CEDL script User's Guide v1.0

April 22, 2005

N. Bert Loosmore `nhl@u.washington.edu`

1 Introduction

This section describes how to use the provided *CEDL - Complete Empirical Distance List* software functions. The functions test an observed pattern against a selected null distribution. In the current version of the software, the null distribution defaults to CSR, and to test a different H_0, the source code must be modified.

The software is written to run under `R` v1.9.1, and requires the `spatstat` library. It should also work under S+ with slight modifications. These have not been included and have not been tested.

2 Obtaining and loading the software into `R`

The latest `R` source code, including additional documentation, is available for download at:

`http://faculty.washington.edu/edford/research/research_home.html`

Please review the newest version of the documentation as these instructions or the interface may change.

Currently, the software is provided as a script (text) file and thus can easily be modified by the user. To load the script file and hence make the necessary functions available, at the `R` prompt (>) type:

```
> source("cedl_scr.r")
```

If the script loads correctly, it should return with a clean prompt.

3 To test an observed pattern

Two main functions, one for the G and one for the K, exist within the script file that should be called by the user. Additional functions exist within the file, but these are helper functions that the basic user will not need to modify.

The interface to the two functions are the similar. To test a pattern using the G statistic use `cedl.G()`. The function declaration is:

```
cedl.G <- function(obs.pp, s, lambda=obs.pp$n, tmax=1, prec=0.00001,
        use.exist=F, verbose=F)
```

Input Variables:

`obs.pp` The observed pattern, which should be of type `pp` as defined in the `spatstat` library. Also, observed pattern must be scaled to a unit square boundary box.

`s` The number of patterns from which to create the null distribution. Note since the null distribution contains the observed pattern, set this number to 1 less than the actual s value. For example, for a null distribution with 1000 total patterns, set $s = 999$.

`lambda` Optional. Allows the user to set the point intensity of the simulated pattern. If not specified, this is extracted from the observed pattern.

`tmax` Optional. Specifies the upper limit of integration. Can be used to set a transistion point for $w(t) = 0$.

`use.exist` Optional. If set to `T` causes the function to use the existing `my.pp.list` pattern list instead of generating a new one. This allows, for example, a user to run the goodness of fit test against the same set of patterns used for drawing a simulation envelope. Alternatively, this allows a user to build the pattern list from non CSR patterns to test a different H_0.

`verbose` Optional. Causes additional status information to be printed to the screen (e.g. which subfunctions are being called) during processing. Useful since some parameterizations take a while to execute.

Return Values:

The function returns a list containing three variables:

`p.val` The p-value of the observed pattern when tested against the null hypothesis that it comes from the same distribution as the simulated patterns.

`ui` The list of u_i values for the observed pattern u_1 and the simulated patterns u_i, $i = 2, \ldots, s$. This is returned to allow the user to inspect the distribution of the u_i.

`Go` The grand mean calculated over all s patterns.

`my.dist` The distances at which the grand mean is calculated. Ranges from 0 to `tmax`.

The p-value of the observed pattern is printed to the screen if `verbose` is set to `T`. The p-value of the observed pattern is calculated as:

```
p.val <- (length(ui[ui>=ui[1]]))/(length(ui))
```

Hence if the observed pattern has the largest u_i value, then the p-value is calculated as $1/s$.

Similarly, to test a pattern against the K statistic use `cedl.K()`, where the function declaration is:

```
cedl.K <- function(obs.pp, s, lambda=obs.pp$n, tmax=0.25,
        prec=0.00001, use.exist=F, verbose=F)
```

Note here that the `tmax` value defaults to 0.25. As described above, this should be set based on the researchers knowledge of the ecological system. All other input variables and the return variables are the same.

4 To generate an alternate H_0

This section desribes how to generate a pattern list from patterns other than from a CSR process. Note that under normal operation, the `cedl.G()` or `cedl.K()` functions will generate the pattern list.

The `gen.ext.pp.list()` function is used to generate the external pattern list. It defaults to creating CSR patterns. To generate patterns from an alternate process, two things are necessary. First, this function needs to be modified. Modify the following line:

```
    my.pp.list[[i]] <<- rSSI(r=0.0045, n=npts, giveup=10000)
```

to include the process of interest. The current example is for a Strauss inhibition process with an inhibition radius of 0.0045. See the `spatstat` user's guide for detail about other possible spatial processes. Second, call this function prior to the call to `cedl.G()` or `cedl.K()`. When calling `gen.ext.pp.list()` set the `type` variable to something other than `"CSR"`.

Lastly, as described above, when calling `cedl.G()` or `cedl.K()`, set `use.exist=T`.

5 Example code

```
# load the cedl_scr.r script file (make sure it can be found in the path!)
> source("cedl_scr.r")

# create an inhibition pattern with 100 points with an inhibition radius
#  of 0.02 (on a unit square).
> my.pp <- rSSI(r=0.02, n=100)

# test the pattern using the K statistic against H0 of CSR based on
#  2000 total patterns, but only up to a distance of 0.10
> my.K.rslt <- cedl.K(obs.pp=my.pp, s=1999, tmax=0.10)
```