear models

Local likelihood fitting was proposed by Tibshirani (1984) and Hastie and Tibshirani (1987). The basic idea is that whenever a likelihood function is available we can maximize it locally in order to obtain an estimate of a response variable at some point, say x_0. Data points in a neighbourhood of x_0 are weighted according to a smooth function which gives weights that decrease as x_i increases in distance from x_0. A technical discussion of the local likelihood approach is given by Fan et al. (1995), while a full account of the topic is provided by Loader (1999).

This approach allows to fit a smooth curve on a scatterplot of data, taking directly into account the nature of the response variable. In the case of a binomial response, and even more for a bernoulli variable, smoothers developed for a continuous variable are not satisfactory because they do not use the information available. For instance, if we smooth a scatterplot of a binary vs a continuous predictor with a smoother for continuous data, a clear drawback is that we are not taking into account the binary nature of the response, so the fitted smooth curve could lie outside the range $[0, 1]$. On the contrary, using a local likelihood approach we can model directly this information and obtain a smooth curve that fit the data in a more sensible way. The same arguments apply for bounded variables, such as nonnegative variables, or discrete variables, such as those expressing counts.

The `sm` library provided by Bowman and Azzalini (1997) allows to smooth binomial and Poisson data. The corresponding *Xlisp-Stat* functions are described in Section 6.10. A more general approach suitable for any GLM family should be included in the further release of the `sm` library for *S-Plus*. In the current release of the `sm` library for *Xlisp-Stat*, we developped several functions for applying the local likelihood approach to any GLM, and these are described in the following sections.

6.1 A review of the theory of local likelihood for GLMs

Assume that a response variable Y has a distribution in the exponential family, taking the form

$$f_Y(y; \theta, \phi) = \exp\{[y\theta - b(\theta)]/a(\phi) + c(y, \phi)\}$$

where θ, the canonical parameter, and ϕ, the dispersion parameter, are scalar values. $a()$, $b()$, $c()$ are known functions, which determine the specific parametric family of distributions (see McCullagh and Nelder, 1989, Table 2.1, p. 30). Usually, θ is the parameter of interest, whereas ϕ is regarded as a nuissance parameter.

The log-likelihood for a set of n i.i.d. observations is given by

$$l(\boldsymbol{\mu}; \mathbf{y}) = \sum_{1=1}^{n} \log f_Y(y_i; \theta_i)$$

where $\mathrm{E}(\mathbf{Y}) = \boldsymbol{\mu}$ whose ith component is $\mu_i = b'(\theta_i)$. Assuming we have n observations on a set of p covariates $\mathbf{x}_1, \ldots, \mathbf{x}_p$, the systematic component of a GLM is given by $\boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta}$, where $\boldsymbol{\eta}$ is called the linear predictor and its ith component is $\eta_i = \boldsymbol{\beta}^\top \mathbf{x}_i$.

Finally, the link function connects the random and systematic components, that is

$$\eta_i = g(\mu_i)$$

so the mean function is given by $\mathrm{E}(y_i|\mathbf{x}_i) = g^{-1}(\eta_i) = g^{-1}(\boldsymbol{\beta}^\top \mathbf{x}_i)$.

Maximum likelihood estimates for $\boldsymbol{\beta}$ are obtained by setting the first derivative of the log-likelihood with respect to each parameter equal to zero and then solving the system of equations. Since this is usually a set of nonlinear equations, estimates are obtained numerically through the Fisher scoring algorithm or the Newton-Raphson method (Nelder and Wedderburn, 1972).

6.2 Local likelihood function

Suppose we have a single predictor, and we denote the log-likelihood as $l(\boldsymbol{\beta}; \mathbf{y})$. The local likelihood approach essentially applies a weighting mechanism to the log-likelihood, so we can write

$$l_{h,x}(\boldsymbol{\beta}; \mathbf{y}) = \sum_i l_i(\boldsymbol{\beta}; y_i) w(x_i - x; h) \tag{1}$$

where $w(x_i - x; h)$ is a weighting function that determines the relative importance of each contribution $l_i(\boldsymbol{\beta}; y_i)$ to the log-likelihood.

Generally, $w(z; h)$ is a smoothing function which peaks at 0 and decreases monotonically as z increases in size. This puts more weight on the observations x_is close to x, and decreasing weights for x_is that are distant from x. The smoothing parameter h controls the bandwidth of the smoothing function, that is the width of the nearest neighbourhood of x. Only the observations that lie in this neighbourhood will contribute to the log-likelihood. The proportion of points that lie in it is called the span. Thus, the smoothing parameter controls the degree of smoothing applied to the data.

The local likelihood in equation (1) for any point of evaluation x, and for a given smoothing parameter h, is equal to the sum of the contribution $l_i(\boldsymbol{\beta}; y_i)$ from each observed value x_i, weighted by a smoothing function of the distance of x_i from x.

For any x we can maximize the local likelihood (1) with respect to $\boldsymbol{\beta}$, which provides the local estimate $\hat{\boldsymbol{\beta}}$ (in a more formal notation we should indicate the coefficient estimates at x with $\hat{\boldsymbol{\beta}}_x$, but we avoid this for simplicity of notation). The fitted value at x can then be computed as

$$\hat{\eta}_x = \hat{\boldsymbol{\beta}}^\top \mathbf{x}$$

where $\mathbf{x} = [1, x]^\top$.

Applying the inverse link function transformation, we can express the fitted value in the response variable scale as

$$\hat{\mu}_x = g^{-1}(\hat{\boldsymbol{\beta}}^\top \mathbf{x})$$

6.3 Smoothing weights

Since the local likelihood estimates depend on the weighting function $w(x_i - x; h)$, we need to define the weights to use on each $l_i(\boldsymbol{\beta}; y_i)$.

Let $\mathbf{x}_0 = (x_{0_1}, \ldots, x_{0_k})^\top$ be the $k \times 1$ vector of points where evaluate equation (1), and $\mathbf{x}_{(i)}$ a $k \times 1$ vector of values all equal to the ith ordered observation x_i. Define a $k \times n$ matrix as

$$\mathbf{W} = \underbrace{\left[\mathbf{x}_{(1)}, \mathbf{x}_{(2)}, \ldots, \mathbf{x}_{(n)}\right]}_{k \times n} - \underbrace{\left[\mathbf{x}_0, \mathbf{x}_0, \ldots, \mathbf{x}_0\right]}_{k \times n} = \begin{bmatrix} (x_1 - x_{0_1}) & (x_2 - x_{0_1}) & \ldots & (x_n - x_{0_1}) \\ (x_1 - x_{0_2}) & (x_2 - x_{0_2}) & \ldots & (x_n - x_{0_2}) \\ \vdots & \vdots & & \vdots \\ (x_1 - x_{0_k}) & (x_2 - x_{0_k}) & \ldots & (x_n - x_{0_k}) \end{bmatrix}$$

where the generic (j, i)-th element is given by $(x_i - x_{0_j})$. Therefore, the $\mathbf{W}$ matrix has elements equal to the difference between the point of estimation and the observed value of the predictor. The smoothing weights are then obtained by applying a smoothing function, which depends on h, to $\mathbf{W}$. For instance, we can use a normal kernel density with standard deviation equal to h, so the smoothing weights are defined as

$$w(x_i - x_{0_j}; h) = \exp\left\{-\frac{1}{2}\left(\frac{x_i - x_{0_j}}{h}\right)^2\right\}$$

When $|x_i - x_{0_j}|$ is zero the ith observation has weight equal to one, and as the distance increases the weights decrease approaching zero for observations x_is far from the point of evaluation x_{0_j}. Since we are using a normal kernel, observations outside a range of $\pm 3h$ standard deviations from x_{0_j} have approximately zero weight and their contribution to the local likelihood will be nearly zero.

Once we have obtained the weights we can use the IRWLS procedure for fitting a generalized linear model using as prior weights $w(x_i - x; h)$ for $i = 1, \ldots, n$, and repeating the fitting procedure k times, one for each evaluation point $x \equiv x_{0_j}$.

6.4 Variability bands

As in all statistical models, it is useful to have a measure of the uncertainty associated with the estimated values. Usually this is done providing standard errors for the estimates and/or confidence intervals for a given confidence level. An informal approach can be pursued providing the so-called *variability bands* (see Bowman and Azzalini, 1997, pp. 75–76). These bands indicate pointwise confidence intervals for $\mathrm{E}[\hat{\eta}_x]$ rather than $\hat{\eta}_x$.

Assume that for each evaluation point $\mathbf{x}_0 = [1, x_0]^\top$ we have fitted a local regression model, and we have obtained a vector of coefficient estimates $\hat{\boldsymbol{\beta}}$. Associated with these estimates, a covariance matrix given by $(\mathbf{X}^\top \mathbf{V}_{x_0} \mathbf{X})^{-1}$ is provided, where $\mathbf{X} = [\mathbf{1}, \mathbf{x}]$ is the design matrix and $\mathbf{V}_{x_0}$ is the matrix of weights obtained at the end of the iteratively reweighted least squares procedure. For the fitted value of the linear predictor

$$\hat{\eta}_{x_0} = \hat{\boldsymbol{\beta}}^\top \mathbf{x}_0$$

the variance can be computed as

$$\mathrm{Var}\,(\hat{\eta}_{x_0}) = \phi\, \mathbf{x}_0^\top (\mathbf{X}^\top \mathbf{V}_{x_0} \mathbf{X})^{-1} \mathbf{x}_0$$

where, if required, we need to plug-in an estimate for ϕ. A variability band in the linear predictor scale can be constructed to indicate the size of two standard errors above and below $\hat{\eta}_{x_0}$. Applying the inverse link function transformation to the lower and upper limits of this band, we obtain a set of variability bands for the mean.

6.5 Approximate degrees of freedom

As we saw in Section 5.2 the approximate fitted degrees of freedom expresses the "number of parameters" used by the smoothing procedure. For generalized linear models, the number of coefficients estimates can be computed through the hat matrix defined as

$$\mathbf{H} = \mathbf{V}^{1/2} \mathbf{X} (\mathbf{X}^\top \mathbf{V} \mathbf{X})^{-1} \mathbf{X} \mathbf{V}^{1/2}$$

where $\mathbf{X}$ is the deign matrix and $\mathbf{V}$ is the matrix of known weights obtained at the end of the iterative fitting algorithm (see McCullagh and Nelder, 1989, p. 405). In analogy with the linear regression models, the trace of the hat matrix equals the number of parameters.

In a local likelihood GLM with evaluation points equal to the observed points, for each $\mathbf{x}_i = [1, x_i]^\top$ the corresponding leverage is given by

$$h_i = v_i^{1/2} \mathbf{x}_i^\top (\mathbf{X}^\top \mathbf{V}_{x_i} \mathbf{X})^{-1} \mathbf{x}_i v_i^{1/2}$$

where v_i is the ith diagonal element of $\mathbf{V}_{x_i}$ for the current fit. Summing h_i over all i provides the approximate number of fitted degrees of freedom

$$df_{\mathrm{fit}} = \sum_{i=1}^{n} h_i$$

Residual degrees of freedom are easily obtained as

$$df_{\text{error}} = n - df_{\text{fit}}$$

6.6 Goodness of fit measures

A general measure of goodness of fit used in GLM is the *deviance*. This is defined as twice the difference between the log-likelihood of a saturated model (which has as many parameters as the number of observations) and the log-likelihood for the current model. The exact form depends on the exponential family distribution under study, and it can be expressed as the sum of n components

$$D(\mathbf{y}, \hat{\boldsymbol{\mu}}) = \sum_{i=1}^{n} D(y_i, \hat{\mu}_i)$$

where $D(y_i, \hat{\mu}_i)$ is the contribution from the ith observation. This definition easily extends to local likelihood GLM, where in this case $\hat{\mu}_i = g^{-1}(\hat{\eta}_{(x_i)})$, the mean evaluated at the ith observation.

Another measure of discrepancy is the *Pearson* X^2 statistic

$$X^2 = \sum_{i=1}^{n} \frac{(y_i - \hat{\mu}_i)^2}{\text{Var}(\hat{\mu}_i)}$$

which extends to local likelihood models in a way similar to the deviance.

6.7 Residuals

Residuals for local likelihood are defined similarly to residuals for GLM based on the full likelihood. Some possible choices are:

- raw residuals
$$r_i = y_i - \hat{\mu}_i$$
- deviance residuals
$$r_{D,i} = \text{sign}(y_i - \hat{\mu}_i) D(y_i, \hat{\mu}_i)^{1/2}$$
- Pearson residuals
$$r_{P,i} = \frac{y_i - \hat{\mu}_i}{\text{Var}(\hat{\mu}_i)^{1/2}}$$

where $\hat{\mu}_i = g^{-1}(\hat{\eta}_{x_i})$ and $\text{Var}(\hat{\mu}_i)$ is the variance function evaluated at the ith observation.

6.8 Choosing the smoothing parameter

As for linear nonparametric regression models, the choice of the smoothing parameter is a crucial step in local likelihood models. An extension of the Akaike Information Criterion (AIC) can be used to guide this selection.

The AIC, which is an estimate of the expected Kullback-Leibler information (Burnham and Anderson, 1998), is defined as

$$\text{AIC} = -2 \log \mathcal{L}(\hat{\theta}|y) + 2k$$

where $\log \mathcal{L}(\hat{\theta}|y)$ is the likelihood for the unknown parameter θ evaluated at the estimate $\hat{\theta}$ based on data y, and k is the number of estimated parameters which acts as a penalizing term. Recalling the definition of the deviance, the AIC is proportional to the deviance from the fitted model plus a penalty term:

$$\text{AIC} \propto D + 2k$$

The AIC criterion applied to a local likelihood model selects the smoothing parameter h for which the following statistic is minimized:

$$\mathrm{AIC}\,(h) = \sum_{i=1}^{n} D(y_i, \hat{\mu}_i) + 2k$$

where $k = df_{\mathrm{fit}}$. The AIC criterion is valid asymptotically, so refinements have been proposed for small to moderate sample sizes. In particular, when the ratio n/k is small (less than 40 it is generally recommended) the penalizing term should be equal to $2k(n/(n-k-1))$, so the corrected AIC criterion can be computed as

$$\mathrm{AIC}_c = \mathrm{AIC} + \frac{2k(k+1)}{n-k-1}$$

6.9 The `sm-glm` function

The function `sm-glm` allows to estimate generalized linear models using the local likelihood approach.

`sm-glm`

```
Args: (x y &key (trials nil) (offset nil) (h nil) (family nil) (link nil)
                (ngrid *sm-glm-ngrid*) (eval-points nil)
                (display "estimate") (band nil)
                (add nil) (verbose T)
                (xlab "x") (ylab "y") (points-on-plot T)
                (color 'black) (symbol 'disk)
                (type 'solid) (width 1))
```

The required arguments are the covariate `x` (list or vector of values), the response variable `y` (a list or vector of values), the smoothing parameter `h`, and the family.

The `:family` argument must be a string specifying the distribution for the response variable. Families supported are: `"normal"`, `"binomial"`, `"poisson"`, and `"gamma"`. Strictly connected is the `:link` argument, which must be a *Xlisp-Stat* link object. By default, canonical link functions are used:

Family	Canonical link
`"normal"`	`identity-link`
`"binomial"`	`logit-link`
`"poisson"`	`log-link`
`"gamma"`	`inverse-link`

Other link functions are available (as `probit-link`, `cloglog-link`, `power-link`, `sqrt-link`) or may be created by the user as described by Tierney (1991).

The argument `:trials` is used if the family is `"binomial"`, and by default is assumed to contain all 1's. The argument `:offset` can be used to provide a list or a vector of offset values.

The graphical output is controlled by the keyword argument `:display`. The setting `"none"` or `NIL` will prevent any graphical output from being produced. The default setting `"estimate"` or `T` will produce the estimated mean function to be plotted vs the covariate, while the setting `"se"` will in addition produce a variability band. The latter may also be obtained setting the argument `:band` to T.

The smooth mean function is evaluated by default on a grid of equally spaced points with number of points given by `:ngrid`. However, these may be specified through the argument `eval-points` (a list or a vector of values). If the evaluation points are set equal to the observed points a summary of the local likelihood fitting is printed.

The function `sm-glm` returns a `sm-glm-proto` object as its result. Then, the user may assign the returned object to a variable and send messages to it.

The slot values can be retrieved sending the messages summarized on Table 8, and the results from the local likelihood fitting are stored in the slots reported on Table 9. As we saw for density and regression smoothing, slot values can be modified and the needed computations invoked sending the message `:compute`, while a new plot may be obtained through the `:plot` method (see Table 6.9).

Table 8: Methods for accessing slot values in a `sm-glm-proto` object

Method	Description
`:x`	data for the covariate
`:y`	data on the response variable
`:h`	value of the smoothing parameter
`:eval-points`	points where the local likelihood model have been fitted
`:ngrid`	number of points used in a regular grid for estimation
`:band`	this slot is set to `T` if a variability band must be plotted
`:display`	controls the graphical output
`:add`	object address which contains the graphical output if required, `NIL` otherwise

The AIC criterion for choosing the smoothing parameter in local likelihood model can be invoked through the following function:

`haic-loclik`

```
Args: (x y &key (trials nil) (family nil) (link nil)
                (ngrid 10) (hstart nil) (hend nil)
                (display ″none″) (add nil) (verbose T)
                (color 'black) (type 'solid) )
```

where the arguments have the same meaning as in the `hcv`, `hgcv`, and `hcp` functions for bandwidth selection already discussed. Furthermore, arguments required by the `sm-glm` function must be provided as well.

6.10 Local likelihood for logistic and Poisson regression and PRLT test

In the `sm` library for *S-Plus* two functions are available for fitting binomial and Poisson local likelihood models with canonical link function. These functions are also available in the *Xlisp-Stat* version of the library. Actually, they merely provide a quick way of calling the `sm-glm` function with the required arguments but without specifying the family and link function to use.

`sm-logit`

```
Args: (x y &key (trials (repeat 1 (length y)))
                (h nil)
                (ngrid *sm-glm-ngrid*)
                (eval-points nil)
                (display "estimate")
                (verbose t) (add nil)
                (xlab "x") (ylab "y")
                (points-on-plot T)
                (color 'black) (symbol 'disk)
                (type 'solid) (width 1))
```

This function estimates the logistic regression curve using the local likelihood approach for the observations on a binomial response and an associated set of covariate values. The arguments have the same meaning discussed for the `sm-glm` function (see Section 6.9).

Table 9: Methods for accessing results from a `sm-glm-proto` object

Method	Description
`:estimate`	mean function estimates at each evaluation point (for binomial models this provides estimated probability, while if an offset have been provided it is not included)
`:se`	standard errors at each evaluation point
`:leverages`	leverages at each evaluation points
`:scale`	scale parameter estimated from the corresponding global GLM model
`:glm-model`	the glim-proto object for the fitted model based on the full likelihood
`:span`	approximate span for the current fit
`:mu`	list of estimated mean values †
`:variances`	list of variance values †
`:deviances`	list of deviance values †
`:deviance`	deviance †
`:pearson-x2`	Pearson X^2 †
`:raw-residuals`	list of raw residuals †
`:residuals`	list of Pearson residuals †
`:deviance-residuals`	list of deviance residuals †
`:chi-residuals`	list of Pearson X^2 residuals †
`:fitted-df`	approximate fitted degrees of freedom †
`:df`	approximate degrees of freedom for the error †
`:aic`	Akaike Information Criterion †
`:sm-weights`	matrix of smoothing weights for the current evaluation points
`:variability-band`	a list of upper and lower limits of a variability band
`:upper`	variability band upper limits at each evaluation point, computed as the regression estimates plus 2 times the standard errors
`:lower`	variability band lower limits at each evaluation point, computed as the regression estimate minus 2 times the standard errors
`:graph`	address of the last graph drawn from the object

† The method applies only if the evaluation points are equal to the observed points.

`sm-poisson`

```
Args: (x y &key (h nil)
                (offset nil)
                (eval-points nil)
                (ngrid *sm-glm-ngrid*)
                (display "estimate")
                (verbose T) (add nil)
                (xlab "x") (ylab "y")
                (points-on-plot T)
                (color 'black) (symbol 'disk)
                (type 'solid) (width 1))
```

This function estimates the regression curve using the local likelihood approach for a vector of Poisson observations and an associated vector of covariate values. Also in this case, the arguments have the same meaning discussed for the `sm-glm` function (see Section 6.9).

Table 10: Methods for computing and graphing nonparametric regression estimates

Method	Description
`:compute`	starts computing local likelihood estimates based on the information stored in the slots of the current object
`:plot`	draws a plot of the estimated mean function vs the covariate
`:plot-residuals`	plots the deviance and Pearson residuals vs the covariate †
`:summary`	Prints a summary of the fitting for the local likelihood model †

† The method applies only if the evaluation points are equal to the observed points.

A comparison between a parametric GLM and a nonparametric model may be performed through the so-called *Pseudo-Likelihood Ratio Test* (see Azzalini et al. 1989, Azzalini and Bowman, 1997, pp. 98–104). This bootstrap goodness-of-fit test allows to test a parametric model vs a nonparametric model. The functions for applying such tests are:

`sm-logit-bootstrap`

```
Args: (x y &key (trials (repeat 1 (length y)))
                (h nil)    (nboot 100)
                (degree 1) (phi 1)
                (plot nil) (xlab nil) (ylab nil))
```

This function performs a Pseudo-Likelihood Ratio Test for the goodness-of-fit of a standard parametric logistic regression. The argument `nboot` gives the number of bootstrap samples (by default is set equal to 100). The parametric model is fitted using a polynomial in `x` with degree given by `:degree` (default 1). A dispersion parameter can be provided through the argument `:phi`. The function returns a list of lists containing:

- the plot object representing the bootstrap samples;
- the observed value of the Pseudo-Likelihood Ratio Test statistic;
- the observed p-value estimated via the bootstrap method.

A similar function is also provided for a Poisson regression model:

`sm-poisson-bootstrap`

```
Args: (x y &key (offset nil) (h nil)
                (nboot 100) (degree 1) (scaled-dev nil)
                (plot nil) (xlab nil) (ylab nil))
```

The arguments take the same meaning as in the previous function, except `:scaled-dev` which if set to `T` allows the differences in deviance to be scaled by an estimate of the dispersion parameter. The function returns a list as in the previous case.

Example 6.1 (mortality rate) The dataset from Henderson and Sheppard (1919) in the file `mortable.lsp` reports the observed number of deaths (`D`) for ages from 55 to 99 (`Age`), and the corresponding total number of patients in each class (`N`) . We could think to model this data as they come from a binomial distribution where the response variable is the number of death people and the trials are the number of subjects in the class.

Suppose we want to estimate a function that smooths the data and gives a summary of the mortality rates. To apply a local likelihood fitting, we must choose the smoothing parameter. The AIC criterion applied to this dataset gives the following results:

```
> (load "mortable")
> (def h (haic-loclik Age D :trials N :family "binomial" :display T :ngrid 20))
AIC criterion for local likelihood models
-----------------------------------------
          h            AIC      fitted df
    2.20000       53.80545        9.34915
    2.41756       53.59445        8.69225
    2.65664       53.50962        8.08704
    2.91937       53.55760        7.53128
    3.20807       53.71496        7.00712
    3.52533       53.97245        6.50615
    3.87396       54.29760        6.01681
    4.25706       54.65898        5.54063
    4.67805       55.01668        5.08565
    5.14068       55.33454        4.66456
    5.64906       55.58525        4.28740
    6.20771       55.75592        3.95722
    6.82161       55.85155        3.67092
    7.49621       55.89387        3.42291
    8.23754       55.91216        3.20760
    9.05217       55.93459        3.02082
    9.94736       55.98114        2.85936
   10.93109       56.06173        2.72068
   12.01209       56.17750        2.60241
   13.20000       56.32337        2.50225

> h
(2.69162 #<Object: 56c798, prototype = SCATTERPLOT-PROTO>)
> (sm-span Age 2.69)
0.366818
```

The bandwidth selected is 2.69162, which corresponds to a span of approximately 0.37. Then, the local likelihood GLM can be fitted as follows:

```
> (def loclik (sm-glm Age D :trials N :family "binomial" :h 2.69 :band T
                            :xlab "Age" :ylab "Pr(Death)" :eval-points Age))
Computing local likelihoods  . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . .
Local likelihood model
Family   = Binomial           Kernel function: Normal(0,h)
Link     = logit-link                         h    =      2.690
Response = Pr(Death)                          span =      0.367
Term     = Age

Scale:                1.000
Number of cases:         45
Fitted df:            8.011
Residual df:         36.989
Deviance:            37.487
AIC:                 57.520
```

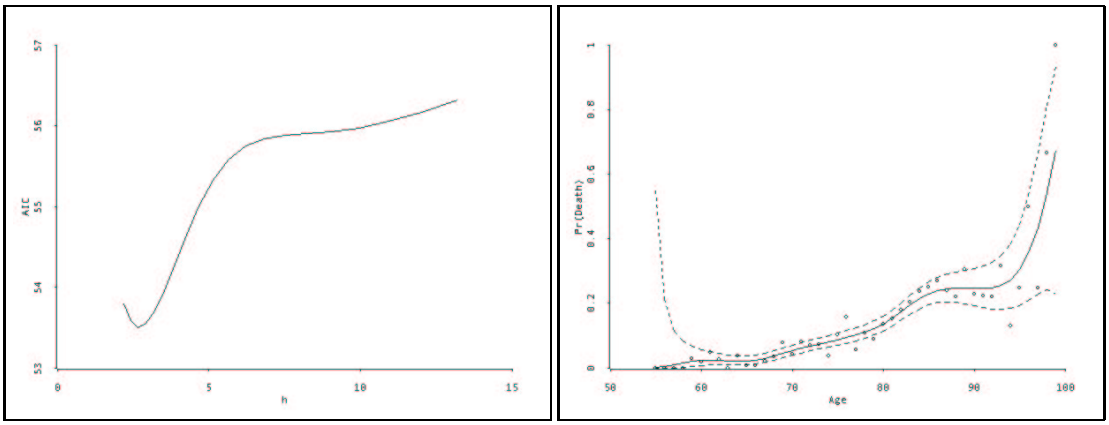

Figure 11: AIC plot for bandwidth selection (left panel) and the fitted smooth curve obtained by local logistic model with variability bands for the regression of death rates vs `Age` with $h = 2.690$.

Figure 11 shows the smooth curve obtained applying the local likelihood estimation with smoothing parameter $h = 2.69$. We also plot the variability bands for the estimated smooth curve. The death rate increases as age increases up to age 88 approximately, then becomes constant and rapidly increases after 95 years. However, we must be careful in the interpretation of this latter shape, since there are few observations after age 95, and for age 99 just one case, which is a death. This uncertainty is revealed also by the width of the variability bands, which is quite large for older and younger ages.

Depending on the context we could stop here the analysis. Anyway, we proceed a little further, assuming that, based on the nonparametric local fitting, we want to seek for a suitable parametric model that captures the main features on the data. We fitted a polynomial logistic model and Table 11 shows that a linear and a quadratic term in `Age` seem to be required.

Table 11: Analysis of Deviance Table

Source	df	Deviance	p-value	AIC
Null model	44	273.487		275.580
`Age`	43	54.344	0.00000	58.630
`Age`+`Age`^{2}	42	49.927	0.03558	56.512
`Age`+`Age`^{2}+`Age`^{3}	41	49.135	0.37368	58.135
`s(Age)`	36.989	37.487		57.520

A way of testing the adequacy of a parametric model compared to a nonparametric one obtained through local likelihood may be obtained via the pseudo-LRT:

```
> (sm-logit-bootstrap Age D :trials N :h 2.69 :degree 1
                            :xlab "Age" :ylab "Pr(Death)")
Bootstrap goodness-of-fit test for a logistic regression model.
-----------------------------------
Dispersion parameter  = 1
PLRT test statistic   = 16.85730
Observed significance = 0.09000

> (sm-logit-bootstrap Age D :trials N :h 2.69 :degree 2
                            :xlab "Age" :ylab "Pr(Death)")
Bootstrap goodness-of-fit test for a logistic regression model.
-----------------------------------
```

```
Dispersion parameter  = 1
PLRT test statistic   = 12.44006
Observed significance = 0.16000
```

The observed significance seems to suggest that a second degree polynomial on `Age` should suffice, so providing a further confirmation of what discussed previously. ■

Example 6.2 (birds in paramo vegetation) A study was conducted to investigate the number of bird species in isolated islands of paramo vegetation in the northern Andes. The aim was to investigate how the number of species is related to some covariates. The dataset is contained in the file `paramo.lsp` and we focus on the relationship between the number of species and the area of the island, so a Poisson model seems to be reasonable.

Again, the AIC criterion can be used for bandwidth selection. Since the number of observations is small, we may adopt the corrected version of AIC (see Section 6.8) by setting the global variable `*SM-GLM-AIC*` to `T`. Then, we fit the Poisson local likelihood model.

```
> (load "paramo")
> (def log[AR] (log AR))
> (def *SM-GLM-AIC* T)
> (setf h (haic-loclik log[AR] N :family "poisson" :ngrid 15 :display T))
(0.424878 #<Object: fa1580, prototype = SCATTERPLOT-PROTO>)
> (def loclik (sm-glm log[AR] N :family "poisson" :h 0.424878 :band T
                                :xlab "log[AR]" :ylab "N" :eval-points log[AR]))
Computing local likelihoods  . . . . . .
Local likelihood model
Family   = Poisson           Kernel function: Normal(0,h)
Link     = log-link                           h    =      0.425
Response = N                                  span =      0.595
Term     = log[AR]

Scale:               1.000
Number of cases:        14
Fitted df:           5.378
Residual df:         8.622
Deviance:           23.594
AIC:                43.348
```

The fitted curve is shown on the right panel of Figure 12. ■

7 Nonparametric regression with survival data

The function `sm-survival` creates a smooth, nonparametric estimate of the quantile of the distribution of survival data as a function of a single covariate. A weighted Kaplan-Meier survivor function is obtained by smoothing across the covariate scale. A small amount of smoothing is then also applied across the survival time scale in order to achieve a smooth estimate of the quantile (for details on the methodology see Bowman and Azzalini, 1997, pp. 59–62).

`sm-survival`

```
Args: (x y status h &key (hv 0.05) (p 0.5) (status.code 1)
                         (eval-points nil) (ngrid 50)
                         (display "lines") (add nil)
                         (xlab "x") (ylab "y")
                         (color 'black) (type 'solid) (widht 1))
```

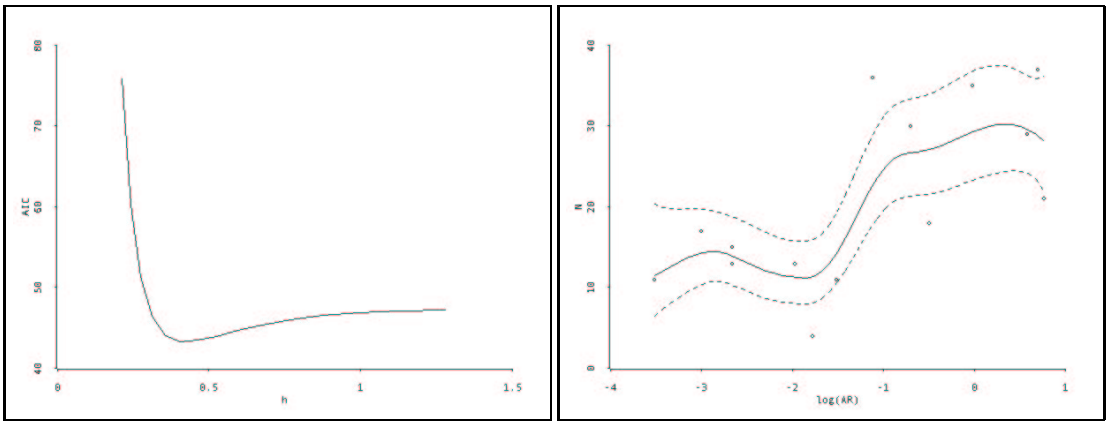

Figure 12: AIC plot for bandwidth selection (left panel) and the fitted smooth curve with variability bands for the regression of `N` on `log[AR]`

The required arguments are `x`, a list of covariate values, `y`, a list of survival times, and `status`, a list of indicator values defining the complete or censored survival times (by default an uncensored observation is identified by the status given through the `:status.code` argument). Also required is `h`, the smoothing parameter applied to the covariate scale. A normal kernel function is used and `h` is its standard deviation.

The smoothing parameter applied to the weighted Kaplan-Meier functions derived from the smoothing procedure in the covariate scale is provided by `:hv`. This ensures that a smooth estimate is obtained. The quantile to be estimated at each covariate value is given by `:p`, by default the median is estimated. The remaining arguments have the same meaning already discussed for the functions `sm-density`, `sm-regression`, etc.

This function returns a list whose elements are:

- the estimated quantiles at the covariate evaluation values;
- the covariate evaluation values;
- the smoothing parameter used for quantile estimation;
- the smoothing parameter used for the weighted Kaplan-Meier;
- the graph address of the plot.

Example 7.1 (Stanford data) As an example consider the data on the survival times of patients of different ages from the Stanford Heart Transplant Study. Thus, the response variable is taken to be the survival time on a log scale (`Log.time`) and the covariate is given the patient age in years (`Age`).

```
> (load "stanford")
> (def sm (sm-survival Age Log.time Status 7 :xlab "Age" :ylab "Log.time" :width 2))
> (sm-survival Age Log.time Status 7 :p 0.25 :add (select sm 4) :color 'blue)
> (sm-survival Age Log.time Status 7 :p 0.10 :add (select sm 4) :color 'red :type 'dashed)
> (send (select sm 4) :legend 58 4 '("Est.Surv curves:" "p = 0.50" "p = 0.25" "p = 0.10")
                      :color '(white black blue red)
                      :type '(solid solid solid dashed)
                      :width '(1 2 1 1)
                      :frame T)
```

The first call to the function `sm-survival` estimates the median of the survival times, while the following the 25th and 10th percentile respectively which are added to the first plot. A legend is also added to the plot, which is shown in Figure 13. ■

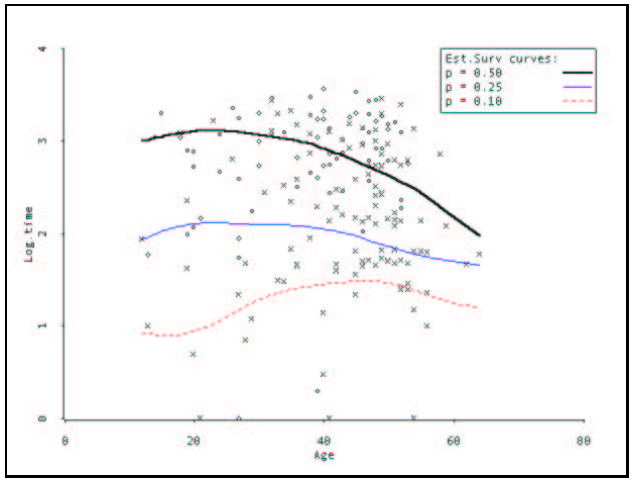

Figure 13: Smooth survival curves for the `Stanford` data at selected percentiles.

8 Extra functions accompanying the `sm` library

The file `sm-miscfun.lsp` provides several functions which can be of interest to *Xlisp-Stat* programmers and users. Most of them originated from existing *S-Plus* functions. They have been written during the development of the *Xlisp-Stat* version of the `sm` library, but hopefully they can result useful per se. In the following we briefly describe some of the most user-oriented, while the remaining functions can be read off directly from the source code.

Perspective plot

persp.s

```
Args: (x1 x2 y &key (add nil)     (spline 3)
                    (axes nil)    (frame T)
                    (type 'solid) (color 'black)
                    (variable-labels nil))

Creates a perspective plot given a matrix that represents heights on an
evenly spaced grid (similar to a S-Plus function).

x1 = vector containing x1 coordinates of the grid over which y is evaluated.
     The values should be in sorted order.
x2 = vector containing x2 coordinates of the grid over which y is evaluated.
     The values should be in sorted order.
y  = matrix of size length(x1) by length(x2) giving the surface height at grid
     points, i.e. y[i,j] is evaluated at (x1[i], x2[j]). The rows of y are
     indexed by x1, and the columns by x2.

add  = if nil a new plot is returned, otherwise the perspective plot
       is added to the graph object given as argument.
```

```
axes = if axis is T then axes are plotted on the graph.
frame = if frame is T then a frame box is drawn on the graph.
variable-labels = if not NIL they provide the labels for X, Y, and Z axes.
                  It must be a list of three strings.
```

Contour plot

contour.s

```
Args: (x1 x2 y &key (add nil) (variable-labels nil)
                    (nlevels 5) levels plevels
                    (color 'black) (type 'solid) (width 1))

Creates a contour plot given a matrix that represents heights on an
evenly spaced grid (similar to a S-Plus function).

x1 = vector containing x1 coordinates of the grid over which y is evaluated.
     The values should be in sorted order.
x2 = vector containing x2 coordinates of the grid over which y is evaluated.
     The values should be in sorted order.
y  = matrix of size length(x1) by length(x2) giving the surface height at grid
     points, i.e. x1[i,j] is evaluated at (x1[i], x2[j]). The rows of y are
     indexed by x1, and the columns by x2.

add     = if nil a new plot is returned, otherwise the contour is added to the
          graph object given as argument.
variable-labels = if not NIL they provide the labels for X1, Y, and x2 axes.
                  It must be a list of three strings.
nlevels = the approximate number of contour intervals desired. This is not
          needed if either levels or plevels are specified.
levels  = list of heights of contour lines. By default, nlevels lines are
          drawn which cover the range of y.
plevels = a list of heights of contour lines expressed as the proportion of
          the maximum height of the surface.
color   = color for contour lines (see *colors*)
type    = line type, either 'solid or 'dashed
width   = line width, an integer value by default equal to 1 (normal line)
```

Adds or draws polygons to a plot

polygon

```
Args: (x y &key (add nil) (frame nil) (color 'red)
                (title "Polygon Plot") variable-labels)

Adds or draws a polygon with the specified vertices to a graph object.

X, Y = lists of values providing the coordinates of the vertices of a polygon.
       Each list must be ordered, i.e. the ith point is connected to the i+1st.

ADD   = if nil a new plot is returned, otherwise the polygon is added to the
        graph object given as argument.
FRAME = if T a border is drawn around the polygonal region.
COLOR = specifies the color of the area within the polygon.
TITLE = a string for the plot title.
VARIABLE-LABELS = a list of two strings providing labels for the axes.

The function returns a plot object.
```

```
To remove a polygon send the message :remove-polygon to a graph object.
```

A `graph-proto` method for formatting axes labels and tick marks of a graph

```
:set-axis

Meesage args:  (axis &optional (min nil) (max nil)
                               (n.ticks nil) (n.minors nil)
                               (format nil))

This method allows formatting axes labels and tick marks.

axis = 0 for x axis and 1 for y axis.
min = the minimum of the axis
max = the maximum of the axis
n.ticks = the number of major tick marks
n.minors = the number of minor tick marks
format = a string specifying the format for labeling the major tick marks.
         Standard lisp format parameters can be used.
Any values which are not to be changed should be specified as NIL.
```

A `graph-proto` method for adding a legend to a graph

```
:legend

Message args: (x y legend &key color type width symbol
                               (frame T) (outer nil))

Adds a legend to the plot at the location (x,y), the top left corner of the
rectangle. The content of the legend is specified by the argument legend, a
list of string to be associated with color, line types, plotting symbols and
width. If frame is T then the a box surrounding the legend is drawn.
To remove a legend send the :remove-legend message to a graph.
```

Functions related to matrices

```
var

Args: (x &key (print nil) (varname nil))
If x is a matrix the function returns the sample variance-covariance matrix,
if x is list or a vector it returns the sampling variance.
If print is t then a nice printing of the covariance matrix is returned, else
the matrix in lisp format. varname may be a list of strings with the name
of the variables.
```

```
corr

Args: (x &key (print nil) (varname nil) (r-squared nil))
Computes the correlation matrix of x, which can be a matrix or a list of lists.
If r-square is T the squared correlation matrix is returned.
If print is t then a nice printing of the correlation matrix is returned, else
the matrix in lisp format. varname may be a list of strings with the name
of the variables.
```

```
diag

Args: (x)
If x is a matrix, returns the diagonal of x. If x is a sequence, returns a
diagonal matrix of rank equal to (length x) with diagonal elements equal to
```

```
the elements of x. If x is a number, returns an identity matrix of dimension
n x n.
```

```
dim
```

```
Args: (x)
Returns the dimensions of array x.
```

```
nrow
```

```
Args: (x)
Returns the number of rows of matrix x.
```

```
ncol
```

```
Args: (x)
Returns the number of columns of matrix x.
```

```
rbind
```

```
Args: (&rest args)
The ARGS can be matrices, vectors, or lists. Arguments are bound into a matrix
along their rows.
Example:
> (rbind (list 0 0) #2a((1 2)(3 4)) #(5 6))
#2A((0 0) (1 2) (3 4) (5 6))
```

```
cbind
```

```
Args: (&rest args)
The ARGS can be matrices, vectors, or lists. Arguments are bound into a matrix
along their columns.
Example:
> (cbind (list 0 0) #2a((1 2)(3 4)) #(5 6))
#2A((0 1 2 5) (0 3 4 6))
```

```
scale
```

```
Args: (x &optional (center nil))
Returns the standardized version of x, or if center is T the centered form
of x, which can be a list, a list of lists, or a matrix.
Both the standardization and the centering are performed by column.
```

```
block-matrix
```

```
Args: (x y)
Returns the block-diagonal matrix of dimension nrow(x)+nrow(y) by ncol(x)+ncol(y)
given by
             x | 0
             -----
             0 | y
```

Functions related to time-series data

```
diff.s
```

```
Args: (x &key (lag 1))
Translation of the S-Plus diff function.
```

```
autocov
```

```
Args: (x &optional (k 1) maxlag)
Returns the autocovariance function for the values in the list x.
The argument k provides the lags for the computation. If maxlag is
provided then k is set as a sequence of integer from 1 to maxlag.
```

```
autocorr
```

```
Args: (x &optional (k 1) maxlag)
Returns the autocorrelation function for the values in the list x.
The argument k provides the lags for the computation. If maxlag is
provided then k is set as a sequence of integer from 1 to maxlag.
```

```
acf-plot
```

```
Args: (y &key (time (iseq (length y)))
              (maxlag nil)
              (ACF nil)
              (alpha 0.05)
              (title "Correlogram"))
Plot autocorrelation function for the time series in the list y.
If ACF is provided, no autocorrelation function is computed but the values
provided are used instead.
```

9 Interface with *Arc*

Arc is a statistical software written by Cook and Weisberg (1999) in *Xlisp-Stat* for the analysis of regression data. In addition to the standard *Xlisp-Stat* environment, *Arc* provides a set of utilities for reading, analyzing and graphing data.

All the datasets used by Bowman and Azzalini (1997), as well as the datasets used for the examples of this document, are provided in *Arc* format (see Cook and Weisberg, 1999, 554–559). This means that they are plain text files containing the data values formatted by columns and some additional information, such as variable names and dataset description.

A minimal interface with *Arc* is provided. *Arc* allows to use scatterplot smoothers acting on a slider which appears on any 2D plot. By default a *lowess* smoother is used. A new item called **sm-reg** has been added to the nonparametric slidebar, and it can be accessed through the corresponding pop-up menu. This provides a very simple way of invoking the function `sm-regression`. The smoothing parameter, expressed via the approximate span discussed in Section 4.1, is then selected by dragging the slider.

Local likelihood models may also be fitted through some items added to the nonparametric slidebar. These allows to smooth binomial, Poisson and gamma data using canonical link functions. The smoothing parameter is again selected through the corresponding slidebar. Finally, a slidebar has been added to any 3D plot drawn from *Arc*, from which a smooth regression function for the two predictors case may be obtained and the resulting smooth curve added to the plot.

References

Bowman A.W. and Azzalini A. (1997), *Applied Smoothing Techniques for Data Analysis*. Oxford, Oxford University Press.

Bowman A.W. and Azzalini A. (2001), Computational aspects of nonparametric smoothing with illustrations from the sm library. To appear.

Burnham K.P. and Anderson D.R. (1998), *Model Selection and Inference. A Practical Information-Theoretic Approach*. New York, Springer-Verlag.

Cleveland W.S. (1993), *Visualizing Data*. Summit, New Jersey, Hobart Press.

Cleveland W.S. and Devlin S.J. (1988), Locally weighted regression: An approach to regression analysis by local fitting. *Journal of the American Statistical Association*, 83, 596–610.

Cook R.D. and S. Weisberg (1999), *Applied Regression Including Computing and Graphics*. New York, Wiley.

Fan J., Heckman N.E. and Wand M.P. (1995), Local Polynomial Kernel Regression for Generalized Linear Models and Quasi-likelihood Functions. *Journal of the American Statistical Association*, 90, 141–150.

Hand D.J., Daly F., Lunn A.D., McConway K.J. and Ostrowski E. (1994), *A Handbook of Small Data Sets*. London, Chapman and Hall, pp. 41-42.

Loader C. (1999), *Local Regression and Likelihood*. New York, Springer.

Hastie T. J. and Tibshirani R. J. (1987), Local Likelihood Estimation, *Journal of the American Statistical Association*, 82, 559-567.

Hastie T. J. and Tibshirani R. J. (1990), *Generalized Additive Models*. London, Chapman and Hall.

Henderson, R. and Sheppard, H. N. (1919), *Graduation of mortality and other tables*. Actuarial society of America, New York.

McCullagh P. and Nelder J.A. (1989), *Generalized Linear Models* (2nd edn). London, Chapman and Hall.

Nelder J.A. and Wedderburn R.W.M. (1972), Generalized Linear Models. *Journal of Royal Statistical Society* A, 135, 370–384.

Silverman B.W. (1985), Some aspects of the spline smoothing approach to non–parametric curve fitting. *Journal of the Royal Statistical Society*, B, 47, 1–52.

Steele G.L. (1990), *Common Lisp the Language*, 2nd edition, Digital Press.

Tibshirani R. J. (1984), *Local Likelihood Estimation*, Ph. D. Thesis, Department of Statistics, Stanford University.

Tierney L. (1990), *Lisp–Stat: An Object–Oriented Environment for Statistical Computing and Dynamic Graphics*. New York, Wiley.

Tierney L. (1991), *Generalized Linear Models in Lisp–Stat*, Technical Report No. 557, School of Statistics, University of Minnesota.
URL (html): `http://www.stat.umn.edu/~luke/xls/glim/glim/glim.html`
URL (PostScript): `ftp://ftp.stat.umn.edu/pub/xlispstat/doc/glim.ps`

Wand M.P. and Jones M.C. (1995), *Kernel Smoothing*. London, Chapman and Hall.