NISS

Design Interface: A graphical tool for analyzing and constructing spatial designs

Douglas Nychka, Nancy Saltzman,
and Andrew Royle

Technical Report Number 42
April, 1996

National Institute of Statistical Sciences
19 T. W. Alexander Drive
PO Box 14006
Research Triangle Park, NC 27709-4006
www.niss.org

Design Interface: A graphical tool for analyzing and constructing spatial designs

Douglas Nychka Nancy Saltzman
Andrew Royle
National Institute of Statistical Sciences*

April 9, 1996

Abstract

Design Interface (*Di*) is a graphical, interactive tool for spatial design. Its purpose is to facilitate construction and comparison of spatial designs by visualization of the designs themselves and statistics that measure their performance. This package has been implemented as functions within the S programming environment and leverages the wealth of statistical and graphical tools already available in Splus. *Di* also makes use of object-oriented programming ideas to provide common summary and plot functions for different types of design data sets. User interaction is simplified by the use of menu boxes and other graphical inputs and the package is structured so that new features may be easily incorporated into the program. This manual explains how to use *Di* , gives some examples and also explains how to add new features.

*Contact Address: National Institute of Statistical Sciences, P.O. Box 14162, Research Triangle Park, NC 27709-4162. Funding for this research was provided by the US Environmental Protection Agency through grant CR#819638, and by the National Science Foundation through grant DMS 9208758, both to the National Institute of Statistical Sciences.

1 Introduction

A basic problem associated with spatial data is to predict the value of a spatial variable at locations where it is not measured. In environmental applications the concentration of a pollutant or other physical or chemical variables are often made at a limited set of instrument locations. These locations that form the monitoring network will be referred to as the *design* in this manual. Also, we will use *spatial field* to refer to the continuous spatial surface that would result from an extensive sampling of the variable. The goal then is to determine the locations of measurements to maximize the amount of information about the spatial field.

The motivation for creating this set of programs was a need for a flexible graphical tool to evaluate spatial designs. Although several researchers have created algorithms to calculate optimal designs under different criteria there is a practical need to determine a given design's robustness to other measures of performance and to understand the effect of adding and deleting points from an "optimal" set of locations. Also in many applications spatial networks of monitors are already in place and it is of interest to analyze the effect of thinning such networks or moving locations.

2 Model for spatial fields

Before discussing the details of *Di* is useful to define the underlying statistical model to evaluate different designs. The properties of a design are found under the assumption that the spatial field can be modeled as a random surface with an isotropic covariance function. Let $Z(\mathbf{x})$ denote the value of the field at location $\mathbf{x} = (x_1, x_2)$. We assume that $E(Z(\mathbf{x})) = 0$ Also let $COV(Z(\mathbf{x}), Z(\mathbf{x}')) = k(\mathbf{x}, \mathbf{x}')$ denote the covariance function. The simplest use of *Di* assumes that the covariance function is isotropic.

$$k(\mathbf{x}, \mathbf{x}') = h(\|\mathbf{x} - \mathbf{x}'\|)$$

where $\|\mathbf{x} - \mathbf{x}'\|$ is the Euclidean (or great circle) distance between two points.

The key feature of a network is its ability to predict values of the field at locations off of the design points. Given N network locations $\{\mathbf{x}_j\}$ for $1 \leq j \leq N$, let $z_j = Z(\mathbf{x}_j)$ and let K be the $N \times N$ covariance matrix $K_{ij} = COV(z_i, z_j)$. To estimate the field at a location off of the network it is reasonable to use the best linear estimate. Let $\gamma_j = k(\mathbf{x}, \mathbf{x}_j)$ and $\hat{Z}(\mathbf{x}) = \lambda^T \mathbf{z}$ where $\lambda = K^{-1}\gamma$. Simple linear statistics can be used to derive the variance of this estimate.

$$VAR\left(\hat{Z}(\mathbf{x}) - Z(\mathbf{x})\right) = k(\mathbf{x}, \mathbf{x}) - 2\lambda^T\gamma + \lambda^T K^{-1}\lambda$$

or simplifying

$$VAR\left(\hat{Z}(\mathbf{x}) - Z(\mathbf{x})\right) = k(\mathbf{x}, \mathbf{x}_j) - \gamma^T K^{-1}\gamma$$

.

This variance formula is the basis for quantifying the performance of designs in *Di* .

3 Basic Structure of *Di*

Our choice was to implement *Di* by adding a small set of extra functions to the Splus statistical package. This simplified the amount of programming and also makes it easier to add new features. The basic steps to use *Di* are to first start Splus, attach the set of *Di* functions and then to edit a design. In order to have these new designs returned they must be saved. After one quits the edit function the saved designs are returned as an S data set (a network catalog object). This output data set is set up so that it can be used as the input to the edit function. In this way one can keep modifying a design and/or build up a sequence of different designs to compare. During the editing process one is given menu options to summarize or plot designs. Because these are just S functions applied to the current design they can also be reproduced outside the editor.

3.1 Sample design analysis

To make this process concrete here is an example. (Comments are after the # signs) The reader is referred to Section 5 for a script of some possible interactive steps inside the editor.

```
% Splus        # in UNIX start Splus
#  following are Splus commands
> attach("/zork/Di/.Data") # In place of /zork/Di give the full UNIX path name
                           # for the  DI directory on your system
> square.nw                # square.nw is a sample 5X5 grid of points
                           # just typing its name gives a summary of
                           # the design
> edit.nw(square.nw) -> ex.out  # use the design editor to view and modify design
> edit.nw( ex.out)-> ex.final # edit the output from the previous step
> summary( ex.final)        #summarize the final designs
```

3.2 Editing the design

The design editing function (`edit.nw`) provides interaction by a menu box that lists the different types of actions that can be taken. Some of these actions are to modify the current design while others summarize, plot or save the designs that have been created. Where appropriate secondary menus are generated to control the options for the plots and summaries. Some points about using `edit.nw`.

- Clicking on a button only *selects* the action you must then click on the OK button to initiate the action. This logic (?) is the same as for PC menu windows and is a constraint of the Motif dialog boxes.

- While using the design editor useful instructions and comments will be printed in the main S plus window.

- In the design editor one must save the design before it will be included in the design summary.

- Any plot window provided by *Di* can be resized or moved using the usual operations for Motif windows and Splus graphics windows. For example you

can print the window by clicking on the print option from the graph menu.

3.3 Design data sets

The input to the edit function is a network object. This is a list that contains the location of the design points, a covariance function for the random field and several other pieces of information. At first it may seem overly complicated to use a list to represent a design. After all it is just a set of points in two dimensions. However, the list structure is important to keep all the necessary information about the design together with the coordinates. The names of the example network objects included in *Di* are `chicago.nw`, `houston.nw`, `gulfstates.nw`, and `square.nw`.

Furthermore it is easy to create a network object from just the coordinates. This is done using the `as.nw` function. Suppose one wanted to look at 3X3 unit grid with an exponential covariance function

$$k(\mathbf{x}, \mathbf{x}') = 5e^{-\|\mathbf{x}-\mathbf{x}'\|/2}$$

.

```
> locs<- cbind( c( 1,2,3,1,2,3,1,2,3), c( 1,1,1,2,2,2,3,3,3))
> locs.nw<- as.nw(locs)
> locs.nw$cov<- "5.0*exp(-d/2.0)"
> locs.nw   # print out description
```

The S data set `locs.nw` is now ready for editing. If you ever need to see all the components of the object just use `print.list( locs.nw)` A network catalog object is a list where each comnpnent is a network object. This structure is a natural way to accumlate spatial designs as they are created in the design editor. Although `edit.nw` always returns a network catalog object it is easy to extract one design fromteh lilst. For example supppose that after editing a design you have a network catalog data set , `work.nw` that has 5 different networks. You would just like the

fourth one as a single network object. Use the subscripts for lists: `work.nw[[4]]->favorite.network` .

4 Getting and Installing Di

The file `di.tar.Z` contains the compressed and tarred programs for the Di package and can be obtained via "anonymous FTP" from the NISS server

1. In UNIX: `ftp server.niss.rti.org`
2. Log in as `anonymous`
3. Give your e-mail address as the password
4. Commands within FTP to get the *Di* tar file:

```
cd pub
get di.tar.Z
exit
```

You should now have a file called `di.tar.Z` on your local directory
Now in Unix do the following:

1. Create a directory Di and move the file di.tar.Z to this directory
2. `uncompress di.tar.Z`
3. Extract the files from the tar file `tar -xfv di.tar`
4. Setup S functions and help files `make all`

If you are curious what the program `make` does to install *Di* take a look at the file `Makefile` in the Di directory.

There will be several files now in the Di directory. Among them are `manual.tex` and `manual.ps`. These are the LaTeXand postscript versions of this manual.

5 An example

This section will give an example of using the network editor on an artifical network. The EXAMPLES section in the help file for the design editor contains this example and three others. In these examples unless noted "click" refers to pressing the left mouse button and text in parenthesis refers to a button in a menu. Some of the graphical results from this example are reproduced in Figures 1 and 2.

```
 > square.catalog <- edit.nw(square.nw)
# Perform the following manual actions with the mouse:
# Plot current design.
  1) click (plot) followed by (ok) to view spatial plot of the PSE.
     click (return) on the submenu that pops up.
# Modify design by deleting points.
  2) click (delete points) followed by (ok)
     click left mouse button on points (2,2), (2,4), (4,2) and
     (4,4) of the main plot window.  click the middle mouse button.
# Plot new design.
  3) click (plot) followed by (ok)
     click (return) on the submenu
# Save this modified design.
  4) click (save) followed by (ok)
# View summary of prediction variance for original and new designs.
  5) click (summary) followed by (ok)
# Note summary of prediction variance for both designs is print out
# in the work window.
# Exit edit.nw
  6) click on (exit)
# Now examine the output, square.catalog
  > square.catalog
# Note that attributes of the two designs are print to the screen.
# Now we wish to start edit.nw up using the modified design from
# the previous editing session and save any new designs in
# square.catalog.
  > square.catalog <- edit.nw(square.catalog,root.id=2)
# Further modifications made to this design may be saved in
# ''square.catalog'' and the numbering will begin at 3.
# Plot prediction variance surface.
  7) click (plot) (ok)
  8) click (add) (ok)
  9) click left button at the points (2,2) and (4,2) click on
     middle button when you are done
 10) click (plot) (ok)
```

```
  11) click (save) (ok)
# summarize the three designs
  12) click (summary) (ok)
# plot summarry information
  13) click (plot summary) (ok)
  14) click (return)
  15) click (exit)
```

Within Splus now one can analyze the three designs in square.catalog. For example `summary( square.catalog)` gives

```
  size   mean PV    max PV median PV
1   25 0.4009779 0.5078271 0.4361211
2   21 0.4626009 0.6608440 0.4720894
3   23 0.4317996 0.6607474 0.4514928
```

6 Details on automatic methods for design construction and thinning

6.1 Leaps

The initial version of *Di* has only one automatic proceudre for thinning the number of points. What is referred to as the *leaps* algorithm determines the subset of locations that most closely estimates the average of the full network. For example the Chicago network has 21 locations. One may be interested in finding a subset of 5 locations that best estimate the 21 point average. Of course the answer will depend on the properties of the spatial field and the leaps procedure must use the covariance function the as part of the current design.

7 How *Di* works in S

The key idea in *Di* is the use of a network object (`nw`) to bundle together all the information that is needed to evaluate and visualize a design. Supporting these func-

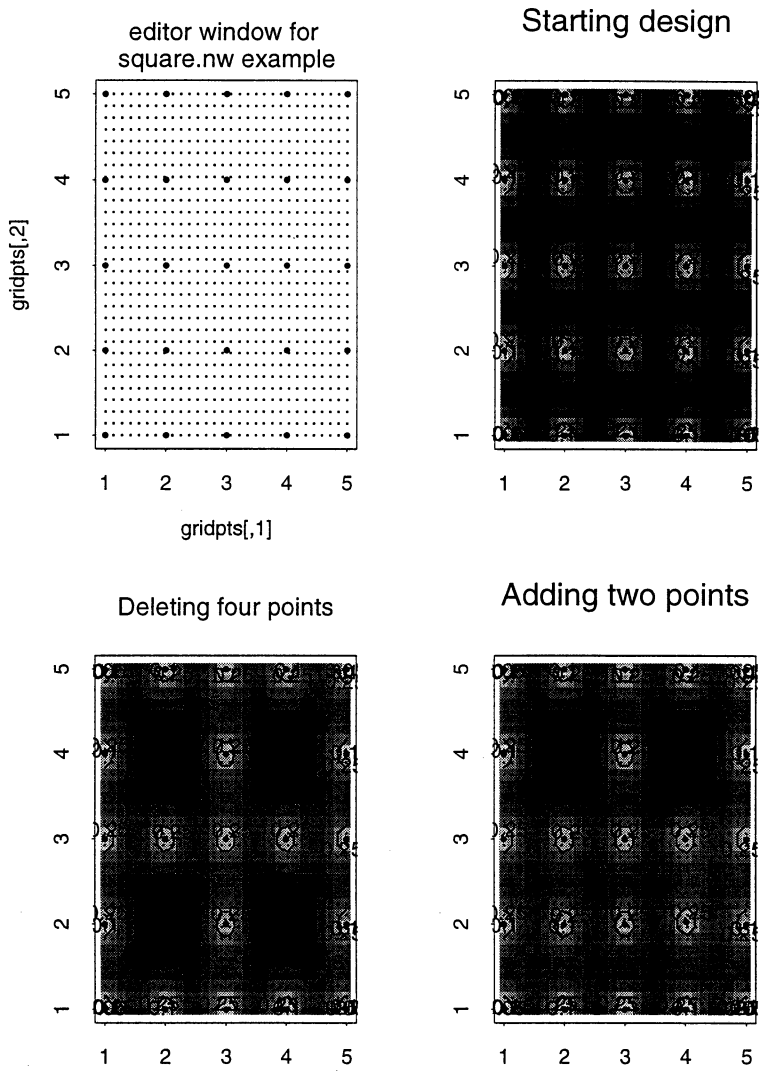

Figure 1: Results from applying `edit.nw` to the square example. Top right plot is the inital design in the editor window. Remaining three plots are the prediction variance surface for the full design, omitting four points and then adding back in two. These plots appear as separate windows within the design editor.

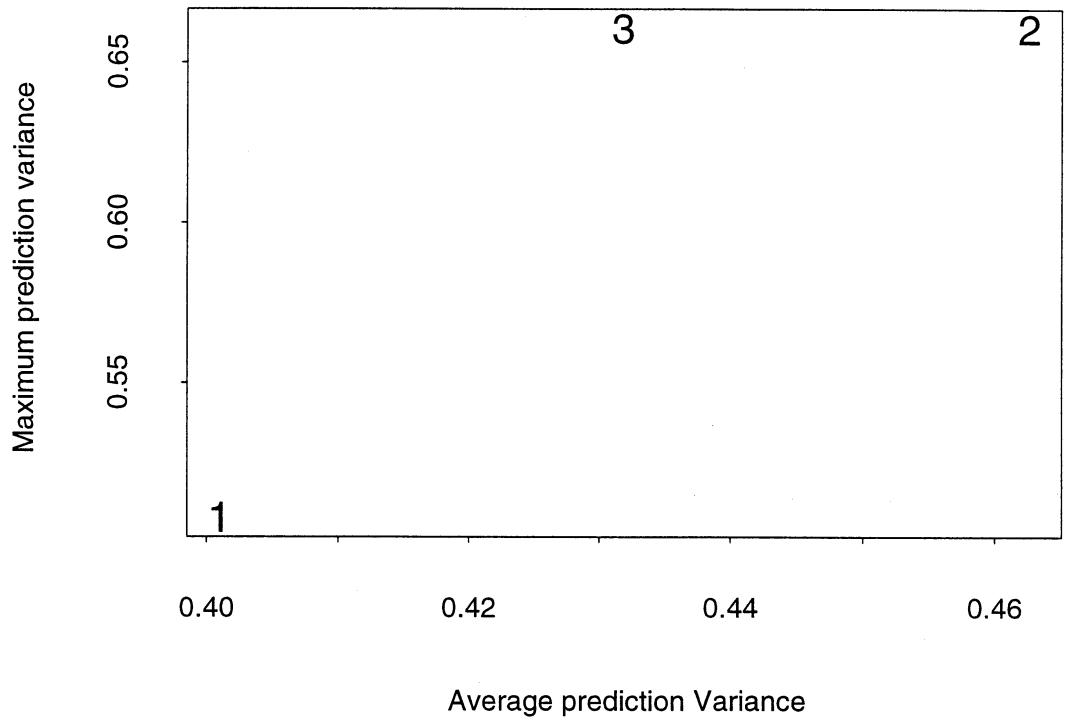

Figure 2: Plot summary of the three designs from the example (`square.nw`). This plot would appear as a separate window within the design editor.

tions is the network catalog object (`nw.catalog`) to keep several network objects together. With this structure we use the object oriented tools in Splus to simplify looking at designs. For example the command `print( square.nw)` is equivalent to `print.nw( square.nw)` because `square.nw` is of class `nw` and we have written a special purpose print function for these objects. This is the reason why a network object is does not print as a list. The functions `summary` and `plot` function in a similar manner. The functions `print` and `summary` also have been extended to network catalog objects.

7.1 Design editor

The structure of the editor is simple in terms of manipulating objects. One supplies the editor with a network (or network catalog) object. This is converted to a network catalog object and one of the network objects is displayed. This design is the one that can now be edited and the main menu now gives choices for different actions. Each action either applies a function to the current design object or modifies the current design object to produce a new one. When a modified design is saved it is simply added to the network catalog. Finally when the editor function is quit it returns a network catalog object.

User interaction with *Di* occurs through Motif dialog boxes (as implemented in S) and Motif graphics windows. There is one main dialog box through which all actions are accessed. *Di* has been written so that is it simple to add new actions to this menu. Each button corresponds to a few lines of S code and thus the basic loop in the editor is to identify the selected button and evaluate the S code that corresponds to that button. The list linking button names (the *items* in the main menu) to S code that performs the right operations is in the data set `di.items`. A listing of this data set is in the appendix. Section 7.2 explains how to modify this data set to change what the the *Di* editor will do.

7.2 Modifying the main menu

The previous section explains how the main menu provides actions that either describe or change the current network object (design) To add more actions in the main menu box or to change the current set all that is needed is to change the *Di* action data set, `di.items`. This can be done within S by editing the actions list. For example to use the textedit editor for the SUN workstations

```
> di.items-> di.items.old   # save the current version
>    tx(di.items) -> di.items
```

When writing the S code for new actions one should keep in mind two working data sets that are used within the editor. `nw` is the name of the current network object being studied. If any changes are made to `nw` then the component `nw$id` should be set to zero. This way the save function will recognize this design as a new one and save it. The other important object in the editor is `design.history`. This is a network catalog object that contains the initial network object and any others that are saved in the course of using the editor.

Reading through the default version of `di.items` is a good way to see how network objects are modified and summarized. Note that some of the more complicated actions are separate S functions that take the `nw` or `design.history` as one input argument. It is good idea to test out new S code outside of the editor rather trying to develop and debug new code by directly editing `di.items`. Change the actions list once you are confident that your new source code works.

Appendix 1: Listing of *Di* functions and data sets

Di functions

RDIST	Computes distance between two vectors
US	draws US map
as.nw	converts a matrix to a network object
as.nw.catalog	converts matrix or network object to a network catalog object
as.surface	reformats vector for surface plotting
best.subset.leaps	Constructs covariance matrix an calls low level leaps function
count	counts the number of true values in vector of logical values
di.add	Di function to add points to design
di.chcov	Di function to change the covariance function
di.chgrid	Di function to change range for plotting
di.default.cov	Di default covariance (unit exponential)
di.del	Di function to delete points form a design
di.leaps	top level Di function to find subset using leaps
di.recalc	Di function to recalculate the prediction variance of design
di.replot	Di function to replot the editor window
edit.nw	Edit a network object
extra	lists objects that are not part of the DI distribution
help.to.latex	converts help file to a form to use in LaTeX
is.nw	tests for network object
is.nw.catalog	test for network catalog
leaps.subset.r2	Basic function for leaps algorithm
make.ftp	creates the UNIX source files for FTP tar files.
make.qfiles	copies functions and data sets to source code directory
make.surface.grid	creates a grid for surface evaluation
plot.nw	plot a network object
plot.summary.nw.catalog	plot summaries of a network catalog
pred	calculate prediction variance at set locations
print.nw	print network object
print.nw.catalog	print network catalog object
rdist.earth	great circle distance between two lat/lon locations
summary.nw	summarize network object
summary.nw.catalog	summarize a network catalog object
tx	edits an S dataset using the SUN textedit editor

Di data sets

DI.version	version number of *Di*
di.items	list matching button values with S code
dump.list	list of all *Di* function and data sets from distribution

Di example data sets

chi.loc	locations in lat-lon of Chicago area ozone stations
ex	example network object for Chicago ozone locations
houston.nw	example network for Houston,TX coast ozone stations
chicago.nw	example network for Chicago area ozone stations
square.nw	example network 5X5 unit grid
gulfstates.nw	example network ozone stations for US gulf coast region
square.catalog	output from following example 1

Appendix 2: Listing of `di.items`

```
"di.items"<-
expression("Add Points" = {
nw <- di.add(nw)
nw$id <- 0
}
, "Delete Points" = {
nw <- di.del(nw)
nw$id <- 0
}
, "Change Grid" = {
nw <- di.chgrid(nw)
}
, "Change Cov" = {
temp <- list(n = nw, history = history.design)
temp <- di.chcov(temp)
nw <- temp$n
history.design <- temp$history
nw$id <- 0
}
, Leaps = {
nw <- di.leaps(nw)
nw$id <- 0
gridpts <- di.recalc(nw)
di.replot(nw, gridpts)
}
, Plot = {
plot.nw(nw, winsize = "400x400", winloc = "+0-0", ask = ask)
dev.set(mainwin)
}
, "Plot Summary" = {
sumwin <- plot.summary.nw.catalog(summary(history.design), sumwin =
sumwin, ask = ask)
dev.set(mainwin)
}
, Summary = {
print(summary(history.design))
dev.set(mainwin)
}
, Print = {
print(history.design)
```

```
}
, Reinitialize = {
nw <- history.design[[root.id]]
gridpts <- di.recalc(nw)
di.replot(nw, gridpts)
}
, Save = {
if(nw$id == 0) {
new.id <- length(history.design) + 1 ##
nw$id <- new.id ##
history.design[[format(new.id)]] <- nw ##
}
}
)
```

Appendix 3: Splus help file for `edit.nw` and `as.nw`

To save space the EXAMPLES sections of the help file have been omitted form these listings. They can be read using the help from within Splus.

`edit.nw`

```
Edit NetWork object or catalog


DESCRIPTION:
       Allows the interactive design and  evaluation  of  spatial
       networks.

USAGE:
       nw.ouput<-edit.nw(nw=ex, cov.char, root.id=1,ask=FALSE)

REQUIRED ARGUMENTS:
nw:    An object of class "nw"  or  class  "nw.catalog".  An  nw
       object contains the following components:

       $npts : A list with components $x and $y,  the  x  and  y-
       coordinates of the data.

       $grid       :       A       vector       containing
       (min(x),min(y),max(x),max(y),ngx,ngy)  where  ngx  and ngy
       are the x and y  dimensions  of  the  grid  on  which  the
       properties of the random field are evaluated.

       $cov  : A text string evaluated as a function of  distance
       that  describes  the  covariance  between  two points. The
       default covariance function in the provided nw objects  is
       the exponential model: c(d) = exp(-d/a), which is supplied
       as "exp(-d/a)" for some numerical value of  the  parameter
       a.   The  function  as.nw  will  use  this function as the
       default.   The  covariance  function  may  be  modified
       interactively  once  edit.nw  has  been  invoked.  (see
       DETAILS).

       $map  : An indicator of the coordinate system of the data.
       T indicates longitude/latitude coordinates and F indicates
       regular Euclidean coordinates.  If  T,  state  and  county
       lines will be produced on any spatial plot.

       $id   : Design number.  If nw object is  a  member  of  an
       object  of  class  nw.catalog,  then  $id indicates design
       number.
```

as.nw will coerce a matrix of coordinates into an nw object.

OPTIONAL ARGUMENTS:

ask: If FALSE, a new windowis opened for each new plot. If TRUE, plot requests generate a menu of plot window choices, which includes all available graphics windows plus the option to create a new window.

cov.char: An alternative character vector which is an expression that evaluates to the covariance between two points. If this is non-missing, it overrides $cov from the nw object.

root.id: If nw is of class "nw.catalog", then root.id refers to the design which is to be loaded as the default.

VALUE:

nw.ouput contains a history of all designs manually saved by clicking on the (save) button of the main menu. nw.output is of class "nw.catalog".

SIDE EFFECTS:

Most commands of the main menu produce a secondary plot window and/or a secondary pop-up menu and/or an edit window and/or produce output to the working window.

DETAILS:

When "edit.nw" is invoked with a particular nw object, the data locations are plotted in the primary plot window and the network may be evaluated, manipulated and modified using the following commands of the main menu:

(add points) : Allows addition of points to network using the mouse.

(delete points) : Allows deletion of points to network using the mouse.

(change grid) : Allows the number of points in the grid, and the corners of the grid to be modified.

(change cov) : Allows modification of the covariance function either by changing parameter values or changing the functional form of the covariance. The 'apply retroactively' button applies the new covariance function to all previous designs.

(leaps) : Uses regression subset selection algorithm to select the best "p" site subset of existing sites.

(plot) : Produces a graphical display of the prediction

variance over space for the current design.

(plot summary) : Produces a plot of the maximum prediction variance vs. the mean prediction variance for all saved designs.

(summary) : Outputs the mean, maximum, and median prediction variance to the working window, for all saved designs.

(print) : Prints attributes of all saved designs to the work window.

(reinitialize) : Replots design root.id in the main plot window.

(save) : Adds the current design to the nw.catalog which is returned upon clicking (exit).

(ok) : Executes one of the preceeding commands.

(exit) : Exits edit.nw

These commands are invoked by first clicking on the corresponding command, and then clicking on the (ok) command.

IMPORTANT: After invoking most commands, instructions
appear in the S+ work window. These are very useful.
If edit.nw appears hung, look for a possible explanation
here. Additionally, output from many of the actions
appears here.

SECONDARY MENUS: After most commands are invoked, a secondary menu appears. You must click on (return) of these menus before returning to the main menu.

SECONDARY PLOT WINDOWS: If the (plot) command is invoked, a spatial plot of the prediction variance (PV) is made. There are options associated with these plots. The graphical ouput from each (plot) call remains in the workspace unless dismissed, or until the memory of the workstation is used up, at which point you are thrown out of DI and lose all work. (plot summary) also produces a secondary plot window of maximum PV vs. mean PV. If the argument 'ask' is TRUE,plot requests generate a menu of plot window choices which includes all available graphics windows (the currently active window is listed first and highlighted) plus the option to create a new window. Using ask=TRUE is helpful for reducing crashes due to excessive plotting resource requests during extended

editing sessions.

SAVING DESIGNS: If you wish to save a design after any given manipulation (changing the covariance, adding or deleting points, changing the grid size, etc..), the (save) command must be invoked. The first time (save) is invoked, the default design of the nw object is saved as design 1, and the current and all subsequent designs are numbered beginning with 2.

COVARIANCE: The covariance function must be supplied in the nw object. It may be given as an argument to edit.nw, in which case $cov from the nw object is overridden. Once edit.nw is invoked, the covariance function may be edited by selection (change cov) on the main menu. If this is done, a pop-up editor is provided with the current covariance function loaded. The covariance function for a particular design is saved with the nw object of that design provided (save) is done. Thus the nw.catalog produced at the end of an edit.nw session will have the covariance information for all designs saved.

REFERENCES:

SEE ALSO:

as.nw, is.nw, plot.nw, as.nw.catalog, is.nw.catalog

DETAILS:

as.nw takes the coordinate matrix and computes the relevant components of $grid. A default grid size of 30x30 is used. $cov defaults to the exponential model with unit parameter "exp(-d/1)". $map defaults to F. $id defaults to 1.

REFERENCES:

SEE ALSO:

edit.nw, is.nw, plot.nw, is.nw.catalog, as.nw.catalog

EXAMPLES:

```
# Coerce a 3 x 3 grid into an object of class "nw", check that this is
# of the correct class, nw,  and examine the object and it's elements.

  > grid <- cbind(c(1,1,1,2,2,2,3,3,3),c(1,2,3,1,2,3,1,2,3))
  > grid.nw <- as.nw(grid)
  >
  > is.nw(grid.nw)
    [1] T
  >
  > grid.nw
    size grid x1 grid x2 grid y1 grid y2
      9       1       3       1       3
  >
  > names(grid.nw)
    [1] "npts" "grid" "cov"  "map"  "id"
  >
  > grid.nw$npts
    $x:
    [1] 1 1 1 2 2 2 3 3 3

    $y:
    [1] 1 2 3 1 2 3 1 2 3
  >
  > grid.nw$grid
    [1]  1  1  3  3 30 30
  >
  > grid.nw$cov
    [1] "exp( -d/1.0)"
  >
  > grid.nw$map
    [1] F
  >
  > grid.nw$id
    [1] 1
```

as.nw

As.NetWork

DESCRIPTION:

as.nw will coerce an nx2 coordinate matrix, or a list with x and y components into an object of class "nw".

USAGE:

```
as.nw(o, map=F)
```

REQUIRED ARGUMENTS:

o: An n x 2 matrix of spatial locations.

OPTIONAL ARGUMENTS:

map: If T, map indicates that coordinates are in latitude/longitude. Default is set to F and indicates non-map coordinates.

VALUE:

Returns an object of class "nw" which is suitable for use in the function edit.nw. An object of class "nw" is a list which contains the following attributes:

$npts : A list with components $x and $y, the x (longitude) and y (latitude) coordinates of the data.

$grid : A vector with components (min(x),min(y),max(x),max(y),npx,npx) where npx and npy are the size of the grid in the x and y directions. npx and npy default to 40.

$cov : A text string defining the covariance (as a function of distance) between any two points. The default covariance is an exponential model with unit parameter: "exp(-d)".

$map : T to indicate that coordinates are latitude/longitude, F otherwise. Default is F.

$id : A pointer to identify the design number in the case several nw objects are nested within a single object of class "nw.catalog". For a single nw object, id will have the value 1.

SIDE EFFECTS: