

Some R and Spatstat Basics

1. R

1.1 Introduction

Object-oriented data analysis and programming environment.

Objects (~data *and* clues about how to handle it) have classes and ...

Behavior of functions depends on object class of arguments.

Highly extensible: a huge number of packages are available.

Learning curve is a little steep.

FREE!

1.2 Working directories

In the R GUI:

```
File | Change Directory
```

1.3 Importing data

This is easy, as long as the data are in either comma-delimited or tab-delimited ASCII files.

Here's the basic syntax:

```
read.table("c:/localdata/houses.dat", header = TRUE,  
           row.names = 1)
```

Note the front slashes!!

Note the use of <- for assignment.

1.4 File names in Windows

R uses C-style string handling, \ is treated as an escape character, so that for example one can enter a newline as \n. When you really need a \, you have to escape it with another \.

Thus, in filenames use something like "c:\\data\\money.dat". You can also replace \ by / ("c:/data/money.dat").

1.5 Graphics:

When you invoke a graph procedure, a separate window pops up which has two menus.

File menu allows saving or printing. The **History** menu allows the recording of plots. When plots have been recorded they can be reviewed by PgUp and PgDn, saved and replaced.

Recording can be turned on automatically (the Recording item on the list) or individual plots can be added (Add or the INS key).

1.6 R Libraries

To find out which additional packages are available on your system, type

```
library()
```

at the R prompt.

You can “load” the installed package *pkg* by

```
library(pkg)
```

You can then find out which functions it provides by typing one of

```
library(help = pkg)
help(package = pkg)
```

You can unload the loaded package *pkg* by

```
detach("package:pkg")
```

We need:

```
library(spatstat)
```

2. spatstat

2.1 The steps in our analysis:

-read in the data: `read.table()`

-create an observation window (how big is the study area?): `owin()`, `ripras()`

-create a point-pattern object (which contains both the points and the observation window): `ppp()`

-estimate K-function: `Kest()`.

-estimate pair correlation function: `pcf()`

-KDE: `density.ppp()`

2.2 Create a window: `owin()`

```
owin(xrange=c(0,1), yrange=c(0,1), poly=NULL,
     units=NULL)
```

Arguments:

`xrange` x coordinate limits of enclosing box

`yrange` y coordinate limits of enclosing box

`poly` Optional. Polygonal boundary of window.

`units` Optional. Name of unit of length. Either a single character string, or a vector of two character strings giving the singular and plural forms, respectively.

Example:

```
w <- owin(poly=list(x=c(0.5,1,0.5,0),y=c(0,1,2,1)))
```

For us:

```
attach(limits) #makes individual variables available
house.win1<-owin(poly=list(x=easting,y=northing))
```

2.3 Create a another window , the edges of a Poisson forest: `ripras()`

```
ripras(x, y=NULL)
```

Arguments:

`x` vector of x coordinates of observed points, or a 2-column matrix giving x,y coordinates, or a list with components x,y giving coordinates.

`y` (optional) vector of y coordinates of observed points, if x is a vector.

For us:

```
detach(house) # make individuals variables
unavailable
attach(limits)
```

```
house.win2<-ripras(Easting,Northing)
```

2.4 Create a ppp object: ppp()

```
ppp(x,y, window, marks)
```

Arguments

x Vector of x coordinates of data points
y Vector of y coordinates of data points
window window of observation, an object of class "owin"
marks (optional) vector of mark values

Example:

```
X <- ppp(x, y, window=owin(c(0,1),c(0,1)))
```

For us:

```
attach(house)
house1.ppp<-ppp(Easting,Northing>window=house.win1)
plot(house.ppp)
```

2.5 Estimate the K-function: Kest()

```
Kest(X, ..., r=NULL, correction=c("border",
 "isotropic", "Ripley", "translate"))
```

Arguments:

X The observed point pattern, from which an estimate of $K(r)$ will be computed. An object of class "ppp", or data in any format acceptable to `as.ppp()`.
r Optional. Vector of values for the argument r at which $K(r)$ should be evaluated. Note: the default is supposed to be sensible, but beware! Best to specify this yourself.
correction Optional. A character vector containing any selection of the options "border", "bord.modif", "isotropic", "Ripley" or "translate". It specifies the edge correction(s) to be applied.

Vales returned:

An object of class "fv", see `fv.object`, which can be plotted directly using `plot.fv`. Essentially a data frame containing columns:
r the vector of values of the argument r at which the function K has been estimated
theo the theoretical value $K(r) = \pi * r^2$ for a stationary Poisson process
together with columns named "border", "bord.modif", "iso" and/or "trans", according to the selected edge corrections. These columns contain estimates of the function $K(r)$ obtained by the edge corrections named.

For us:

```
r1<-c(0,5,10,15,20,25,30,35,40,45,50,55,60)
house.K<-Kest(house.ppp,r=r1)
plot(house.K)
```

2.6 Estimate the pair-correlation function: pcf()

```
pcf(X, ..., r = NULL, kernel="epanechnikov", bw=NULL,
 correction=c("translate", "ripley"))
```

Arguments

X A point pattern (object of class "ppp"), OR an estimate of its K function

`r` Vector of values for the argument `r` at which `g(r)` should be evaluated.
`kernel` Choice of smoothing kernel, passed to `density`.
`bw` Bandwidth for smoothing kernel, passed to `density`.
`correction` Choice of edge correction.

For us:

```
r1<-c(0, 5,10,15,20,25,30,35,40,45,50,55,60)
house.PCF<-pcf(house.ppp,r=r1)
plot(house.PCF)
```

Also try:

```
House.PCF1<-pdf(house.K)
```

2.7 Kernel density estimation: `density.ppp()`

```
density(x, sigma, edge=TRUE)
```

Arguments:

`x`: Point pattern (object of class "ppp") to be smoothed.
`sigma`: Standard deviation of isotropic Gaussian smoothing kernel.
`edge`: Logical flag: if TRUE, apply edge correction.

For us:

```
house.kde<-density(house.ppp)
plot(house.kde)
contour(house.kde)
points(house.ppp)
```

2.8 Inhomogeneous K function: `Kinhom()`

```
Kinhom(X, lambda, r = NULL, correction=c("border", "bord.modif",
" isotropic", "translate"))
```

Arguments:

`X` The observed data point pattern, from which an estimate of the inhomogeneous K function will be computed. An object of class "ppp" or in a format recognised by `as.ppp()`
`lambda` Optional. Values of the estimated intensity function. Either a vector giving the intensity values at the points of the pattern `X`, or a pixel image (object of class "im") giving the intensity values at all locations.
`r` vector of values for the argument `r` at which the inhomogeneous K function should be evaluated.
`correction` A character vector containing any selection of the options "border", "bord.modif", "isotropic", "Ripley" or "translate". It specifies the edge correction(s) to be applied.

For us:

```
house.Kinhom<-Kinhom(house.ppp,r=r1,
correction="translate")
plot(house.Kinhom)
house.pcf<-pcf(house.Kinhom)
plot(house.pcf)
```