TECHNICAL REPORT
ST-99-09

`geoR/geoS`: functions for geostatistical analysis using *R* or *S-PLUS*

Revised version
(see date below)

Paulo J. Ribeiro Jr.
Lancaster University / UFPR

and

Peter J. Diggle
Lancaster University

Department of Mathematics and Statistics
Lancaster University
LA1 4YF Lancaster-UK

06 December 2000

Abstract

A set of functions for geostatistical analysis is presented. These functions are grouped in a *R* package called `geoR` and its *S-PLUS* counterpart, `geoS`. The functions in this library implement geostatistical methods including descriptive analysis, parametric and non-parametric variogram estimation and fitting, some types of kriging and simulation of Gaussian random fields. Likelihood-based methods and Bayesian inference/prediction for Gaussian and transformed Gaussian models are also implemented. `geoR/geoS` is a public domain packge/library and instructions about how to download and install the current version are provided. The implemented functions and classes are listed divided into categories. Some commands used to simulate and analyse a data-set are reproduced in this document illustrating the main features of the library. Details about the functions, their arguments and options are also provided.

Keywords: R, package, S-PLUS, library, geostatistics, spatial statistics.

Contents

1 Introduction

This report describes a set of computational tools for the analysis of geostatistical data. The data considered here are of the form (u_i, y_i), where y_i denotes a measurement at a location $u_i \in \mathbb{R}^2$. Statistical analysis of geostatistical data requires exploratory and graphical tools as well as computational resources for extensive calculations.

The softwares *R* (Ihaka and Gentleman, 1996) and *S-PLUS* (Mathsoft (1993), Becker et al. (1988)) provide a favourable environment for this kind of analysis, given its graphical tools, availability of generic functions and flexibility for programming. The standard functions in `R/S-PLUS` are not suitable for geostatistical data analysis. Additional functions and/or libraries are necessary. Some sources of `S-PLUS` functions for geostatistical analysis includes the *S-PLUS Spatial Stats Module* (Mathsoft, 1996), the *Krigit Library* (an unpublished set of routines by B.S. Rowlingson, Lancaster University), the functions provided by Venables & Ripley (1997) and the `R` package `sgeostat` .

A public-domain library for geostatistical analysis, named `geoR/geoS`, is presented here. This report assumes familiarity with geostatistical jargon. Most of the geostatistical methods implemented are described in Journel & Huijbregts (1978), Cressie (1993), Goovaerts (1997), Diggle et al. (1998), Ribeiro Jr & Diggle (1999), Diggle & Ribeiro Jr (2000). Although some tasks are similar to those performed by the other sources, the `geoR/geoS` library also has some distinctive aspects including likelihood-based methods and Bayesian prediction for (transformed) Gaussian models.

The main features of the `geoR/geoS` library are:

- functions for exploratory analysis,
- several variogram plot options,
- variogram envelops,
- a wide range of models for variograms (correlation functions),
- functions for parametric/non-parametric estimation of variogram (covariance) parameters,
- options for computation/plotting of 1D and 2D profile likelihood,
- a general and flexible linear kriging routine which performs simple, ordinary, universal and external trend kriging,
- a function to perform Bayesian parameter estimation and prediction, i. e. spatial prediction taking into account the uncertainty in model parameters,
- function for simulation of Gaussian random fields,

The functions are divided in categories according to the usual steps of a geostatistical analysis. The categories are:

1. data preparation,
2. descriptive analysis,

3. variogram plots,

4. variogram model fitting,

5. spatial prediction (kriging and Bayesian),

6. simulation of Gaussian random fields,

Section 2 provides instructions for downloading and installing the library. Section 3 provides a table with function names divided by category. An example of data analysis using the functions is given in Section 5. The functions are described in detail in Section 6.

Throughout this text, function names, commands and arguments, are always written using the LaTeX `typewriter` font. For example `variog` is the name of the function which computes an empirical variogram.

2 Installing and using the functions

The `geoR/geoS` package/library can be downloaded from:

www.maths.lancs.ac.uk/~ribeiro/geoS.html.

Curretly there are versions for:

1. *R* for Linux
2. *R* for Windows
3. *S-PLUS 3.4* for Windows
4. *S-PLUS 5.x* for Unix
5. *S-PLUS 5.x* for Linux
6. *S-PLUS* for Windows

The `geoR/geoS` home-page also describes the steps to download the latest version of the package/library and provide instruction about how to install the library on the different operational systems.

By the time this document was revised, help files were being written and some of them were already available for the `geoR` distribution.

DISCLAIMER: `geoR/geoS` is still under development. We cannot guarantee correctness. Upgrades are constantly made and indicated in the home-page. Feedbacks are welcome.

3 Available functions

3.1 Functions names and categories

The table below presents the names of *geoS* functions currently available. They are separated by categories which represent sequential stages of data analysis.

The functions indicated inside parentheses are auxiliary functions, typically called by other functions to perform intermediate computations.

Category	**Functions**
data preparation	`polygrid`
descriptive analysis	`plot.geodata`, `points.geodata`, `xyzplot`*
empirical variogram	`variog`, `variog.env`, `(rfm.bin)`
variogram results, plots and lines	`summary.variomodel`, `plot.variogram`, `lines.variogram`, `lines.variomodel`
non-parametric variogram fitting	`olsfit`, `(loss.olsvario)`, `wlsfit`, `(loss.wlsvario)`
parametric variogram fitting	`likfit`, `(proflik.*)`
profile likelihood	`proflik`, `(proflik.*)`, `plot.proflik`
kriging	`krige.conv`, `krige.bayes`, `(dist0.krige)`, `(krige.bayes.aux*)`
simulation of Gaussian random fields	`grf`, `(messa.grf)`, `(grid.grf)`, `(grf.aux*)`, `grfclass`, `plot.grf`, `image.grf`, `wireframe.grf†levelplot.grf†`
other generic auxiliary functions	`(cov.spatial)`, `(varcov.spatial)`, `(coords.aniso)`, `(loglik.spatial)`, `(matern)`, `(trend.spatial)`

†*S-PLUS* only.
function from *Krigit* library, supplied by B.S. Rowlingson.

3.2 New *R/S-PLUS* object classes

Some of the functions above assign classes to the output. Objects with an assigned class can be easily handled by generic functions like `plot`, `lines`, etc. The table below lists the object classes and the functions which generate objects of those classes.

Class	**Functions**
variogram	`variog`, `rfm.bin`
variomodel	`likfit`, `olsfit`, `wlsfit`
grf	`grf`, `grfclass`
kriging	`krige.bayes`, `krige.conv`
krige.bayes	`krige.bayes`
proflik	`proflik`

4 Correlation functions $\rho(x)$ used in geoR/geoS

In the following, x is the distance between two locations and $\phi > 0$ is a range parameter. $K_\kappa(x)$ is the modified Bessel function of the third kind of order κ: for small x, $K_\kappa(x) \sim \frac{1}{2}\Gamma(\kappa)\left(\frac{x}{2}\right)^{-\kappa}$.

Table 1: Common parametric spatial correlation functions

usual name	functional form	parameter bounds
exponential	$\exp(-x/\phi)$	
wave	$\frac{\phi}{x}\sin\left(\frac{x}{\phi}\right)$	
matern	$\rho_m(x\vert\phi,\kappa) = \frac{1}{2^{\kappa-1}\Gamma(\kappa)}\left(\frac{x}{\phi}\right)^\kappa K_\kappa\left(\frac{x}{\phi}\right)$	$\kappa > 0$
gaussian	$\exp(-(x/\phi)^2)$	
spherical	$\rho(x) = \begin{cases} 1 - 1.5(x/\phi) + 0.5(x/\phi)^3 & \text{if } x < \phi \\ 0 & \text{otherwise} \end{cases}$	
circular	Let $\theta = \min(x/\phi, 1)$ and $\gamma(x) = 2(\theta\sqrt{1-\theta^2} + \sin^{-1}\sqrt{\theta})/\pi$ $\rho(x) = \begin{cases} 1 - \gamma(x) & \text{if } x < \phi \\ 0 & \text{otherwise} \end{cases}$	
cubic	Let $\theta = \min(x/\phi, 1)$ and $\gamma(x) = 7\theta^2 - 8.75\theta^3 + 3.5\theta^5 - 0.75\theta^7$ $\rho(x) = \begin{cases} 1 - \gamma(x) & \text{if } x < \phi \\ 0 & \text{otherwise} \end{cases}$	
powered exponential	$\exp(-(x/\phi)^\kappa)$	$0 < \kappa \leq 2$
cauchy	$[1 + (x/\phi)^2]^{-\kappa}$	$\kappa > 0$
gneiting	Let $\theta = \min(x/\phi, 1)$ $\rho_g(x\vert\phi) = (1 + 8\theta + 25\theta^2 + 32\theta^3)(1-\theta)^8$	
gneiting matern	Let $\tau = \phi\kappa_2$ $\rho(x) = \rho_g(x\vert\tau)\,\rho_m(x\vert\phi,\kappa_1)$	$\kappa_1 > 0$ $\kappa_2 > 0$

5 Example

This section illustrates the use of some of the main `geoR/geoS` functions by analysing a simulated data-set. Reproduction of the commands and inspection of the objects generated by them provide a general overview of the methods currently implemented. Most of the functions have large number of arguments and many of them have default options. In these notes for most of the cases the functions are used without modifying these default arguments, However this may not be appropriate when analysing other data-sets and the arguments must be carefully inspect. For more details about argument options for each function see Section 6, the documentation files or type:

```
> args(name.of.the.function.here)
```

Most of the terminology used in this report and in the functions follows the geostatistical jargon and the notation and model description found in Ribeiro Jr & Diggle (1999) and Diggle & Ribeiro Jr (2000). In general, there are almost no differences between the commands when running *R* or *S-PLUS*. Plataform specific details will be commented when relevant. The examples shown here do not cover all of the package/library features.

5.1 Generating the data-set and exploratory analysis

Consider the generation of a simulated data-set with 100 points located in a square area with dimensions 1×1. The sample is a realisation of a process with zero mean, unit variance ($sill = 1$) and exponential correlation function with parameter $\phi = 0.3$ ($range = 0.3$). The data-set analysed here was generated using the function `grf` running on *geoS/S-PLUS*, using the random seed given below. More details about the function `grf` will be provided later in this section.

To generate the same data set using *geoS* type:

```
> .Random.seed <- c(37,22,26,1,45,3,60,52,36,6,8,3)
> s100 <- grf(100,cov.pars=c(1,.3))
```

This data set is included in the *geoR* distribution. Therefore, to get the same data using *geoR* just type:

```
> data(s100)
```

Notice that the random seed specification differs in *S-PLUS* and *R* and this explains why the **same** data-set can not be generated using *geoR*. Alternativelly to the commands above, the user can download the simulated data from the `geoR/geoS` home-page, place the downloaded file (`s100.data`) in the working directory and load it in a *R* or *S-PLUS* session using the command:

```
> source(``s100.data'')
```

For this data-set, the data locations are randomly distributed according to a Poisson process on a unit square. Regular grids can be obtained by using the option `grid='reg'`. The user

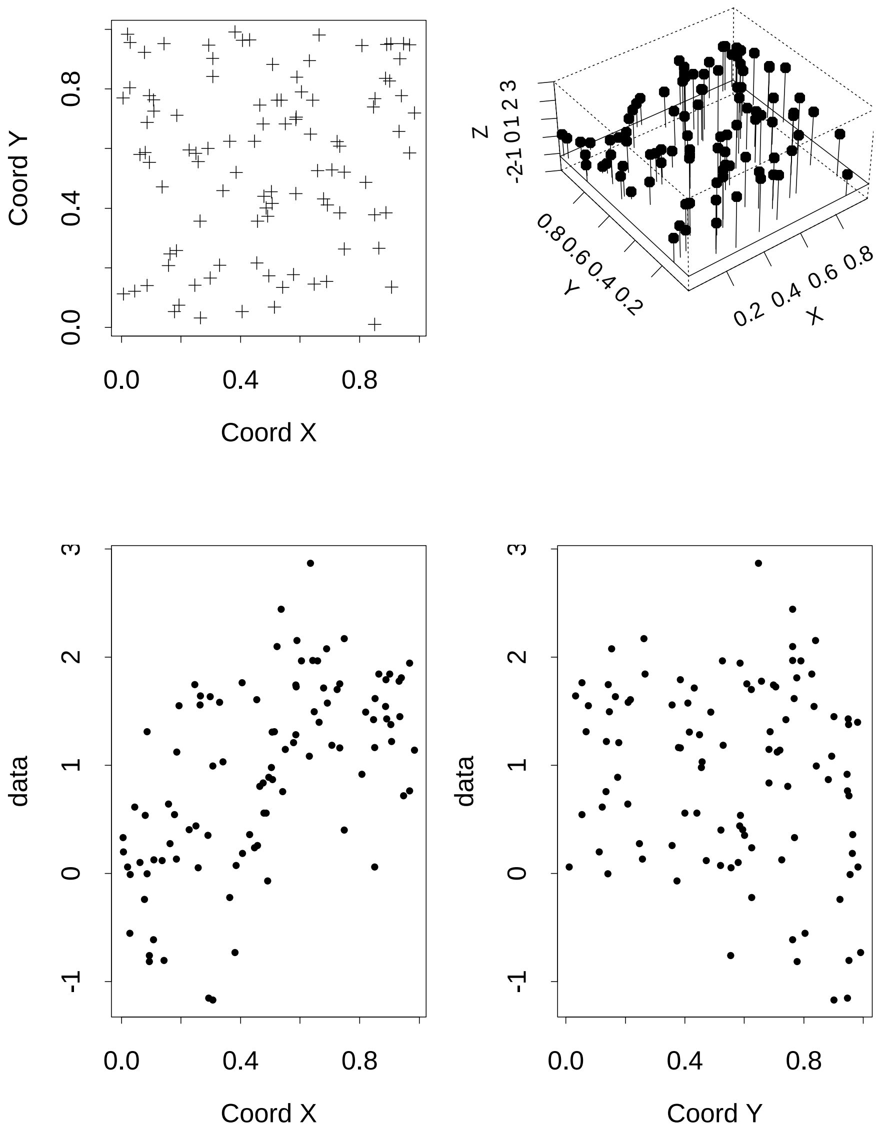

Figure 1: Output of `plot.geodata`.

can also specify a particular set of locations through an $n \times 2$ matrix, whose rows define a set of n data location.

The function `plot.geodata` returns a multiple plot for a quick inspection of the data (Figure 1). If provided, the option `window=T` opens a graphic windows using the command `motif()`.

```
> plot.geodata(s100)
```

The plot in the top left corner of Figure 1 shows the data locations. On the top right, the data values are also included producing a 3-D plot. On the bottom data values are plotted against each of the coordinates. This output can be used to identify discrepant data, trends associated with directions in the area and the data location configuration.

An alternative display of data locations and values is given by the function `points.geodata`. Figure 2 shows two types of output. On the left panel the points sizes are proportional to the data values. On the left panel data were divided in deciles and plotted with increasing sizes through the deciles. The is also optionj for quartiles and points sizes (minimum and maximum), colors and symbols can be specified by the user. The commands to produce these plots are:

```
> points.geodata(s100, xlab=''Coords X'', ylab=''Coord Y'')
> points.geodata(s100, pt.sizes="deciles", xlab=''Coords X'', ylab=''Coord Y'')
```

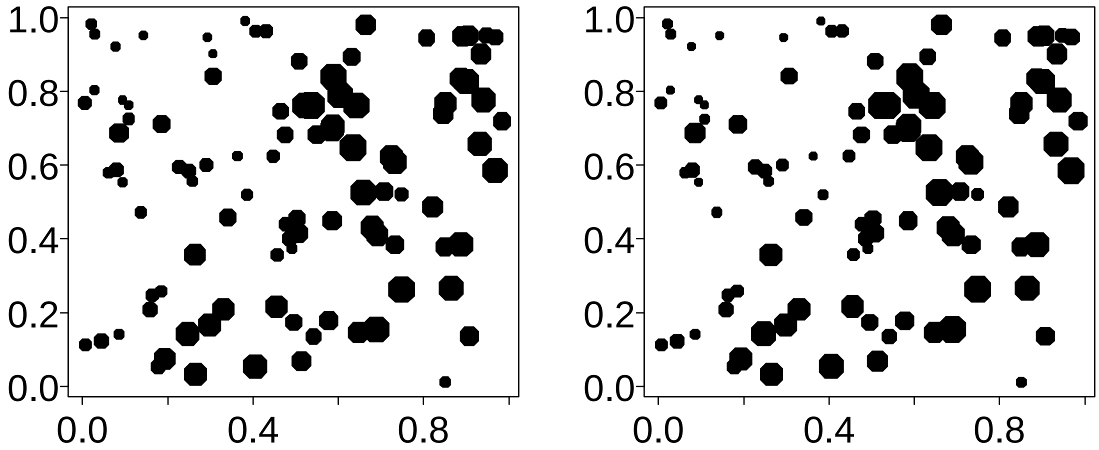

Figure 2: Displays for data location and values, left: points sizes proportional to data values, right: data divided in deciles.

5.2 Empirical variograms

Empirical variograms are computed in two stages: a variogram cloud and a binned (grouped in classes of distance) variogram. The first can be useful to identify anomalous pairs of points which can be highly influential in parameter estimation. The second usually provides a better visual description of the spatial dependence and can be used to compare fitted models.

The function `variog` is a generic function to compute empirical variograms. It can return the variogram in three basic formats:

- as a variogram cloud, i. e. distances and variogram values for all pairs of points (`option ='cloud'`);
- as a binned variogram (the default option), i. e. the cloud is grouped in classes of distances (`option ='bin'`);
- as a smooth variogram, i. e. a smoothed version of the variogram cloud (`option ='smooth'`).

For `option ='cloud'` the variogram values can be computed as:

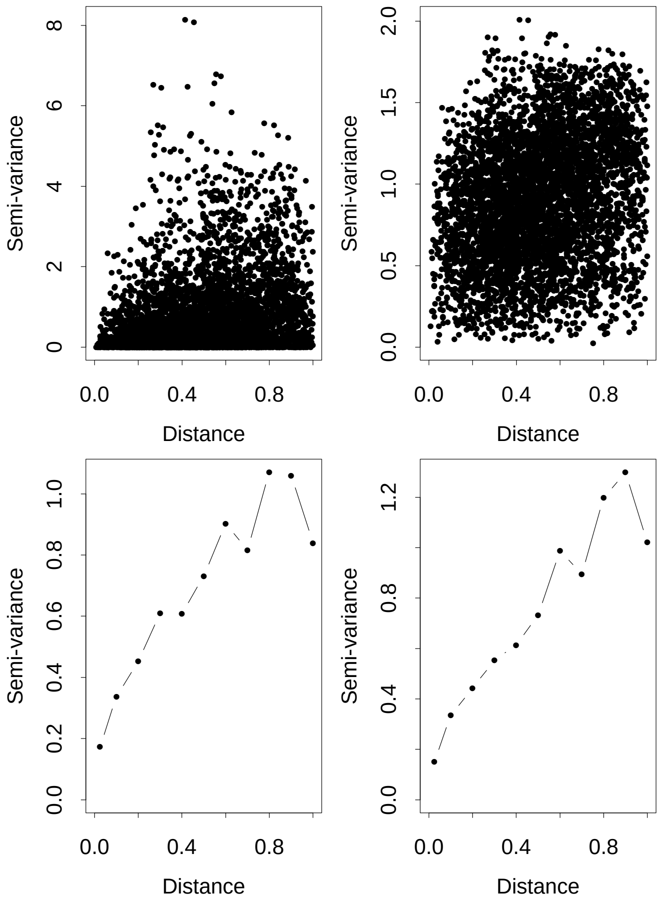

Figure 3: Variogram clouds and binned variograms.

- half squared differences, $\frac{(Y(u+h)-Y(u))^2}{2}$, if `estimator.type ='classical'` (the default option);
- square root differences, $|Y(u+h)-Y(u)|^{\frac{1}{2}}$, if `estimator.type ='robust'`.

Two estimators are implemented for the binned variogram. The default option is the classical estimator (Matheron, 1963) given by:

$$\hat{\gamma}(h) = \frac{1}{2N(h)} \sum_{N(h)} (Y(u+h) - Y(u))^2,$$

where $N(h)$ is the number of pairs of points separated by the distance h. It's common to consider an tolerance ε for the distances and than h is replaced by the interval $(h-\varepsilon, h+err)$.

With the option `estimator.type ='robust'`, the function computes the robust estimator proposed by Cressie & Hawkins (1980):

$$\overline{\gamma}(h) = \frac{\left[\frac{1}{N(h)} \sum_{N(h)} |Y(u+h) - Y(u)|^{\frac{1}{2}}\right]^4}{0.914 + \frac{0.988}{N(h)}}.$$

Some extra arguments for binned variograms, all of them with default options, are available, making the function more flexible.

- The argument `uvec` specifies the midpoints of the distance classes.

- A value of distance can be specified in `nugget.tolerance`. Pairs of points separated less than this distance are considered to be at the same location. This is used for direct estimation of the measurement error.
- The argument `pairs.min` defines a minimum number of pairs in a distance class. Variogram values are not recorded for a distance class (bin) with less than this number of pairs.
- The cloud values can be stored using `bin.cloud = T`. It allows the user to inspect the values of the variogram cloud for each bin.

Figure 3 shows the variogram clouds and the binned variograms obtained using the classical estimator (left) and the robust estimator (right).

The following commands can be used to reproduce the results:

```
> cloud1 <- variog(s100,option='cloud',max.dist=1)
> cloud2 <- variog(s100,option='cloud',estimator.type='robust', max.dist=1)
> bin1 <- variog(s100,uvec=seq(0,1,l=11))
> bin2 <- variog(s100,uvec=seq(0,1,l=11),estimator.type='robust')
> par(mfrow=c(2,2))
> plot(cloud1)
> plot(cloud2)
> plot(bin1)
> plot(bin2)
```

The objects `bin1` and `bin2` are lists whose the main components are: midpoints of the distance classes, variogram values and number of pairs of points for each distance class.

The objects `cloud` and `bin` are of the class `variogram`. Thus, the command `plot` calls the variogram plotting function `plot.variogram`. Binned variograms can be added to another plots using `lines` (which calls `lines.variogram`). This can be useful to compare different variogram estimators and variograms from different variables in the same plot.

The binned variogram can be influenced by outliers and/or pairs with large differences in values. In order to identify such anomalies the variogram cloud can be grouped by bins, and inspected. To save the cloud values grouped by bins, the option `bin.cloud = T` is used. For the example considered here:

```
> bin1 <- variog(s100,uvec=seq(0,1,l=11),bin.cloud=T)
> bin2 <- variog(s100,uvec=seq(0,1,l=11), estimator.type='robust',
                 bin.cloud=T)
```

Now the objects `bin1` and `bin2` each have an extra component, the variogram cloud grouped by bins. Figure 4 shows these cloud values for both estimators using box-plots. The commands to produce the plots are:

```
> par(mfrow=c(1,2))
> plot(bin1, bin.cloud=T)
> plot(bin2, bin.cloud=T)
```

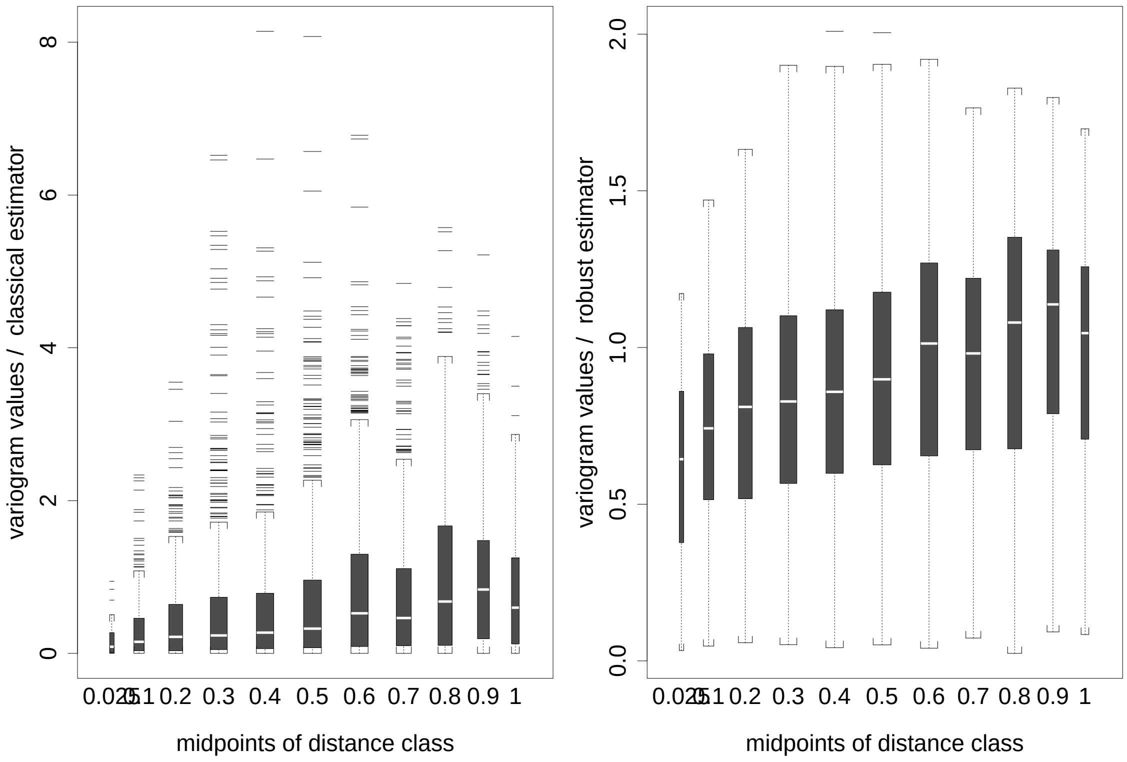

Figure 4: Variogram cloud values for binned variograms. Classical estimator (left) and robust estimator (right).

There are a few more arguments in the function including an option to transform the data using the Box-Cox transformation Box & Cox (1964) and to remove trends, which can be polynomials on the coordinates or a linear model on covariates provided.

A variogram model can be superimposed on a plot with an empirical variogram using the function `lines.variomodel`. Figure 5 shows the empirical binned variogram `bin1`, an exponential variogram model (thick line) with $nugget = 0, sill = 1, range = 0.3$ (i. e. $practical \quad range = 0.9$) and a kernel smoothed variogram. The commands to reproduce these results are:

```
> par(mfrow=c(1,1))
> bin1 <- variog(s100,uvec=seq(0,1,l=11))
> plot(bin1)
> lines.variomodel(list(nugget=0, cov.pars=c(1,0.3), max.dist=1,
                   cov.model='exp'), lwd=3)
> smooth <- variog(s100, option='smooth', max.dist=1, n.points=100)
> lines(smooth, type='l', lty=2)
> legend(0.4, 0.3, c('empirical','exponential model','smooth'),
         lty=c(1,1,2), lwd=c(1,3,1))
```

The function `lines.variomodel` is useful to superimpose and compare a fitted variogram with the empirical variogram. The fitted variogram is the result of a variogram parameter estimation procedure. For example the estimates can be found in an object of the class `variomodel`, returned by the function `likfit`. Other functions which also return a *variomodel* class object are `olsfit` and `wlsfit`. Variogram parameter estimation will be discussed in the next section.

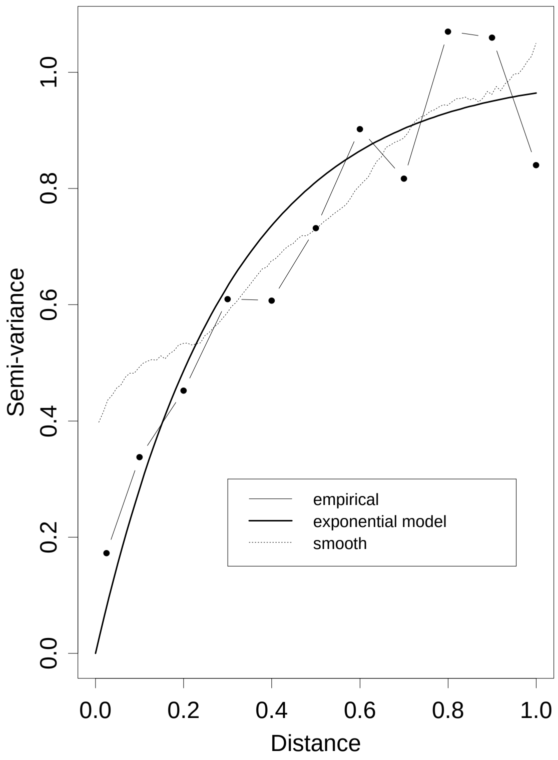

Figure 5: Binned, smoothed and theoretical variograms.

5.3 Parameter estimation

Estimation of variogram (or covariance) parameters can be performed using different methods (Zimmerman & Zimmerman, 1991). The function `likfit` returns maximum likelihood or restricted maximum likelihood estimates. Ordinary least squares estimates are obtained using the function `olsfit` and weighted least squares estimates by using `wlsfit`. All of these functions return objects of the class *variomodel* and the fitted models can be easily plotted. The commands below illustrate the usage of these functions. Variogram models fitted by different methods are plotted in Figure 6.

```
> ml <- likfit(s100,ini=c(0.5,0.5),fix.nugget=T)
> reml <- likfit(s100,ini=c(0.5,0.5),fix.nugget=T,method='RML')
> ols <- olsfit(bin1,ini=c(0.5,0.5),fix.nugget=T)
> wls <- wlsfit(bin1,ini=c(0.5,0.5),fix.nugget=T)
> plot(bin1)
> lines(ml,max.dist=1)
> lines(reml,lwd=2,max.dist=1)
> lines(ols,lty=2,max.dist=1)
> lines(wls,lty=2,lwd=2,max.dist=1)
> legend(0.6,0.3,legend=c('ML','REML','OLS','WLS'), lty=c(1,1,2,2),
         lwd=c(1,2,1,2))
```

As for the empirical variograms, these functions also include options for data transformation and trend removal. Bayesian estimates of the parameters can also be obtained, using the function `krige.bayes`. The use of this function will be described later in Section 5.5.

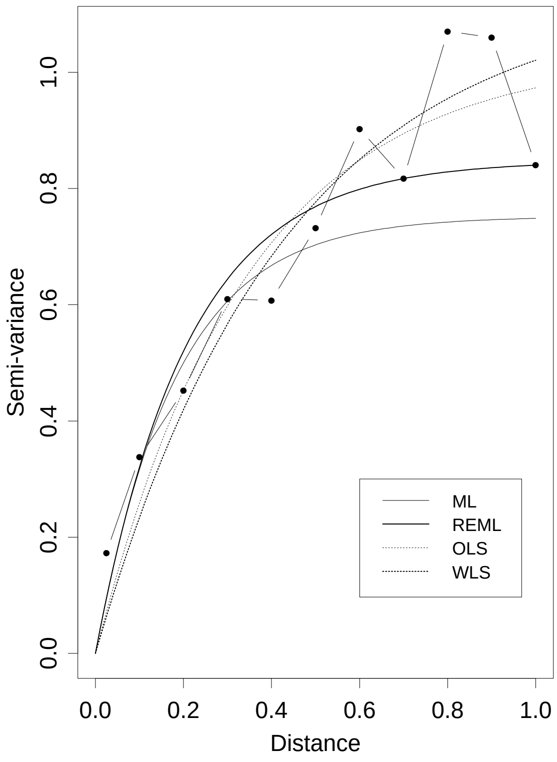

Figure 6: Empirical and estimated variograms.

The function `variog.env` computes envelopes for an empirical binned variogram. The envelopes are based on simulations of Gaussian random fields with variogram parameters provided by the user. Typically the values for these parameters are given by estimates obtained using one of the estimation methods. The number of values used to contruct the envelope depends on the specified level, with the default being 99. For each simulation the empirical variogram is computed. Lower and upper envelopes limits take the minimum and maximum values of the variograms for the simulations within each bin class. The envelopes can be added to a variogram plot using the argument `envelope.obj`. Figure 7 shows the binned variogram with the associated envelopes. The `geoR/geoS` commands are:

```
> env <- variog.env(bin1,s100$coords,model.pars=wls)
> plot(bin1,envelope=env)
```

Another way to inspect the uncertainty in the model parameter estimates is looking at the profile likelihoods plots. Figure 8 show the profiles likelihoods for the sill parameter σ^2 (left), the range parameter ϕ (middle) and the 2-D profile likelihood for these two covariance parameters. The profile likelihoods for each parameter are contructing by maximising the likelihood function for the remaning parameters, each fixed value of the parameter of interest. The two lines represent in the 1-D profile plots are the 90% and 95% quantiles of a $\frac{1}{2}\chi^2(1)$-distribution.

The commands used are:

```
> prof <- proflik(ml, geodata=s100, sill.val=seq(.48, 2, l=21),
        range.val=seq(.1,.52, l=21))
> plot(prof, nlevels=16)
```

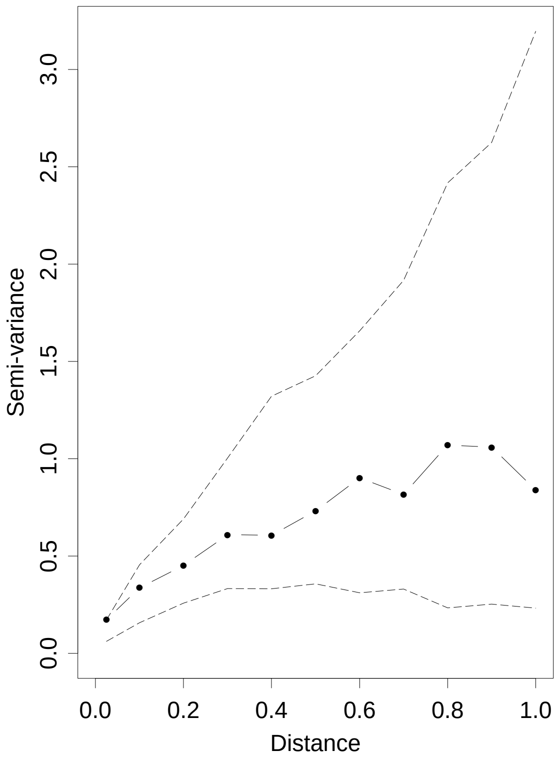

Figure 7: Empirical variogram with envelopes.

5.4 Conventional kriging

Consider now a set of locations chosen for prediction of the variable. Data and prediction locations are plotted in Figure 9. The commands to define locations and to produce Figure 9 are:

```
> loci <- matrix(c(0.2,0.6,0.2,1.1,0.2,0.3,1.0,1.1), ncol=2)
> plot(s100\$coords, xlim=c(0,1.2), ylim=c(0,1.2))
> text(loci, as.character(1:4))                # when using R
> text(loci[,1], loci[,2], as.character(1:4))  # when using S-PLUS
> segments(c(0,0,0,1),c(0,0,1,0),c(1,0,1,1),c(0,1,1,1),lty=2)
```

Conventional geostatistical methods usually estimate the covariance parameters and then plug-in the estimates to perform predictions, as if the estimates were the truth. The function `krige.conv` performs several types of conventional kriging: *simple kriging*, *ordinary kriging*, *kriging with external trend* and *universal kriging*. The command below illustrates the usage of the function `krige.conv` to obtain ordinary kriging (default option) estimates, at the locations indicated on Figure 9. The estimates of the covariance parameters obtained by weighted least squares are used.

```
> kc <- krige.conv(s100,locations=loci,
                   krige = krige.control(cov.pars=wls\$cov.pars))
```

In this example ordinary kriging predicted values and variances are returned. Section 6 and comments included in the body of the function `krige.conv` explains how to use other options of this function to perform other types of kriging.

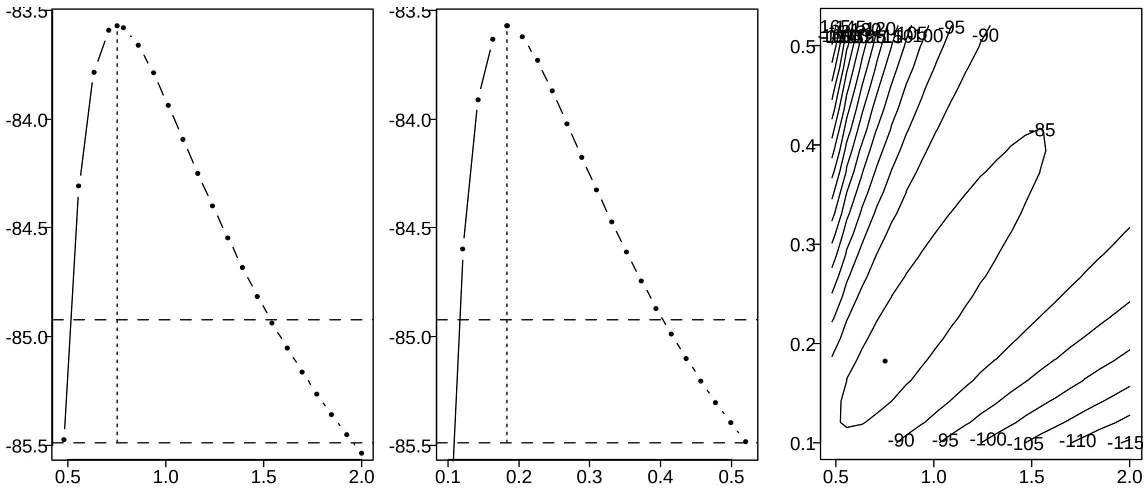

Figure 8: Profile likelihoods for covariance parameters. Left: σ^2, middle: ϕ and right: 2-D profile.

5.5 Bayesian prediction

In Bayesian prediction, parameter uncertainty is taken into account. The function `krige.bayes` is designed to perform Bayesian inference for a Gaussian geostatistical model. The function returns both posterior distributions for the model parameters, and predictive distributions for the variable at prediction locations. Several choices of priors are available. Different levels of uncertainty can be specified, considering some of the model parameters as fixed. A description of the methods implemented in `geoR/geoS` can be found in Ribeiro Jr & Diggle (1999).

For the data-set considered here, the command:

```
> bsp <- krige.bayes(s100, loc=loci, prior =
                    prior.control(range.discrete = c(seq(0,1,l=121))
```

is used to perform parameter estimation and prediction, taking into account the uncertainty in all of the model parameters (default option). Samples from posterior and predictive distributions are returned by the function. Posteriors and predictive distributions can be approximated using non-parametric density estimation.

For the predictions the function returns posterior means at prediction locations as the default option. Alternative estimates, e. g. quantiles, can be chosen instead using the argument `estimator`. This argument can also be used to compute cumulative probabilities up to a certain threshold.

For uncertainty only in the mean and sill parameters, i. e. considering the correlation parameter as fixed, and plugging-in a value of the correlation parameter, the function call is:

```
> bs <- krige.bayes(s100, loc=loci, prior = prior.control(
                     range.prior='fixed', range=0.3))
```

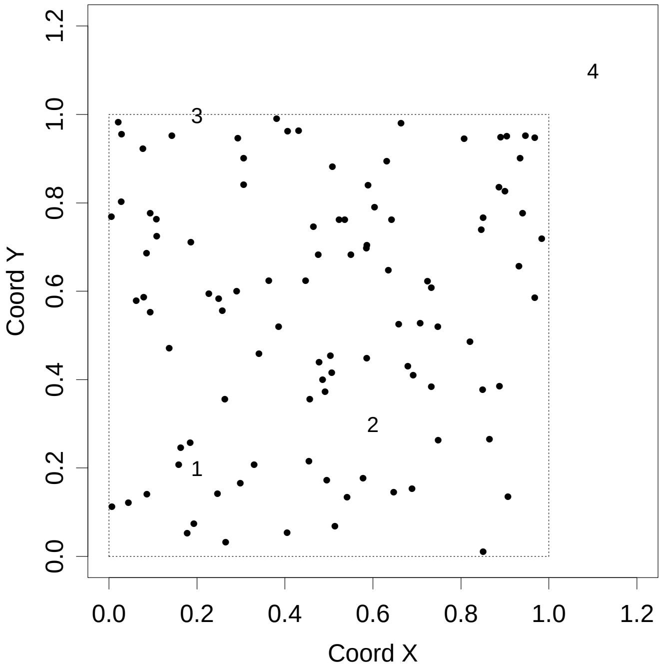

Figure 9: Data locations (•) and the four prediction locations (numbers).

In this example, posterior and predictive distributions are standard probability distributions and sampling from them is no longer necessary.

For uncertainty only in the mean parameter, i. e. considering the sill and the correlation parameter as fixed, three choices for the prior are available: 'normal', 'flat' (default option) and 'degenerate'. The second one (option `beta.prior = 'flat'`) produces predicted values and variances which coincide with ordinary kriging results. Using the option `beta.prior = 'flat'` the results coincide with simple kriging. For all of these cases, sill and range values must be provided. The commands below illustrate the options for the prior distribution. Weighted least squares estimates of the covariance parameters are used.

```
> b <- krige.bayes(s100, loc = loci, prior = prior.control(
                   range.prior = 'fixed', range = wls\$cov.pars[2],
                   sill.prior = 'fixed', sill = wls\$cov.pars[1]))

> bn <- krige.bayes(s100, loc = loci, prior = prior.control(
                    range.prior = 'fixed', range = wls\$cov.pars[1],
                    sill.prior = 'fixed', sill = wls\$cov.pars[1],
                    beta.prior = 'normal',beta = 0, beta.var = .5))

> bd <- krige.bayes(s100, loc=loci, prior = prior.control(
                    range.prior = 'fixed', range = wls\$cov.pars[2],
                    sill.prior = 'fixed', sill = wls\$cov.pars[1],
```

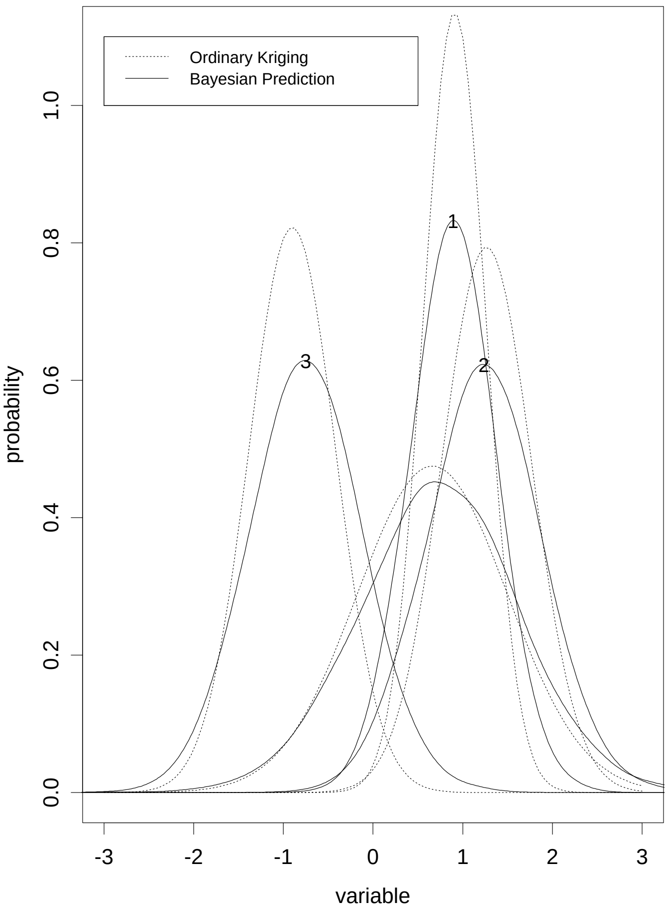

Figure 10: Estimated distributions at prediction locations: from the ordinary kriging method (dashed line) and the Bayesian predictive distribution assuming all parameters unknown.

```
beta.prior = 'fixed', beta=0))
```

The prediction locations are indicated by numbers in Figure 9. Some prediction results for these locations are illustrated in Figure 10. The dashed lines represents ordinary kriging results, assuming Gaussianity. Therefore, the predictive distributions are normal with parameters recorded in the object `krigeconv` above. The means are given by the ordinary kriging predicted values, and the variances are given by the ordinary kriging variances. The full lines are distributions obtained by density estimation from the samples of the predictive distributions. The samples are components of the object `bsp` defined above. The plot was produced using the standard S-PLUS functions `density, dnorm` and `plot`.

5.6 Gaussian random fields

Simulations from Gaussian random fields can be obtained using the function `grf`. This function has several options, see Section 6.7 for more details. The data-set considered so far was generated using this function. Another example, now with 10 simulations and sampling locations on a regular grid, is generated in *S-PLUS* using the commands:

```
> .Random.seed <- c(37,22,26,1,45,3,60,52,36,6,8,3)
> s121 <- grf(121,grid='reg',nsim=10,cov.pars=c(1,.3))
```

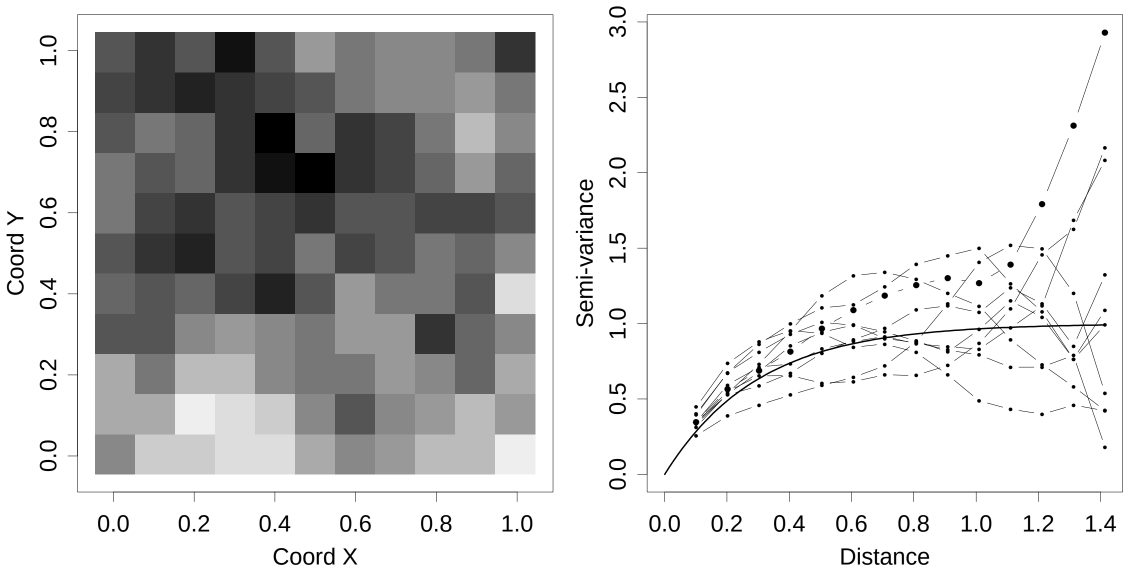

Figure 11: A simulated field generated by `grf` and variograms for several simulations.

As for the `s100` data, this simulated data set is available in the `geoR` distribution as well as in the `geoR/geoS` home-page.

An image plot of the 8^{th} simulation and the variograms for the 10 simulations are shown in Figure 11. The plots were produced using the commands:

```
> image.grf(s121, sim=8)
> plot(s121)
```

The output of the function `grf` is an object of the class *grf*. Some associated functions are `plot.grf`, `image.grf`, `wireframe.grf` and `levelplot.grf`. The last two are only available for *S-PLUS*.

6 Description of the functions

This section describes the main features of the geoS functions. Standard S-PLUS functions are described by Becker et al. (1988) and Venables & Ripley (1997). The latter also provides complementary functions and libraries. Some of the geoS auxiliary functions, i. e. functions internally called by other functions, are not described here.

Most of the functions have comments on their initial lines describing the function purpose and the meaning of each of the function parameters. The majority of the functions return an object in a form of a list. Classes are usually assigned to the objects created by the functions in order to facilitate graphical outputs.

6.1 Data preparation

In order to perform a geostatistical analysis it is necessary to have a data set with data locations (Euclidean coordinates) and the measurement values at each location. The geoS functions usually read these data as: 1) an $n \times 2$ matrix of data locations (n is the number of data) and 2) a vector of data values at each location. If more than one variable or several simulations are available, the second object is a matrix where each column corresponds to one variable or simulation. Data can be stored as a matrix, data-frame or list. The last format is preferable, with a matrix with data location coordinates as the first component and the data as the second.

Coordinates for regular grids can be created using standard S-PLUS commands such as `expand.grid`, `rep` and `seq`. The auxiliary function `grid.grf`, internally called by the simulation function `grf`, can also be used to generate regular grids.

1. `grid.grf(grid, n, xgrid, ygrid, nx, ny)`.

   This function generates a regular rectangular grid over a 2D region. Typically called by `grf` (see Section 6.7) but can also be used to create regular grids for other functions such as the ones which performs kriging (Section 6.6).

2. `polygrid(xgrid, ygrid, poly)`.

   If the study area is not rectangular, this function creates a regular grid inside a polygonal area defined by the argument `poly`. It is useful to create grids with locations to be predicted by kriging functions (see Section 6.6).

6.2 Descriptive analysis

Most of the descriptive analysis can be done using standard S-PLUS commands. The *S-PLUS Spatial Stats Manual* (Mathsoft, 1996) provides some examples of descriptive analysis for geostatistical data. The functions implemented in geoR/geoS are listed here.

1. ```
   plot.geodata(geodata, coords = geodata\$coords, data = geodata\$data,
                trend = "cte", lambda = 1, col.data = 1, window = F, ...).
   ```
   ```

This function generates a graphical output with four plots: data locations, a tri-dimensional plot with data locations and values, data plotted against coordinates X and Y.

2. `points.geodata(geodata, coords = geodata$coords, data = geodata$data,`
```
                col.data = 1, pt.sizes = c("proportional", "quartiles",
                "deciles", "equal"), cex.min, cex.max, pch.seq,
                col.seq, ...)
```

   Data locations are added to a plot. The points indicating data locations can have different colors, symbols or sizes which can be proportional to the value of the observed data at each location.

6.3 Empirical variograms

The generic function used to obtain empirical variograms is the function `variog`. The results of the variogram estimation can be presented in three different ways:

- as a variogram cloud, i. e., $\binom{n}{2}$ pairs $(h, \hat{\gamma}(h))$, where n is the number of data points,
- as a binned variogram, which is a grouped version of the variogram cloud or
- as a smoothed variogram, i. e. a smoothed version of the variogram cloud.

1. `variog(geodata, coords = geodata\$coords, data = geodata\$data,`
```
     lambda = 1, trend = "cte", option = c("bin", "cloud", "smooth"),
     estimator.type = c("classical", "robust"), uvec = "default",
     nugget.tolerance = 0, max.dist = "all", pairs.min = NULL,
     bin.cloud = F, n.points = 50, ...).
```

   This function requires two basic inputs: a matrix with data coordinates and a data vector or matrix. These can be elements of a list provided in the argument `geodata` or provided individually. An object (list) of the class `variogram` is returned. The main output components are the midpoints of the distance classes, the associated variogram values and the number of pairs. If `bin.option = T`, variogram cloud values grouped by bins are also returned.

2. `rfm.bin(cloud, l = 15, uvec = "default", nugget.tolerance = 0,`
```
estimator.type = c("classical", "robust"), bin.cloud = F) .
```

   This function produces a binned variogram from an object returned by `variog` with `option='cloud'`. The default produces bins for distances in an equally spaced sequence of length `l+1`. Other sequences and bin sizes can be defined using the argument `uvec`. If the object in the argument `cloud` has a class, the result has the same class, otherwise the class `variogram` is assigned to the object. The output includes the midpoints of the distance classes, the associated variogram values and the number of pairs. If `bin.option = T` the cloud values grouped by bins are also returned.

3. `variog.env(obj.variog, coords, model.pars, trend = "cte", nsim = 99, messages.screen = T)`.

   This function computes envelopes for a binned variogram. The user must provide an object with a binned variogram computed using `variog`, data coordinates and information about model parameters. Tipically the input for the argument `model.pars` is an object containing the output of a variogram estimation function: `olsfit`, `wlsfit` or `likfit`. The argument `nsim` specifies the number of simulations from a Gaussian random field generated to compute the envelopes. The envelopes are the pointwise maximum and minimum values of the binned variograms computed from the simulations.

Remark:
The function `variog` is able to compute variograms for data presented either as a vector or as a matrix, where each column represents one variable or simulation. For the latter the result is a list where the first component is a vector of distances and the second is a matrix with variogram values and as many columns as the original data matrix. This is useful to compute variograms for several simulations or for multivariate data, whenever all of the simulations or variables have the same locations. However, the option `bin.option = T` is allowed only if the variogram is being computed for just one variable (or simulation).

6.4 Variogram plots

The class `variogram` is assigned to the objects produced by the function `variog`. Specific functions called by the S-PLUS generic functions `plot` and `lines` were written in order to produce graphs for `variog` outputs. Therefore the functions can be called using simply the commands `plot` or `lines`, with the required arguments.

1. `plot.variogram(obj, max.dist = max(obj$u), ylim = "default", type = "b", scaled = F, var.lines = F, envelope.obj = NULL, bin.cloud = F, ...)`.

   This function produces a plot for an object of the class `variogram`. If several variables or simulations are provided, the empirical variograms for all of them appear in the same plot. If `bin.option = T` box-plots for the variogram cloud values, grouped by bins, are plotted instead.

2. `lines.variogram(obj, dmax = max(obj$u), type = "o", scaled = F, ...)`.

   This function adds lines with empirical variogram(s) to a current plot.

3. `lines.variomodel(obj, dmax = obj$dmax, length = 100, scaled = F, ...)`.

   This function adds lines with a theoretical variogram model to a plot. The theoretical variogram models available are: linear, powered exponential, wave (or hole effect), Gaussian, Matérn class (which includes the exponential) and spherical.

6.5 Variogram Fitting

In geostatistical analysis the variogram model is usually fitted using one of the procedures described below.

- "By eye", i. e. plotting theoretical models over the empirical variogram.
  The parameters are arbitrarily chosen by the user. This can be done with `geoS` using the function `lines.variomodel` to superimpose theoretical models on a plot of the empirical variogram
- Using the empirical variogram estimates returned by `variog` to fit a model. This includes methods like *ordinary least squares (OLS)*, *weighted least squares (WLS)* and *quasi-likelihood (QL)*. Notice that in this case the 'input' for variogram parameter estimation is the binned variogram or the variogram cloud, and not the original data.
- Assuming a parametric model for the original data and running likelihood-based methods. For Gaussian random fields, *maximum likelihood (ML)* and *restricted maximum likelihood (REML)* can be used. For a more general class of processes and the associated inference methods see Diggle et al. (1998).

1. `olsfit(vario, ini, cov.model = c("exponential", "matern", "gaussian", "spherical", "circular", "cubic", "wave", "powered.exponential", "cauchy", "gneiting", "gneiting.matern", "pure.nugget"), kappa = NULL, simul.number = NULL, max.dist = "all", minimisation.function = "nlm", lower = 0, fix.nugget = F, nugget = 0, print.level = 0)` .

   The function `olsfit` performs the OLS fit on either the binned variogram or the variogram cloud. The auxiliary function `loss.olsvario` is called by `olsfit` during the estimation process.

2. `wlsfit(vario, ini, cov.model = c("exponential", "matern", "gaussian", "spherical", "circular", "cubic", "wave", "powered.exponential", "cauchy", "gneiting", "gneiting.matern", "pure.nugget"), kappa = NULL, simul.number = NULL, max.dist = "all", minimisation.function = "nlm", lower = 0, fix.nugget = F, nugget = 0, weight = c("npairs", "cressie"), print.level = 0)` .

   The function `wlsfit` performs the WLS fit on either the binned variogram or the variogram cloud. The auxiliary function `loss.wlsvario` is called by `wlsfit` during the estimation process.

   The argument `weight` defines whether this function performs the WLS fit as suggested by Barry et al. (1997) or by Cressie (1985).

3. `likfit(geodata, coords = geodata$coords, data = geodata$data, ini, trend = "cte", fix.nugget = FALSE, nugget = 0, method = "ML", cov.model = c("matern", "exponential", "gaussian", "spherical", "circular", "cubic", "wave", "powered.exponential", "cauchy", "gneiting", "gneiting.matern", "pure.nugget"), kappa = 0.5, fix.lambda = TRUE, lambda = 1, predicted = FALSE, residuals = FALSE, messages.screen = TRUE, automatic.refit = TRUE, min.func = c("nlmP", "nlm"), range.limits = "default", ...)` .

This function includes options for both maximum likelihood (ML) and restricted maximum likelihood (REML) estimation of the variogram parameters. Different variogram models can be chosen and mean parameter estimates are also returned. The mean can be given by a constant, a first or second order polynomial or external trend. This function also includes options to fix the nugget effect parameter and to plot likelihood surfaces. It returns an object of class `variomodel`, a list with estimates and detailed information about the estimation results.

4. `proflik.*(***)`

   These functions returns the (sometimes negative) values of the maximum or restricted maximum likelihood. Tipically they are called by `likfit` or `proflik` during parameter estimation the estimation process.

5. `(obj.likfit, geodata, coords = geodata$coords, data = geodata$data, nugget.values, nugget.rel.values, sill.values, range.values, lambda.values, sillrange.values = T, sillnugget.values = T, rangenugget.values = T, sillnugget.rel.values = F, rangenugget.rel.values = F, silllambda.values = F, rangelambda.values = T, nuggetlambda.values = F, nugget.rellambda.values = F, uni.only = F, bi.only = F) .`

   This function computes profile likelihood surfaces for the variogram (covariance) parameters.

6. `plot.proflik(obj.proflik, pages = c("user", "one", "two"), uni.only = F, bi.only = F, type.bi = c("contour", "persp"), conf.int = c(0.9, 0.95), yaxis.lims = c("conf.int", "as.computed"), by.col = T, log.scale = F, par.mar.persp = c(0, 0, 0, 0), ...) .`

   This function plots profile likelihoods computed by `proflik`.

6.6 Kriging

1. Function `ksline(ksline) function (geodata, coords = geodata$coords, data = geodata$data, locations, cov.pars = stop("covariance parameters (sigmasq and phi) needed"), nugget = 0, micro.scale = 0, cov.model = c("exponential", "matern", "gaussian", "spherical", "circular", "cubic", "wave", "powered.exponential", "cauchy", "gneiting", "gneiting.matern", "pure.nugget"), kappa = 0.5, lambda = 1, n.samples.backtransform = 500, m0 = "ok", nwin = "full", trend = 1, d = 2, ktedata = NULL, ktelocations = NULL, aniso.pars = F, exact = F, signal = F, messages.screen = T, dist.epsilon = 1e-10) .`

   This function performs simple (SK), ordinary (OK), universal (KT) or external trend (KTE) kriging. An option to specify either a global or a moving neighbourhood is available.

   Although from the model perspective there is no difference between KT (universal or trend kriging) and KTE (kriging with external trend), they are treated separately because the trend matrix for KT is automatically built (for constant, first and second order polynomial trend) whereas for KTE the external trend variable(s) must be provided.

If $exact = T$, the estimated values coincide with data values at data locations. If there is no nugget effect the results are the same for both $exact = T$ or $exact = T$, otherwise the former predicts the signal process and the latter consider that there is no measurement error at data locations.

This function must be incorporated in the next, `krige.conv` in a near future and made obsolete.

2. `krige.conv(geodata, coords = geodata$coords, data = geodata$data, locations, trend.d = "cte", trend.l = "cte", type.krige = "ok", beta = NULL, cov.model = c("exponential", "matern", "gaussian", "spherical", "circular", "cubic", "wave", "powered.exponential", "cauchy", "gneiting", "gneiting.matern", "pure.nugget"), kappa = 0.5, cov.pars = stop("covariance parameters (sigmasq and phi) needed"), nugget = 0, lambda = 1, n.samples.backtransform = 500, aniso.pars = F, signal = F, exact = F, messages.screen = T, ...)` .

   This function is an updated version of `ksline`, but the currently implementation doesn't allow moving neighbourhoods. It has options for several types of kriging: (SK, OK, KTE, UK). Several variogram models are implemented, see the argument `cov.model` above. Anisotropic variogram models are allowed. Nested models, where the variogram model is a linear combination of basic models, are allowed and for this case the argument `cov.pars` is a matrix. The covariance models available are: Matérn class, spherical, Gaussian, wave (hole effect) and power. Others can be easily incorporated using the auxiliary function `cov.spatial`.

3. `krige.bayes(geodata, coords = geodata$coords, data = geodata$data, locations = "no", trend.d = "cte", trend.l = "cte", beta.prior = "flat", beta = NULL, beta.var = NULL, sill.prior = T, sill = NULL, range = NULL, range.prior = c("squared.reciprocal", "reciprocal", "uniform", "exponential", "fixed"), exponential.prior.par = 1, range.limits = "default", n.range = 51, cov.model = c("exponential", "matern", "gaussian", "spherical", "circular", "cubic", "wave", "powered.exponential", "cauchy", "gneiting", "gneiting.matern", "pure.nugget"), kappa = 0.5, nugget.prior = c("fixed", "uniform"), nugget = 0, lambda = 1, aniso.pars = F, n.posterior = 1000, distribution = T, mean.estimator = T, quantile.estimator = F, probability.estimator = F, messages.screen = T, signal = F, info.for.prediction = T, moments = T)` .

   This function performs prediction under the Bayesian approach, assuming a Gaussian model for the observations. Conventional kriging results can also be obtained as a particular case. Uncertainty in the mean and covariance parameters can be taken into account.

   If only the mean parameters are considered unknown then three priors can be used:

   - a 'degenerate' prior, i. e. the user must provide the mean parameter(s) (the results equals the simple kriging ones);

- a 'flat' prior where no input is required for the mean parameter(s) (the results equal the ordinary kriging ones);
- a normal prior where the user must provide the mean and the variance (or covariance matrix) of the prior distribution.

Uncertainty in the mean and scale (sill) parameters produces estimates which coincide with ordinary kriging but with different variances. The default option considers also uncertainty in the range parameter, i. e. all the parameters are considered as random quantities.

For the default option the output provides samples from the posterior and predictive distributions, since analytical results cannot be found. Notice that allowing uncertainty in all the parameters, Bayesian estimates can be obtained for the variogram parameters. This is an alternative estimation method to those discussed in Section6.3. This function allows either isotropic and anisotropic covariance models. (The current implementation doesn't allow models where the covariance function is a linear combination of basic covariance models, i. e. nested models).

4. `cov.spatial(obj, cov.model = c("exponential", "matern", "gaussian", "spherical", "circular", "cubic", "wave", "powered.exponential", "cauchy", "gneiting", "gneiting.matern", "pure.nugget"), kappa, cov.pars = stop("no cov.pars argument provived"))` .

   This is an auxiliary function called by several others. The input is a vector with distance values and the output is a vector with corresponding covariance values, computed according to the covariance model chosen.

5. `varcov.spatial(obj, cov.model = c("exponential", "matern", "gaussian", "spherical", "circular", "cubic", "wave", "powered.exponential", "cauchy", "gneiting", "gneiting.matern", "pure.nugget"), kappa, cov.pars = stop("no cov.pars argument provived")) NULL > args(varcov.spatial) function (coords = NULL, dists.lowertri = NULL, cov.model = c("exponential", "matern", "gaussian", "spherical", "circular", "cubic", "wave", "powered.exponential", "cauchy", "gneiting", "gneiting.matern", "pure.nugget"), kappa = NULL, nugget = 0, cov.pars = stop("no cov.pars argument"), inv = F, det = F, func.inv = "chol", scaled = F, only.decomposition = F, sqrt.inv = F, test.decomp = F)` .

   This auxiliary function builds a covariance matrix for a given set of coordinates (typically data-locations), using the covariance model provided. If $inv = T$ the inverse of the covariance matrix is returned instead. If $det = T$ the squre root of the logarithm of the determinant of the covariance matrix is also returned.

6.7 Simulation of Gaussian random fields

1. `grf(n, grid = "irreg", nx = round(sqrt(n)), ny = round(sqrt(n)), nsim = 1, xlims = c(0, 1), ylims = c(0, 1), cov.model = c("exponential", "matern", "gaussian", "spherical", "circular", "cubic", "wave",`

```
"powered.exponential", "cauchy", "gneiting", "gneiting.matern",
"pure.nugget"), cov.pars = stop("covariance parameters (sigmasq
and phi) needed"), kappa = 0.5, nugget = 0, lambda = 1, method =
c("cholesky", "svd", "eigen", "circular.embedding"), plotgrid = F,
plotvario = F, messages.screen = T, ...)  .
```

This function provides simulations of a Gaussian random field. The main options are listed below.

- Irregular data locations (following a Poisson process) as default option. The argument `grid` allows specification of a regular grid or a fixed set of locations given by the user.
- Several simulations at the same locations can be obtained using the argument *nsim*.
- Several variogram models, see the argument `cov.model`.
- Numerical methods: Cholesky decomposition or singular value decomposition.
- An option for automatic plotting of the data locations.
- An option for automatic plotting of the theoretical variogram model with the empirical variogram(s) for the simulation(s),
- An option to perform parametric estimation of the variogram parameters for the simulation(s).
- option fir different algorithms.

The auxiliary functions `messa.grf`, `grid.grf`, `grf.aux*` and `grfclass` are called by the main function `grf`.

2. `plot.grf(obj, grid = F, model = T, ylim = "default", ...)`.

   This function plots the empirical variograms for simulated data generated by `grf`. The variogram model used for simulation is also plotted by default. If several simulations are generated the function plots all of the empirical variogram in the same graph.

3. `image.grf(obj, sim.number = 1,...)`
   `wireframe.grf(obj, sim.number = 1, ...)`
   `persp.grf(obj, sim.number = 1,...)`
   `levelplot.grf(obj, sim.number = 1,...)`
   . These functions produces gray or colour scaled image and perspective plots for regularly spaced data of the class `grf`, respectively.

Acknowledgement

This report was written during the first author's PhD program at the Department of Mathematics and Statistics – Lancaster University, supported by the grant BEX–1676/96-2 — CAPES/BRASIL.

We thank many users who have reported usage and/or provided comments/suggestions for the package/library.

References

Barry, J., Crowder, M. & Diggle, P. (1997). Parametric estimation of the variogram, *Technical Report ST-97-06*, Dept. Maths and Stats, Lancaster University, Lancaster, UK.

Becker, R., Chambers, J. & Wilks, A. (1988). *The new S language: a programming environment ofr data analysis and graphics*, Wadsworth and Brooks Cole.

Box, G. & Cox, D. (1964). An analysis of transformations (with discussion), *Journal of Royal Statistical Society, B* .

Cressie, N. (1985). Fitting variogram models by weighted least squares, *Mathematical Geology* **47**: 563–586.

Cressie, N. (1993). *Statistics for Spatial Data - revised edition*, Wiley, New York.

Cressie, N. & Hawkins, D. (1980). Robust estimation of the variogram, *Mathematical Geology* **12**: 115–125.

Diggle, P. & Ribeiro Jr, P. (2000). Model-based geostatistics, XIV Sinape (Simpósio Nacional de Probabiulidade e Estatística, Caxambu, MG, Brasil.

Diggle, P., Tawn, J. & Moyeed, R. (1998). Model based geostatistics (with discussion), *Applied Statistics* **47**(3): 299–350.

Goovaerts, P. (1997). *Geostatistics for Natural Resources Evaluation*, Oxford University Press, New York.

Journel, A. & Huijbregts, C. (1978). *Mining Geostatistics*, Academic Press, London.

Matheron, G. (1963). Priciples of geostatistics, *Economic geology* **58**: 1246–1266.

Mathsoft (1993). *S-PLUS User's Manual. Version 3.4.*, Mathsoft Inc.

Mathsoft (1996). *S+SPATIALSTATS User's Manual. Version 1.0.*, Mathsoft Inc.

Ribeiro Jr, P. & Diggle, P. (1999). Bayesian inference in gaussian model-based geostatistics, *Technical Report ST-99-08*, Dept. of Maths and Stats, Lancaster University, Lancaster, UK.

Venables, W. & Ripley, B. (1997). *Modern applied statistics using S-PLUS*, 2 edn, Springer.

Zimmerman, D. & Zimmerman, M. (1991). A comparison of spatial semivariogram estimators and corresponding kriging predictors, *Technometrics* **33**(1): 77–91.