Splancs Supplement - *Spatial and Spatial-Temporal* Analysis

B. S. Rowlingson & P. J. Diggle
Lancaster University, Lancaster, U.K.

July 12, 1996

1 Introduction

Rowlingson and Diggle (1993) describe a set of functions written for the Splus package for performing statistical analysis of spatial point patterns. These functions are known as Splancs and were made available to Splus users for a small charge.

Further functions have been written that work with Splus and Splancs. These concentrate on the analysis of space-time patterns and modelling raised incidence, but additional routines in the purely spatial domain have also been added.

2 Space-time clustering analysis

For space-time clustering analysis, the data consist of a set of events in two-dimensional Euclidean space and time. The events are contained within a polygonal region in space and between upper and lower temporal bounds.

The methodology of space-time clustering can be found in Diggle, Chetwynd, Haggkvist and Morris (1993). An application can be found in Bhopal, Diggle and Rowlingson (1992).

In the function definitions to follow, `pts` is a set of events as used by Splancs and `times` is a vector of the event times. The length of `times` must be the same as the number of points in `pts`. Also, `poly` is a polygon enclosing the points, and `tlimits` is a vector of length two specifying the start and finish of the temporal range. The two vectors `s` and `t` define the spatial and temporal values at which to calculate functions.

The following functions are now included in Splancs.

- `stkhat(pts,times,poly,tlimits,s,t)` estimates the K-function in space, time, and space-time. It returns a list with the following components:

  `$s` The input distance scale vector.

  `$t` The input time scale vector.

$ks The spatial K-function estimate, $\hat{K}_s(s)$. This is a vector of length `length(s)`.

$kt The temporal K-function estimate,$\hat{K}_t(t)$. This is a vector of length `length(t)`.

$kst The space-time K-function estimate, $\hat{K}_{st}(s,t)$. This is a matrix of dimension `[length(s), length(t)]`.

- `stsecal(pts,times,poly,tlimits,s,t)` returns the standard error of the difference between the space-time K-function estimate and the product of spatial and temporal K-functions. The returned value is a matrix of dimension `[length(s), length(t)]`.

- `stvmat(pts,times,poly,tlimits,s,t)` is a function to compute the variance-covariance matrix for the difference between the space-time K-function and the product of the spatial and temporal K-functions. The returned value is a four-dimensional array with dimensions `[length(t), length(s), length(t), length(s)]`. The covariance at distance `s1`, time `t1` and distance `s2`, time `t2` is given by the array element corresponding to `[t1,s1,t2,s2]`

- `stmctest(pts,times,poly,tlimits,s,t,nsim,quiet=F)` performs a Monte-Carlo test of space-time clustering by randomly permuting the time-components of the data and computing the sum of the differences between the space-time K-function and the product of the spatial and temporal functions over the distance and time scales defined by `s` and `t`. This sum is then used as a test statistic.

  The number of trials is given by `nsim`. If `quiet=T` then no output is printed – otherwise the function will report the number of simulations performed and the rank of the test statistic for the data so far. The return value is a vector of length `nsim`. The first element of the vector is the value of the statistic for the actual data, and the rest are those for the temporal permutations.

- `stdiagn(pts,stkh,stse,stmc)` plots four views derived from the data and the space-time K-functions. The plot area is divided into four with `par(mfrow=c(2,2))`. In the top left a spatial map of the data is drawn. Top right is a perspective plot of the difference between the space-time K-function and the product of the spatial and temporal K-functions. Lower left is a plot of the standardised residuals

  $$(\hat{K}_{st}(s,t) - \hat{K}_s(s)\hat{K}_t(t))/\mathrm{SE}(\hat{K}_{st}(s,t))$$

  against $\hat{K}_s(s)\hat{K}_t(t)$.

  The lower right plot is a histogram of the test statistics returned by the `stmctest` function. The value of the statistic for the data is indicated with a vertical line.

  The arguments of `stdiagn` are a set of points, and the objects returned from running `stkhat`, `stsecal`, and `stmctest` with those points. If the polygon or spatial or temporal axes are not the same for each of the calls to those functions, then the diagnostic plots will be meaningless.

  If the `stmc` parameter is not supplied then the fourth plot will be left blank.

  Figure 1 is an example for a set of random points on a unit square and between times zero and one. Note that the D-plot shows no obvious structure, the standardized residuals are not significantly above zero, and the test statistic for the data is nowhere near the upper extreme of the simulated statistic.

3 Space-time Kernel Smoothing

Kernel smoothing of spatial data is discussed in Berman and Diggle (1989). The Splancs package included routines for implementing some of their methods in two dimensions. We now present two routines that use kernel smoothing in three dimensions, where the third dimension is that of time.

- `kernel3d(pts3,xgr,ygr,zgr,hxy,hz)` implements kernel smoothing of three dimensional, and in particular, space-time data. The object `pts3` is a 3-column matrix with each row containing x, y, and t coordinates. The three vectors `xgr`, `ygr`, `zgr` specify the space-time domain over which to evaluate the kernel function. The scalar `hxy` is the kernel width in the x and y coordinate directions, and `hz` specifies the kernel width in the time dimension.

  The kernel used is the quartic, as used in the kernel2d function of Splancs.

  The output from this is a list with several components. The first five are just copies of the three grid specification vectors and the two kernel widths for reference, but the sixth component, `$v`, is a 3-dimensional array of the kernel function evaluated at the points in `xgr`, `ygr`, `zgr`. The third dimension specifies time – hence to look at slices of the kernel function in time we can use `image(kernobj$v[,,3])` for example. To specify the x and y axes correctly, we use

  ```
  image(x=kernobj$xgr,y=kernobj$ygr,z=kernobj$v[,,7]),
  ```

  for example.

- `kerview(pts,times,kernobj,map=T,addimg=T,ncol=1)` is a linked-window system for browsing space-time data. One graphics device must already be opened, and two others are started. The original device is used to display slices in time from the kernel smoothing object `kernobj`. To the right of this is a window which displays the points that contribute to that slice of the kernel smoothing being displayed in the first window. At the bottom of the screen is a histogram of the time coordinates in the data. Clicking on this with the first mouse button sets the time-limits for the slice of the data displayed in the other two windows. Clicking with mouse button two terminates the function and removes the two additional graphics windows.

  Setting `map=F` just produces the kernel images and time histogram. The default `addimg = T` overwrites each image with the new one, whereas `addimg = F` starts a new plot on that graphics device. The default produces smoother results, but as successive plots are stored in the graphics device display memory, screen refreshes can become very slow.

  The `ncol` parameter sets the image and map windows to display with `par(mfrow=c(ncol,ncol))`. This enables several images and maps to be compared. Setting this option to anything greater than one results in `addimg` being set to `F`.

  A sample screen dump is shown in figure 2. Note that this is purely for illustration. The practical value of the kernel smoothing is realised with large data sets, for which a plot of the individual points can be hard to interpret.

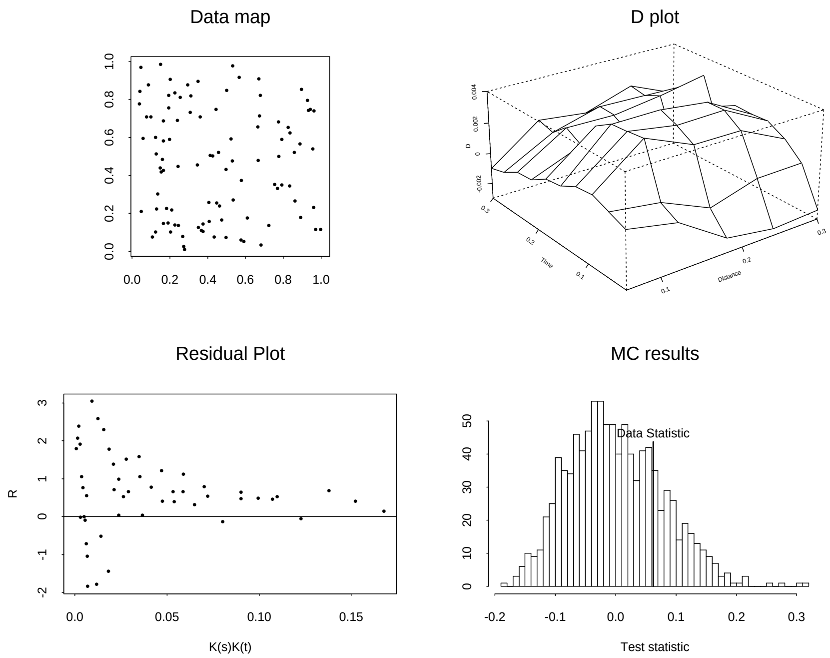

Figure 1: Diagnostic plots for space-time clustering

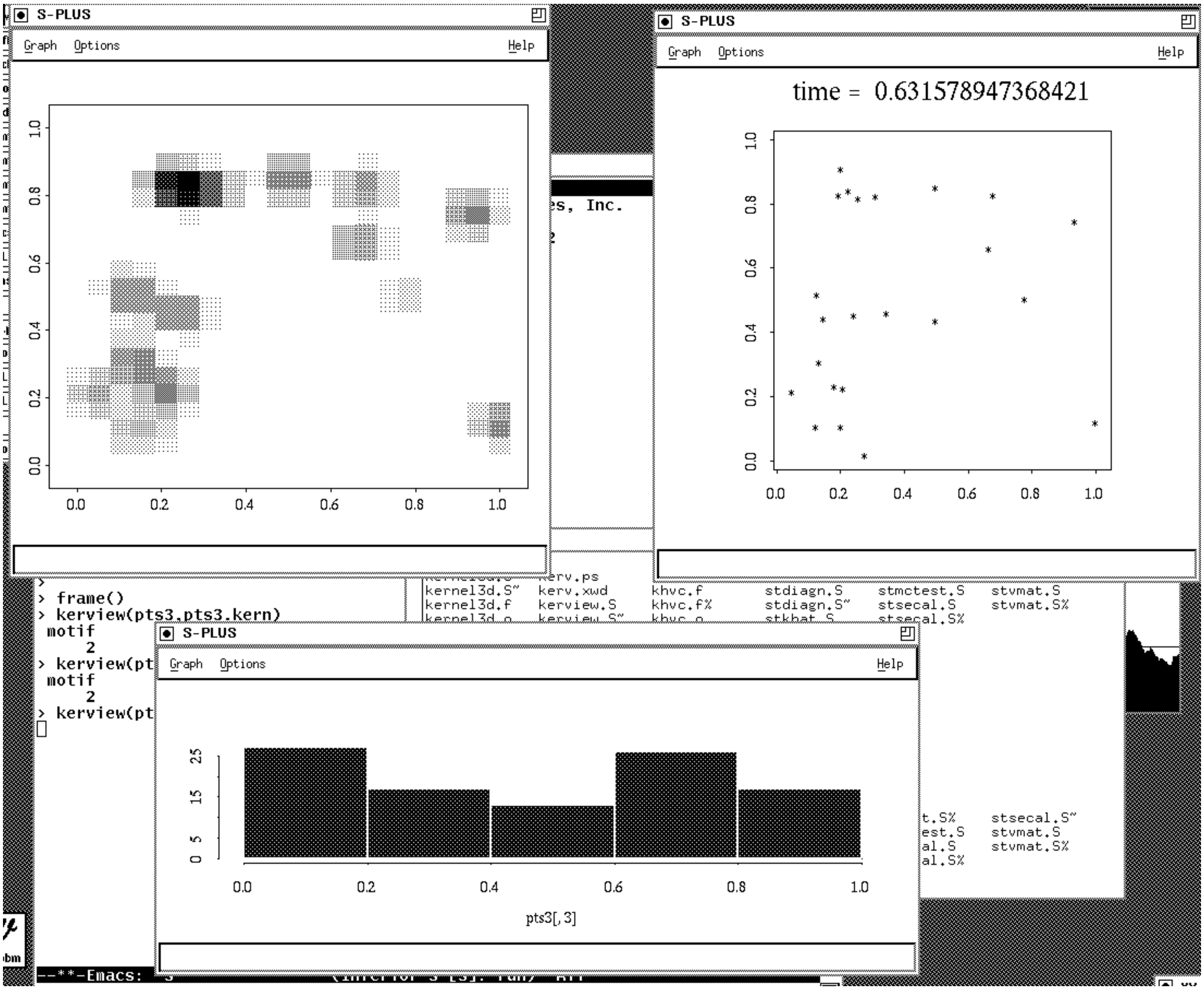

Figure 2: kerview application

4 Raised Incidence Modelling

Some additional functions relevant to modelling of raised incidence around a suspected source have been included. For further details see Diggle and Rowlingson (1993).

In raised incidence modelling the data consist of information on the location and characteristics of a number, N_p, of individuals and a number, N_s, of sites of sources of raised incidence. Typically, the individuals' data will consist of a binary $(0, 1)$ variable indicating whether the individual is a control or case, together with other information relevant to the study in hand. From the locational data the distance of each individual to the each source is calculated. The incidence of the disease is then modelled as a function of distance from the sources.

The probability, p, of an individual being a case is parameterized as:

$$p = \frac{\rho f}{1 + \rho f}$$

where ρ is the ratio of cases to controls, and f is defined as:

$$f = \prod_{i=1}^{N_s} (1 + \alpha_i \exp(-\beta_i d_i^2)) \prod_{j=1}^{N_c} \exp(\theta_j Y_j).$$

This splits into two parts. The first product is concerned with relating incidence to point source distances, and the second part with other covariates.

In the first product, N_s is the number of sources of raised incidence, α_i and β_i are parameters and d_i is the distance of the point to source i.

Note that two or more point sources may have their raised incidence profiles modelled by the same parameter pair (α_i, β_i) if there is an *a priori* reason for believing the sources to have the same effect on incidence. This will reduce the number of distinct parameters in the model.

In the second product, N_c is the number of other covariates included, Y_j is the value of covariate j for the individul in question, and θ_j is a parameter.

The log-likelihood for all cases and controls can then be written as

$$l = \sum_{\text{cases}} \log(p) + \sum_{\text{controls}} \log(1 - p),$$

which is to be maximised with respect to the parameters ρ, α, β, and θ.

The following functions have been developed to implement this model:

- `tribble`

  The `tribble` function maximises the log-likelihood with respect to the parameters. It takes up to 11 arguments, but not all need to be specified for a particular case. For the simplest case it is only necessary to specify the following:

  ```
  tribble( ccflag,  vars, alphas, betas, rho )
  ```

This specifies that we model the incidence as a function of the distance from a number of sources.

The vector `ccflag` is a vector of ones and zeroes specifying a case or control respectively. The length of this vector is equal to the number of points being considered, N_p.

The squared distances to the sources are stored in the matrix `vars`. This is of dimension N_p by N_s, wher N_s is the number of sources beng investigated.

The initial values of the parameters are given in `alphas`, `betas` and `rho`. Since there is one α and β parameter for each source, the length of `alphas` and `betas` must be equal to N_s. There is one ρ parameter, and so `rho` is of length one.

If further information on each case or control is available it can be incorporated into the model as a log-linear covariate term. The argument `covars` is a matrix specifying the values of the covariate for each of the cases and controls. Each row of `covars` specifies a different covariate. An initial parameter estimate must be given with the `thetas` argument to `tribble`. This must be of the same length as the number of covariates.

An argument `which` may be given to `tribble`. This enables distances from multiple sources to modelled by the same pair of α and β parameters. This may be useful if some sources are believed to be having the same effect on incidence. The length of `which` must be equal to the number of sources. Source `i` is modelled with `alpha[which[i]]` and `beta[which[i]]`.

Three further arguments, `steps`, `reqmin`, and `icount` control the operation of the maximisation algorithm.

The function returns a list with the following components:

`alphas` A vector of length `Nd` giving the value of each of the α parameters at the maximum.

`betas` A vector of length `Nd` giving the value of each of the β parameters at the maximum.

`thetas` A vector of length `Nc` giving the value of each of the θ parameters at the maximum.

`rho` Scalar, giving the value of the ρ parameter at the maximum.

`logl` Scalar, the value of the log-likelihood at the maximum.

`null.logl` Scalar, giving the value of the log-likelihood of the null hypothesis of no raised incidence.

`call` An object of mode 'call', giving the function call that produced this object, for later reference.

The object returned has class 'ribfit'. A function `print.ribfit` is included to dispay the results in a simpler, neater format,

- `triblik`

  This function computes the log-likelihood for a given set of points and parameters. It is this that is maximised by `tribble`. Its calling sequence is similar to `tribble`.

```
triblik( ccflag, vars, alphas, betas, rho,
         which, covars, thetas)
```

It returns the single value of the log-likelihood.

- `dsquare(pts,srcs,namepref='d')`

  This function is used for computing distances from point-to-point distances. The object `pts` is a set of `Np` points representing cases or controls, and `srcs` is a set of `Ns` points representing the putative sources of raised incidence. Then `dsquare` returns a data frame with dimensions `[Np,Ns]` giving the distance squared for each point to each source. The names of the columns of this data frame are derived by postfixing `namepref` with the number of the source.

5 Spatial K-function analysis

The Splancs library includes several functions for case-control studies using K-functions. In particular, `khvmat` calculates the covariance matrix for the difference between two K-functions under random labelling. In some applications it may be useful to compute the contribution to the variance due to the cases and controls separately.

- `khvc(pts1,pts2,poly,s)` computes the covariance matrix for the difference in K-functions. The object `pts1` defines the locations of the cases, and `pts2` the locations of the controls. The return value is a list with the following components:

  `$varmat` The upper triangle of the covariance matrix.

  `$k11` The variance of $\hat{K}$ for the cases.

  `$k22` The variance of $\hat{K}$ for the controls.

  `$k12` The covariance of $\hat{K}$ for the cases and $\hat{K}$ for the controls.

6 Availability

Please email `B.Rowlingson@lancaster.ac.uk` for current availability of the Splancs functions.

References

Diggle P.J. and Rowlingson B.S. (1993) A Conditional Approach to Raised Incidence. *J. R Statist Soc. A* (to appear).

Rowlingson B.S. and Diggle P.J. (1993) Splancs: Spatial Point Pattern Analysis Code in S-plus *Computers and Geosciences* **19** 627–655

Diggle P.J., Chetwynd A.G., Haggkvist R., and Morris S. (1993) Second-order Analysis of Space-time Clustering. *J. R. Statist Soc. A* (submitted).

Berman M. and Diggle P.J. (1989) Estimating Weighted Integrals of the Second-order Intensity of a Spatial Point Process *J. R. Statist Soc* **51** 81–92

Bhopal R., Diggle P.J. and Rowlingson B.S. (1992) Pinpointing Clusters of Apparently Sporadic Cases of Legionnaires' Disease. *British Medical Journal* **304** 1022–7

Appendix : Help files

The following appendix gives the Splus online function documentation files. These files are available from within Splus by typing `help(function.name)`.