Spatstat Quick Reference guide

June 6, 2015

`spatstat-package` *The Spatstat Package*

Description

This is a summary of the features of **spatstat**, a package in R for the statistical analysis of spatial point patterns.

Details

spatstat is a package for the statistical analysis of spatial data. Currently, it deals mainly with the analysis of spatial patterns of points in two-dimensional space. The points may carry auxiliary data ('marks'), and the spatial region in which the points were recorded may have arbitrary shape.

The package supports

- creation, manipulation and plotting of point patterns
- exploratory data analysis
- simulation of point process models
- parametric model-fitting
- hypothesis tests and model diagnostics

Apart from two-dimensional point patterns and point processes, **spatstat** also supports point patterns in three dimensions, point patterns in multidimensional space-time, point patterns on a linear network, patterns of line segments in two dimensions, and spatial tessellations and random sets in two dimensions.

The package can fit several types of point process models to a point pattern dataset:

- Poisson point process models (by Berman-Turner approximate maximum likelihood or by spatial logistic regression)
- Gibbs/Markov point process models (by Baddeley-Turner approximate maximum pseudolikelihood, Coeurjolly-Rubak logistic likelihood, or Huang-Ogata approximate maximum likelihood)

- Cox/cluster process models (by Waagepetersen's two-step fitting procedure and minimum contrast, or by composite likelihood)

The models may include spatial trend, dependence on covariates, and complicated interpoint interactions. Models are specified by a `formula` in the R language, and are fitted using a function analogous to `lm` and `glm`. Fitted models can be printed, plotted, predicted, simulated and so on.

Getting Started

For a quick introduction to **spatstat**, read the package vignette *Getting started with spatstat* installed with **spatstat**. To see this document, you can either

- visit `cran.r-project.org/web/packages/spatstat` and click on `Getting Started with Spatstat`
- start R, type `library(spatstat)` and `vignette('getstart')`
- start R, type `help.start()` to open the help browser, and navigate to `Packages > spatstat > Vignettes`.

For a complete 2-day course on using **spatstat**, see the workshop notes by Baddeley (2010), available on the internet. (This is now *very* out-of-date but it will get you started.)

Type `demo(spatstat)` for a demonstration of the package's capabilities. Type `demo(data)` to see all the datasets available in the package.

For information about handling data in **shapefiles**, see the Vignette *Handling shapefiles in the spatstat package* installed with **spatstat**.

To learn about spatial point process methods, see the short book by Diggle (2003) and the handbook Gelfand et al (2010).

Updates

New versions of **spatstat** are released every 8 weeks. Users are advised to update their installation of **spatstat** regularly.

Type `latest.news` to read the news documentation about changes to the current installed version of **spatstat**.

See the Vignette *Summary of recent updates*, installed with **spatstat**, which describes the main changes to **spatstat** since the workshop notes (Baddeley, 2010) were published.

Type `news(package="spatstat")` to read news documentation about all previous versions of the package.

FUNCTIONS AND DATASETS

Following is a summary of the main functions and datasets in the **spatstat** package. Alternatively an alphabetical list of all functions and datasets is available by typing `library(help=spatstat)`.

For further information on any of these, type `help(name)` where `name` is the name of the function or dataset.

CONTENTS:

I. CREATING AND MANIPULATING DATA

Types of spatial data:

The main types of spatial data supported by **spatstat** are:

`ppp`	point pattern
`owin`	window (spatial region)
`im`	pixel image
`psp`	line segment pattern
`tess`	tessellation
`pp3`	three-dimensional point pattern
`ppx`	point pattern in any number of dimensions
`lpp`	point pattern on a linear network

To create a point pattern:

`ppp`	create a point pattern from (x, y) and window information
	`ppp(x, y, xlim, ylim)` for rectangular window
	`ppp(x, y, poly)` for polygonal window
	`ppp(x, y, mask)` for binary image window
`as.ppp`	convert other types of data to a ppp object
`clickppp`	interactively add points to a plot
`marks<-`, `%mark%`	attach/reassign marks to a point pattern

To simulate a random point pattern:

`runifpoint`	generate n independent uniform random points
`rpoint`	generate n independent random points
`rmpoint`	generate n independent multitype random points
`rpoispp`	simulate the (in)homogeneous Poisson point process
`rmpoispp`	simulate the (in)homogeneous multitype Poisson point process
`runifdisc`	generate n independent uniform random points in disc
`rstrat`	stratified random sample of points
`rsyst`	systematic random sample of points
`rjitter`	apply random displacements to points in a pattern

rMaternI	simulate the Matérn Model I inhibition process
rMaternII	simulate the Matérn Model II inhibition process
rSSI	simulate Simple Sequential Inhibition process
rStrauss	simulate Strauss process (perfect simulation)
rHardcore	simulate Hard Core process (perfect simulation)
rDiggleGratton	simulate Diggle-Gratton process (perfect simulation)
rDGS	simulate Diggle-Gates-Stibbard process (perfect simulation)
rNeymanScott	simulate a general Neyman-Scott process
rPoissonCluster	simulate a general Poisson cluster process
rMatClust	simulate the Matérn Cluster process
rThomas	simulate the Thomas process
rGaussPoisson	simulate the Gauss-Poisson cluster process
rCauchy	simulate Neyman-Scott Cauchy cluster process
rVarGamma	simulate Neyman-Scott Variance Gamma cluster process
rthin	random thinning
rcell	simulate the Baddeley-Silverman cell process
rmh	simulate Gibbs point process using Metropolis-Hastings
simulate.ppm	simulate Gibbs point process using Metropolis-Hastings
runifpointOnLines	generate n random points along specified line segments
rpoisppOnLines	generate Poisson random points along specified line segments

To randomly change an existing point pattern:

rshift	random shifting of points
rjitter	apply random displacements to points in a pattern
rthin	random thinning
rlabel	random (re)labelling of a multitype point pattern
quadratresample	block resampling

Standard point pattern datasets:

Datasets in **spatstat** are lazy-loaded, so you can simply type the name of the dataset to use it; there is no need to type `data(amacrine)` etc.

Type `demo(data)` to see a display of all the datasets installed with the package.

Type `vignette(datasets)` for a document giving an overview of all datasets, including background information, and plots.

amacrine	Austin Hughes' rabbit amacrine cells
anemones	Upton-Fingleton sea anemones data
ants	Harkness-Isham ant nests data
bdspots	Breakdown spots in microelectrodes
bei	Tropical rainforest trees
betacells	Waessle et al. cat retinal ganglia data
bramblecanes	Bramble Canes data
bronzefilter	Bronze Filter Section data
cells	Crick-Ripley biological cells data
chicago	Chicago street crimes
chorley	Chorley-Ribble cancer data

clmfires	Castilla-La Mancha forest fires
copper	Berman-Huntington copper deposits data
dendrite	Dendritic spines
demohyper	Synthetic point patterns
demopat	Synthetic point pattern
finpines	Finnish Pines data
flu	Influenza virus proteins
gordon	People in Gordon Square, London
gorillas	Gorilla nest sites
hamster	Aherne's hamster tumour data
humberside	North Humberside childhood leukaemia data
hyytiala	Mixed forest in Hyytiälä, Finland
japanesepines	Japanese Pines data
lansing	Lansing Woods data
longleaf	Longleaf Pines data
mucosa	Cells in gastric mucosa
murchison	Murchison gold deposits
nbfires	New Brunswick fires data
nztrees	Mark-Esler-Ripley trees data
osteo	Osteocyte lacunae (3D, replicated)
paracou	Kimboto trees in Paracou, French Guiana
ponderosa	Getis-Franklin ponderosa pine trees data
pyramidal	Pyramidal neurons from 31 brains
redwood	Strauss-Ripley redwood saplings data
redwoodfull	Strauss redwood saplings data (full set)
residualspaper	Data from Baddeley et al (2005)
shapley	Galaxies in an astronomical survey
simdat	Simulated point pattern (inhomogeneous, with interaction)
spiders	Spider webs on mortar lines of brick wall
sporophores	Mycorrhizal fungi around a tree
spruces	Spruce trees in Saxonia
swedishpines	Strand-Ripley Swedish pines data
urkiola	Urkiola Woods data
waka	Trees in Waka national park
waterstriders	Insects on water surface

To manipulate a point pattern:

plot.ppp	plot a point pattern (e.g. `plot(X)`)
iplot	plot a point pattern interactively
[.ppp	extract or replace a subset of a point pattern `pp[subset]` or `pp[subwindow]`
subset.ppp	extract subset of point pattern satisfying a condition
superimpose	combine several point patterns
by.ppp	apply a function to sub-patterns of a point pattern
cut.ppp	classify the points in a point pattern
split.ppp	divide pattern into sub-patterns
unmark	remove marks
npoints	count the number of points

coords	extract coordinates, change coordinates
marks	extract marks, change marks or attach marks
rotate	rotate pattern
shift	translate pattern
flipxy	swap x and y coordinates
reflect	reflect in the origin
periodify	make several translated copies
affine	apply affine transformation
scalardilate	apply scalar dilation
density.ppp	kernel estimation of point pattern intensity
Smooth.ppp	kernel smoothing of marks of point pattern
nnmark	mark value of nearest data point
sharpen.ppp	data sharpening
identify.ppp	interactively identify points
unique.ppp	remove duplicate points
duplicated.ppp	determine which points are duplicates
connected.ppp	find clumps of points
dirichlet	compute Dirichlet-Voronoi tessellation
delaunay	compute Delaunay triangulation
delaunayDistance	graph distance in Delaunay triangulation
convexhull	compute convex hull
discretise	discretise coordinates
pixellate.ppp	approximate point pattern by pixel image
as.im.ppp	approximate point pattern by pixel image

See spatstat.options to control plotting behaviour.

To create a window:

An object of class "owin" describes a spatial region (a window of observation).

owin	Create a window object
	owin(xlim, ylim) for rectangular window
	owin(poly) for polygonal window
	owin(mask) for binary image window
Window	Extract window of another object
Frame	Extract the containing rectangle ('frame') of another object
as.owin	Convert other data to a window object
square	make a square window
disc	make a circular window
ripras	Ripley-Rasson estimator of window, given only the points
convexhull	compute convex hull of something
letterR	polygonal window in the shape of the R logo
clickpoly	interactively draw a polygonal window
clickbox	interactively draw a rectangle

To manipulate a window:

plot.owin	plot a window.

	`plot(W)`
`boundingbox`	Find a tight bounding box for the window
`erosion`	erode window by a distance r
`dilation`	dilate window by a distance r
`closing`	close window by a distance r
`opening`	open window by a distance r
`border`	difference between window and its erosion/dilation
`complement.owin`	invert (swap inside and outside)
`simplify.owin`	approximate a window by a simple polygon
`rotate`	rotate window
`flipxy`	swap x and y coordinates
`shift`	translate window
`periodify`	make several translated copies
`affine`	apply affine transformation

Digital approximations:

`as.mask`	Make a discrete pixel approximation of a given window
`as.im.owin`	convert window to pixel image
`pixellate.owin`	convert window to pixel image
`commonGrid`	find common pixel grid for windows
`nearest.raster.point`	map continuous coordinates to raster locations
`raster.x`	raster x coordinates
`raster.y`	raster y coordinates
`as.polygonal`	convert pixel mask to polygonal window

See `spatstat.options` to control the approximation

Geometrical computations with windows:

`edges`	extract boundary edges
`intersect.owin`	intersection of two windows
`union.owin`	union of two windows
`setminus.owin`	set subtraction of two windows
`inside.owin`	determine whether a point is inside a window
`area.owin`	compute area
`perimeter`	compute perimeter length
`diameter.owin`	compute diameter
`incircle`	find largest circle inside a window
`connected.owin`	find connected components of window
`eroded.areas`	compute areas of eroded windows
`dilated.areas`	compute areas of dilated windows
`bdist.points`	compute distances from data points to window boundary
`bdist.pixels`	compute distances from all pixels to window boundary
`bdist.tiles`	boundary distance for each tile in tessellation
`distmap.owin`	distance transform image
`distfun.owin`	distance transform
`centroid.owin`	compute centroid (centre of mass) of window
`is.subset.owin`	determine whether one window contains another

is.convex	determine whether a window is convex
convexhull	compute convex hull
as.mask	pixel approximation of window
as.polygonal	polygonal approximation of window
is.rectangle	test whether window is a rectangle
is.polygonal	test whether window is polygonal
is.mask	test whether window is a mask
setcov	spatial covariance function of window
pixelcentres	extract centres of pixels in mask

Pixel images: An object of class "im" represents a pixel image. Such objects are returned by some of the functions in **spatstat** including Kmeasure, setcov and density.ppp.

im	create a pixel image
as.im	convert other data to a pixel image
pixellate	convert other data to a pixel image
as.matrix.im	convert pixel image to matrix
as.data.frame.im	convert pixel image to data frame
as.function.im	convert pixel image to function
plot.im	plot a pixel image on screen as a digital image
contour.im	draw contours of a pixel image
persp.im	draw perspective plot of a pixel image
rgbim	create colour-valued pixel image
hsvim	create colour-valued pixel image
[.im	extract a subset of a pixel image
[<-.im	replace a subset of a pixel image
rotate.im	rotate pixel image
shift.im	apply vector shift to pixel image
affine.im	apply affine transformation to image
X	print very basic information about image X
summary(X)	summary of image X
hist.im	histogram of image
mean.im	mean pixel value of image
integral.im	integral of pixel values
quantile.im	quantiles of image
cut.im	convert numeric image to factor image
is.im	test whether an object is a pixel image
interp.im	interpolate a pixel image
blur	apply Gaussian blur to image
Smooth.im	apply Gaussian blur to image
connected.im	find connected components
compatible.im	test whether two images have compatible dimensions
harmonise.im	make images compatible
commonGrid	find a common pixel grid for images
eval.im	evaluate any expression involving images
scaletointerval	rescale pixel values
zapsmall.im	set very small pixel values to zero
levelset	level set of an image

solutionset	region where an expression is true
imcov	spatial covariance function of image
convolve.im	spatial convolution of images
transect.im	line transect of image
pixelcentres	extract centres of pixels

Line segment patterns

An object of class "psp" represents a pattern of straight line segments.

psp	create a line segment pattern
as.psp	convert other data into a line segment pattern
edges	extract edges of a window
is.psp	determine whether a dataset has class "psp"
plot.psp	plot a line segment pattern
print.psp	print basic information
summary.psp	print summary information
[.psp	extract a subset of a line segment pattern
as.data.frame.psp	convert line segment pattern to data frame
marks.psp	extract marks of line segments
marks<-.psp	assign new marks to line segments
unmark.psp	delete marks from line segments
midpoints.psp	compute the midpoints of line segments
endpoints.psp	extract the endpoints of line segments
lengths.psp	compute the lengths of line segments
angles.psp	compute the orientation angles of line segments
superimpose	combine several line segment patterns
flipxy	swap x and y coordinates
rotate.psp	rotate a line segment pattern
shift.psp	shift a line segment pattern
periodify	make several shifted copies
affine.psp	apply an affine transformation
pixellate.psp	approximate line segment pattern by pixel image
as.mask.psp	approximate line segment pattern by binary mask
distmap.psp	compute the distance map of a line segment pattern
distfun.psp	compute the distance map of a line segment pattern
density.psp	kernel smoothing of line segments
selfcrossing.psp	find crossing points between line segments
crossing.psp	find crossing points between two line segment patterns
nncross	find distance to nearest line segment from a given point
nearestsegment	find line segment closest to a given point
project2segment	find location along a line segment closest to a given point
pointsOnLines	generate points evenly spaced along line segment
rpoisline	generate a realisation of the Poisson line process inside a window
rlinegrid	generate a random array of parallel lines through a window

Tessellations

An object of class "tess" represents a tessellation.

tess	create a tessellation
quadrats	create a tessellation of rectangles
hextess	create a tessellation of hexagons
quantess	quantile tessellation
as.tess	convert other data to a tessellation
plot.tess	plot a tessellation
tiles	extract all the tiles of a tessellation
[.tess	extract some tiles of a tessellation
[<-.tess	change some tiles of a tessellation
intersect.tess	intersect two tessellations
	or restrict a tessellation to a window
chop.tess	subdivide a tessellation by a line
dirichlet	compute Dirichlet-Voronoi tessellation of points
delaunay	compute Delaunay triangulation of points
rpoislinetess	generate tessellation using Poisson line process
tile.areas	area of each tile in tessellation
bdist.tiles	boundary distance for each tile in tessellation

Three-dimensional point patterns

An object of class "pp3" represents a three-dimensional point pattern in a rectangular box. The box is represented by an object of class "box3".

pp3	create a 3-D point pattern
plot.pp3	plot a 3-D point pattern
coords	extract coordinates
as.hyperframe	extract coordinates
subset.pp3	extract subset of 3-D point pattern
unitname.pp3	name of unit of length
npoints	count the number of points
runifpoint3	generate uniform random points in 3-D
rpoispp3	generate Poisson random points in 3-D
envelope.pp3	generate simulation envelopes for 3-D pattern
box3	create a 3-D rectangular box
as.box3	convert data to 3-D rectangular box
unitname.box3	name of unit of length
diameter.box3	diameter of box
volume.box3	volume of box
shortside.box3	shortest side of box
eroded.volumes	volumes of erosions of box

Multi-dimensional space-time point patterns

An object of class "ppx" represents a point pattern in multi-dimensional space and/or time.

ppx	create a multidimensional space-time point pattern
coords	extract coordinates
as.hyperframe	extract coordinates
subset.ppx	extract subset

unitname.ppx	name of unit of length
npoints	count the number of points
runifpointx	generate uniform random points
rpoisppx	generate Poisson random points
boxx	define multidimensional box
diameter.boxx	diameter of box
volume.boxx	volume of box
shortside.boxx	shortest side of box
eroded.volumes.boxx	volumes of erosions of box

Point patterns on a linear network

An object of class "linnet" represents a linear network (for example, a road network).

linnet	create a linear network
clickjoin	interactively join vertices in network
simplenet	simple example of network
lineardisc	disc in a linear network
delaunayNetwork	network of Delaunay triangulation
dirichletNetwork	network of Dirichlet edges
methods.linnet	methods for linnet objects

An object of class "lpp" represents a point pattern on a linear network (for example, road accidents on a road network).

lpp	create a point pattern on a linear network
methods.lpp	methods for lpp objects
subset.lpp	method for subset
rpoislpp	simulate Poisson points on linear network
runiflpp	simulate random points on a linear network
chicago	Chicago street crime data
dendrite	Dendritic spines data
spiders	Spider webs on mortar lines of brick wall

Hyperframes

A hyperframe is like a data frame, except that the entries may be objects of any kind.

hyperframe	create a hyperframe
as.hyperframe	convert data to hyperframe
plot.hyperframe	plot hyperframe
with.hyperframe	evaluate expression using each row of hyperframe
cbind.hyperframe	combine hyperframes by columns
rbind.hyperframe	combine hyperframes by rows
as.data.frame.hyperframe	convert hyperframe to data frame

Layered objects

A layered object represents data that should be plotted in successive layers, for example, a background and a foreground.

layered	create layered object
plot.layered	plot layered object
[.layered	extract subset of layered object

Colour maps

A colour map is a mechanism for associating colours with data. It can be regarded as a function, mapping data to colours. Using a colourmap object in a plot command ensures that the mapping from numbers to colours is the same in different plots.

colourmap	create a colour map
plot.colourmap	plot the colour map only
tweak.colourmap	alter individual colour values
interp.colourmap	make a smooth transition between colours
beachcolourmap	one special colour map

II. EXPLORATORY DATA ANALYSIS

Inspection of data:

summary(X)	print useful summary of point pattern X
X	print basic description of point pattern X
any(duplicated(X))	check for duplicated points in pattern X
istat(X)	Interactive exploratory analysis

Classical exploratory tools:

clarkevans	Clark and Evans aggregation index
fryplot	Fry plot
miplot	Morisita Index plot

Smoothing:

density.ppp	kernel smoothed density/intensity
relrisk	kernel estimate of relative risk
Smooth.ppp	spatial interpolation of marks
bw.diggle	cross-validated bandwidth selection for density.ppp
bw.ppl	likelihood cross-validated bandwidth selection for density.ppp
bw.scott	Scott's rule of thumb for density estimation
bw.relrisk	cross-validated bandwidth selection for relrisk
bw.smoothppp	cross-validated bandwidth selection for Smooth.ppp
bw.frac	bandwidth selection using window geometry
bw.stoyan	Stoyan's rule of thumb for bandwidth for pcf

Modern exploratory tools:

clusterset	Allard-Fraley feature detection
nnclean	Byers-Raftery feature detection
sharpen.ppp	Choi-Hall data sharpening
rhohat	Kernel estimate of covariate effect
rho2hat	Kernel estimate of covariate effect
spatialcdf	Spatial cumulative distribution function

Summary statistics for a point pattern: Type `demo(sumfun)` for a demonstration of many of the summary statistics.

intensity	Mean intensity
quadratcount	Quadrat counts
intensity.quadratcount	Mean intensity in quadrats
Fest	empty space function F
Gest	nearest neighbour distribution function G
Jest	J-function $J = (1 - G)/(1 - F)$
Kest	Ripley's K-function
Lest	Besag L-function
Tstat	Third order T-function
allstats	all four functions F, G, J, K
pcf	pair correlation function
Kinhom	K for inhomogeneous point patterns
Linhom	L for inhomogeneous point patterns
pcfinhom	pair correlation for inhomogeneous patterns
Finhom	F for inhomogeneous point patterns
Ginhom	G for inhomogeneous point patterns
Jinhom	J for inhomogeneous point patterns
localL	Getis-Franklin neighbourhood density function
localK	neighbourhood K-function
localpcf	local pair correlation function
localKinhom	local K for inhomogeneous point patterns
localLinhom	local L for inhomogeneous point patterns
localpcfinhom	local pair correlation for inhomogeneous patterns
Ksector	Directional K-function
Kscaled	locally scaled K-function
Kest.fft	fast K-function using FFT for large datasets
Kmeasure	reduced second moment measure
envelope	simulation envelopes for a summary function
varblock	variances and confidence intervals for a summary function
lohboot	bootstrap for a summary function

Related facilities:

plot.fv	plot a summary function
eval.fv	evaluate any expression involving summary functions
harmonise.fv	make functions compatible
eval.fasp	evaluate any expression involving an array of functions

with.fv	evaluate an expression for a summary function
Smooth.fv	apply smoothing to a summary function
deriv.fv	calculate derivative of a summary function
nndist	nearest neighbour distances
nnwhich	find nearest neighbours
pairdist	distances between all pairs of points
crossdist	distances between points in two patterns
nncross	nearest neighbours between two point patterns
exactdt	distance from any location to nearest data point
distmap	distance map image
distfun	distance map function
nnmap	nearest point image
nnfun	nearest point function
density.ppp	kernel smoothed density
Smooth.ppp	spatial interpolation of marks
relrisk	kernel estimate of relative risk
sharpen.ppp	data sharpening
rknn	theoretical distribution of nearest neighbour distance

Summary statistics for a multitype point pattern: A multitype point pattern is represented by an object X of class "ppp" such that marks(X) is a factor.

relrisk	kernel estimation of relative risk
scan.test	spatial scan test of elevated risk
Gcross,Gdot,Gmulti	multitype nearest neighbour distributions $G_{ij}, G_{i\bullet}$
Kcross,Kdot, Kmulti	multitype K-functions $K_{ij}, K_{i\bullet}$
Lcross,Ldot	multitype L-functions $L_{ij}, L_{i\bullet}$
Jcross,Jdot,Jmulti	multitype J-functions $J_{ij}, J_{i\bullet}$
pcfcross	multitype pair correlation function g_{ij}
pcfdot	multitype pair correlation function $g_{i\bullet}$
pcfmulti	general pair correlation function
markconnect	marked connection function p_{ij}
alltypes	estimates of the above for all i, j pairs
Iest	multitype I-function
Kcross.inhom,Kdot.inhom	inhomogeneous counterparts of Kcross, Kdot
Lcross.inhom,Ldot.inhom	inhomogeneous counterparts of Lcross, Ldot
pcfcross.inhom,pcfdot.inhom	inhomogeneous counterparts of pcfcross, pcfdot

Summary statistics for a marked point pattern: A marked point pattern is represented by an object X of class "ppp" with a component X$marks. The entries in the vector X$marks may be numeric, complex, string or any other atomic type. For numeric marks, there are the following functions:

markmean	smoothed local average of marks
markvar	smoothed local variance of marks
markcorr	mark correlation function
markvario	mark variogram
Kmark	mark-weighted K function

Emark	mark independence diagnostic $E(r)$
Vmark	mark independence diagnostic $V(r)$
nnmean	nearest neighbour mean index
nnvario	nearest neighbour mark variance index

For marks of any type, there are the following:

Gmulti	multitype nearest neighbour distribution
Kmulti	multitype K-function
Jmulti	multitype J-function

Alternatively use cut.ppp to convert a marked point pattern to a multitype point pattern.

Programming tools:

applynbd	apply function to every neighbourhood in a point pattern
markstat	apply function to the marks of neighbours in a point pattern
marktable	tabulate the marks of neighbours in a point pattern
pppdist	find the optimal match between two point patterns

Summary statistics for a point pattern on a linear network:

These are for point patterns on a linear network (class lpp). For unmarked patterns:

linearK	K function on linear network
linearKinhom	inhomogeneous K function on linear network
linearpcf	pair correlation function on linear network
linearpcfinhom	inhomogeneous pair correlation on linear network

For multitype patterns:

linearKcross	K function between two types of points
linearKdot	K function from one type to any type
linearKcross.inhom	Inhomogeneous version of linearKcross
linearKdot.inhom	Inhomogeneous version of linearKdot
linearmarkconnect	Mark connection function on linear network
linearmarkequal	Mark equality function on linear network
linearpcfcross	Pair correlation between two types of points
linearpcfdot	Pair correlation from one type to any type
linearpcfcross.inhom	Inhomogeneous version of linearpcfcross
linearpcfdot.inhom	Inhomogeneous version of linearpcfdot

Related facilities:

pairdist.lpp	distances between pairs
crossdist.lpp	distances between pairs
nndist.lpp	nearest neighbour distances
nncross.lpp	nearest neighbour distances

nnwhich.lpp	find nearest neighbours
nnfun.lpp	find nearest data point
distfun.lpp	distance transform
envelope.lpp	simulation envelopes
rpoislpp	simulate Poisson points on linear network
runiflpp	simulate random points on a linear network

It is also possible to fit point process models to lpp objects. See Section IV.

Summary statistics for a three-dimensional point pattern:

These are for 3-dimensional point pattern objects (class pp3).

F3est	empty space function F
G3est	nearest neighbour function G
K3est	K-function
pcf3est	pair correlation function

Related facilities:

envelope.pp3	simulation envelopes
pairdist.pp3	distances between all pairs of points
crossdist.pp3	distances between points in two patterns
nndist.pp3	nearest neighbour distances
nnwhich.pp3	find nearest neighbours
nncross.pp3	find nearest neighbours in another pattern

Computations for multi-dimensional point pattern:

These are for multi-dimensional space-time point pattern objects (class ppx).

pairdist.ppx	distances between all pairs of points
crossdist.ppx	distances between points in two patterns
nndist.ppx	nearest neighbour distances
nnwhich.ppx	find nearest neighbours

Summary statistics for random sets:

These work for point patterns (class ppp), line segment patterns (class psp) or windows (class owin).

Hest	spherical contact distribution H
Gfox	Foxall G-function
Jfox	Foxall J-function

III. MODEL FITTING (CLUSTER MODELS)

Cluster process models (with homogeneous or inhomogeneous intensity) and Cox processes can be fitted by the function kppm. Its result is an object of class "kppm". The fitted model can be printed, plotted, predicted, simulated and updated.

kppm	Fit model
plot.kppm	Plot the fitted model
fitted.kppm	Compute fitted intensity
predict.kppm	Compute fitted intensity
update.kppm	Update the model
improve.kppm	Refine the estimate of trend
simulate.kppm	Generate simulated realisations
vcov.kppm	Variance-covariance matrix of coefficients
Kmodel.kppm	K function of fitted model
pcfmodel.kppm	Pair correlation of fitted model

The theoretical models can also be simulated, for any choice of parameter values, using `rThomas`, `rMatClust`, `rCauchy`, `rVarGamma`, and `rLGCP`.

Lower-level fitting functions include:

lgcp.estK	fit a log-Gaussian Cox process model
lgcp.estpcf	fit a log-Gaussian Cox process model
thomas.estK	fit the Thomas process model
thomas.estpcf	fit the Thomas process model
matclust.estK	fit the Matern Cluster process model
matclust.estpcf	fit the Matern Cluster process model
cauchy.estK	fit a Neyman-Scott Cauchy cluster process
cauchy.estpcf	fit a Neyman-Scott Cauchy cluster process
vargamma.estK	fit a Neyman-Scott Variance Gamma process
vargamma.estpcf	fit a Neyman-Scott Variance Gamma process
mincontrast	low-level algorithm for fitting models by the method of minimum contrast

IV. MODEL FITTING (POISSON AND GIBBS MODELS)

Types of models

Poisson point processes are the simplest models for point patterns. A Poisson model assumes that the points are stochastically independent. It may allow the points to have a non-uniform spatial density. The special case of a Poisson process with a uniform spatial density is often called Complete Spatial Randomness.

Poisson point processes are included in the more general class of Gibbs point process models. In a Gibbs model, there is *interaction* or dependence between points. Many different types of interaction can be specified.

For a detailed explanation of how to fit Poisson or Gibbs point process models to point pattern data using **spatstat**, see Baddeley and Turner (2005b) or Baddeley (2008).

To fit a Poison or Gibbs point process model:

Model fitting in **spatstat** is performed mainly by the function `ppm`. Its result is an object of class `"ppm"`.

Here are some examples, where `X` is a point pattern (class `"ppp"`):

command *model*

`ppm(X)`	Complete Spatial Randomness
`ppm(X ~ 1)`	Complete Spatial Randomness
`ppm(X ~ x)`	Poisson process with intensity loglinear in x coordinate
`ppm(X ~ 1, Strauss(0.1))`	Stationary Strauss process
`ppm(X ~ x, Strauss(0.1))`	Strauss process with conditional intensity loglinear in x

It is also possible to fit models that depend on other covariates.

Manipulating the fitted model:

`plot.ppm`	Plot the fitted model
`predict.ppm`	Compute the spatial trend and conditional intensity of the fitted point process model
`coef.ppm`	Extract the fitted model coefficients
`formula.ppm`	Extract the trend formula
`intensity.ppm`	Compute fitted intensity
`Kmodel.ppm`	K function of fitted model
`pcfmodel.ppm`	pair correlation of fitted model
`fitted.ppm`	Compute fitted conditional intensity at quadrature points
`residuals.ppm`	Compute point process residuals at quadrature points
`update.ppm`	Update the fit
`vcov.ppm`	Variance-covariance matrix of estimates
`rmh.ppm`	Simulate from fitted model
`simulate.ppm`	Simulate from fitted model
`print.ppm`	Print basic information about a fitted model
`summary.ppm`	Summarise a fitted model
`effectfun`	Compute the fitted effect of one covariate
`logLik.ppm`	log-likelihood or log-pseudolikelihood
`anova.ppm`	Analysis of deviance
`model.frame.ppm`	Extract data frame used to fit model
`model.images`	Extract spatial data used to fit model
`model.depends`	Identify variables in the model
`as.interact`	Interpoint interaction component of model
`fitin`	Extract fitted interpoint interaction
`is.hybrid`	Determine whether the model is a hybrid
`valid.ppm`	Check the model is a valid point process
`project.ppm`	Ensure the model is a valid point process

For model selection, you can also use the generic functions `step`, `drop1` and `AIC` on fitted point process models.

See `spatstat.options` to control plotting of fitted model.

To specify a point process model:

The first order "trend" of the model is determined by an R language formula. The formula specifies the form of the *logarithm* of the trend.

`X ~ 1`	No trend (stationary)

`X ~ x`	Loglinear trend $\lambda(x, y) = \exp(\alpha + \beta x)$ where x, y are Cartesian coordinates
`X ~ polynom(x,y,3)`	Log-cubic polynomial trend
`X ~ harmonic(x,y,2)`	Log-harmonic polynomial trend
`X ~ Z`	Loglinear function of covariate `Z` $\lambda(x, y) = \exp(\alpha + \beta Z(x, y))$

The higher order ("interaction") components are described by an object of class `"interact"`. Such objects are created by:

`Poisson()`	the Poisson point process
`AreaInter()`	Area-interaction process
`BadGey()`	multiscale Geyer process
`Concom()`	connected component interaction
`DiggleGratton()`	Diggle-Gratton potential
`DiggleGatesStibbard()`	Diggle-Gates-Stibbard potential
`Fiksel()`	Fiksel pairwise interaction process
`Geyer()`	Geyer's saturation process
`Hardcore()`	Hard core process
`Hybrid()`	Hybrid of several interactions
`LennardJones()`	Lennard-Jones potential
`MultiHard()`	multitype hard core process
`MultiStrauss()`	multitype Strauss process
`MultiStraussHard()`	multitype Strauss/hard core process
`OrdThresh()`	Ord process, threshold potential
`Ord()`	Ord model, user-supplied potential
`PairPiece()`	pairwise interaction, piecewise constant
`Pairwise()`	pairwise interaction, user-supplied potential
`SatPiece()`	Saturated pair model, piecewise constant potential
`Saturated()`	Saturated pair model, user-supplied potential
`Softcore()`	pairwise interaction, soft core potential
`Strauss()`	Strauss process
`StraussHard()`	Strauss/hard core point process
`Triplets()`	Geyer triplets process

Note that it is also possible to combine several such interactions using `Hybrid`.

Finer control over model fitting:

A quadrature scheme is represented by an object of class `"quad"`. To create a quadrature scheme, typically use `quadscheme`.

`quadscheme`	default quadrature scheme using rectangular cells or Dirichlet cells
`pixelquad`	quadrature scheme based on image pixels
`quad`	create an object of class `"quad"`

To inspect a quadrature scheme:

`plot(Q)`	plot quadrature scheme Q
`print(Q)`	print basic information about quadrature scheme Q
`summary(Q)`	summary of quadrature scheme Q

A quadrature scheme consists of data points, dummy points, and weights. To generate dummy points:

`default.dummy`	default pattern of dummy points
`gridcentres`	dummy points in a rectangular grid
`rstrat`	stratified random dummy pattern
`spokes`	radial pattern of dummy points
`corners`	dummy points at corners of the window

To compute weights:

`gridweights`	quadrature weights by the grid-counting rule
`dirichletWeights`	quadrature weights are Dirichlet tile areas

Simulation and goodness-of-fit for fitted models:

`rmh.ppm`	simulate realisations of a fitted model
`simulate.ppm`	simulate realisations of a fitted model
`envelope`	compute simulation envelopes for a fitted model

Point process models on a linear network:

An object of class `"lpp"` represents a pattern of points on a linear network. Point process models can also be fitted to these objects. Currently only Poisson models can be fitted.

`lppm`	point process model on linear network
`anova.lppm`	analysis of deviance for point process model on linear network
`envelope.lppm`	simulation envelopes for point process model on linear network
`predict.lppm`	model prediction on linear network
`linim`	pixel image on linear network
`plot.linim`	plot a pixel image on linear network
`eval.linim`	evaluate expression involving images
`linfun`	function defined on linear network
`methods.linfun`	conversion facilities

V. MODEL FITTING (SPATIAL LOGISTIC REGRESSION)

Logistic regression

Pixel-based spatial logistic regression is an alternative technique for analysing spatial point patterns that is widely used in Geographical Information Systems. It is approximately equivalent to fitting a Poisson point process model.

In pixel-based logistic regression, the spatial domain is divided into small pixels, the presence or absence of a data point in each pixel is recorded, and logistic regression is used to model the presence/absence indicators as a function of any covariates.

Facilities for performing spatial logistic regression are provided in **spatstat** for comparison purposes.

Fitting a spatial logistic regression

Spatial logistic regression is performed by the function `slrm`. Its result is an object of class `"slrm"`. There are many methods for this class, including methods for `print`, `fitted`, `predict`, `simulate`, `anova`, `coef`, `logLik`, `terms`, `update`, `formula` and `vcov`.

For example, if `X` is a point pattern (class `"ppp"`):

command	*model*
`slrm(X ~ 1)`	Complete Spatial Randomness
`slrm(X ~ x)`	Poisson process with intensity loglinear in x coordinate
`slrm(X ~ Z)`	Poisson process with intensity loglinear in covariate `Z`

Manipulating a fitted spatial logistic regression

`anova.slrm`	Analysis of deviance
`coef.slrm`	Extract fitted coefficients
`vcov.slrm`	Variance-covariance matrix of fitted coefficients
`fitted.slrm`	Compute fitted probabilities or intensity
`logLik.slrm`	Evaluate loglikelihood of fitted model
`plot.slrm`	Plot fitted probabilities or intensity
`predict.slrm`	Compute predicted probabilities or intensity with new data
`simulate.slrm`	Simulate model

There are many other undocumented methods for this class, including methods for `print`, `update`, `formula` and `terms`. Stepwise model selection is possible using `step` or `stepAIC`.

VI. SIMULATION

There are many ways to generate a random point pattern, line segment pattern, pixel image or tessellation in **spatstat**.

Random point patterns:

`runifpoint`	generate n independent uniform random points
`rpoint`	generate n independent random points
`rmpoint`	generate n independent multitype random points
`rpoispp`	simulate the (in)homogeneous Poisson point process
`rmpoispp`	simulate the (in)homogeneous multitype Poisson point process
`runifdisc`	generate n independent uniform random points in disc
`rstrat`	stratified random sample of points
`rsyst`	systematic random sample (grid) of points
`rMaternI`	simulate the Matérn Model I inhibition process

rMaternII	simulate the Matérn Model II inhibition process
rSSI	simulate Simple Sequential Inhibition process
rStrauss	simulate Strauss process (perfect simulation)
rNeymanScott	simulate a general Neyman-Scott process
rMatClust	simulate the Matérn Cluster process
rThomas	simulate the Thomas process
rLGCP	simulate the log-Gaussian Cox process
rGaussPoisson	simulate the Gauss-Poisson cluster process
rCauchy	simulate Neyman-Scott process with Cauchy clusters
rVarGamma	simulate Neyman-Scott process with Variance Gamma clusters
rcell	simulate the Baddeley-Silverman cell process
runifpointOnLines	generate n random points along specified line segments
rpoisppOnLines	generate Poisson random points along specified line segments

Resampling a point pattern:

quadratresample	block resampling
rjitter	apply random displacements to points in a pattern
rshift	random shifting of (subsets of) points
rthin	random thinning

See also varblock for estimating the variance of a summary statistic by block resampling, and lohboot for another bootstrap technique.

Fitted point process models:

If you have fitted a point process model to a point pattern dataset, the fitted model can be simulated.

Cluster process models are fitted by the function kppm yielding an object of class "kppm". To generate one or more simulated realisations of this fitted model, use simulate.kppm.

Gibbs point process models are fitted by the function ppm yielding an object of class "ppm". To generate a simulated realisation of this fitted model, use rmh. To generate one or more simulated realisations of the fitted model, use simulate.ppm.

Other random patterns:

rlinegrid	generate a random array of parallel lines through a window
rpoisline	simulate the Poisson line process within a window
rpoislinetess	generate random tessellation using Poisson line process
rMosaicSet	generate random set by selecting some tiles of a tessellation
rMosaicField	generate random pixel image by assigning random values in each tile of a tessellation

Simulation-based inference

envelope	critical envelope for Monte Carlo test of goodness-of-fit
qqplot.ppm	diagnostic plot for interpoint interaction
scan.test	spatial scan statistic/test

VII. TESTS AND DIAGNOSTICS

Classical hypothesis tests:

quadrat.test	χ^2 goodness-of-fit test on quadrat counts
clarkevans.test	Clark and Evans test
cdf.test	Spatial distribution goodness-of-fit test
berman.test	Berman's goodness-of-fit tests
envelope	critical envelope for Monte Carlo test of goodness-of-fit
scan.test	spatial scan statistic/test
dclf.test	Diggle-Cressie-Loosmore-Ford test
mad.test	Mean Absolute Deviation test
dclf.progress	Progress plot for DCLF test
mad.progress	Progress plot for MAD test
anova.ppm	Analysis of Deviance for point process models

Sensitivity diagnostics:

Classical measures of model sensitivity such as leverage and influence have been adapted to point process models.

leverage.ppm	Leverage for point process model
influence.ppm	Influence for point process model
dfbetas.ppm	Parameter influence

Diagnostics for covariate effect:

Classical diagnostics for covariate effects have been adapted to point process models.

parres	Partial residual plot
addvar	Added variable plot
rhohat	Kernel estimate of covariate effect
rho2hat	Kernel estimate of covariate effect (bivariate)

Residual diagnostics:

Residuals for a fitted point process model, and diagnostic plots based on the residuals, were introduced in Baddeley et al (2005) and Baddeley, Rubak and Møller (2011).

Type `demo(diagnose)` for a demonstration of the diagnostics features.

diagnose.ppm	diagnostic plots for spatial trend
qqplot.ppm	diagnostic Q-Q plot for interpoint interaction
residualspaper	examples from Baddeley et al (2005)
Kcom	model compensator of K function
Gcom	model compensator of G function
Kres	score residual of K function
Gres	score residual of G function
psst	pseudoscore residual of summary function
psstA	pseudoscore residual of empty space function

psstG	pseudoscore residual of G function
compareFit	compare compensators of several fitted models

Resampling and randomisation procedures

You can build your own tests based on randomisation and resampling using the following capabilities:

quadratresample	block resampling
rjitter	apply random displacements to points in a pattern
rshift	random shifting of (subsets of) points
rthin	random thinning

VIII. DOCUMENTATION

The online manual entries are quite detailed and should be consulted first for information about a particular function.

The paper by Baddeley and Turner (2005a) is a brief overview of the package. Baddeley and Turner (2005b) is a more detailed explanation of how to fit point process models to data. Baddeley (2010) is a complete set of notes from a 2-day workshop on the use of **spatstat**.

Type `citation("spatstat")` to get these references.

Licence

Acknowledgements

Kasper Klitgaard Berthelsen, Abdollah Jalilian, Marie-Colette van Lieshout, Tuomas Rajala, Dominic Schuhmacher and Rasmus Waagepetersen made substantial contributions of code.

Additional contributions by Ang Qi Wei, Sandro Azaele, Malissa Baddeley, Colin Beale, Melanie Bell, Thomas Bendtsen, Ricardo Bernhardt, Andrew Bevan, Brad Biggerstaff, Anders Bilgrau, Leanne Bischof, Roger Bivand, Jose M. Blanco Moreno, Florent Bonneu, Julian Burgos, Simon Byers, Ya-Mei Chang, Jianbao Chen, Igor Chernayavsky, Y.C. Chin, Bjarke Christensen, Jean-Francois Coeurjolly, Robin Corria Ainslie, Marcelino de la Cruz, Peter Dalgaard, Sourav Das, Peter Diggle, Patrick Donnelly, Ian Dryden, Stephen Eglen, Ahmed El-Gabbas, Olivier Flores, David Ford, Peter Forbes, Shane Frank, Janet Franklin, Funwi-Gabga Neba, Oscar Garcia, Agnes Gault, Marc Genton, Shaaban Ghalandarayeshi, Julian Gilbey, Jason Goldstick, Pavel Grabarnik, C. Graf, Ute Hahn, Andrew Hardegen, Martin Bøgsted Hansen, Martin Hazelton, Juha Heikkinen, Mandy Hering, Markus Herrmann, Kurt Hornik, Philipp Hunziker, Jack Hywood, Ross Ihaka, Aruna Jammalamadaka, Robert John-Chandran, Devin Johnson, Mahdieh Khanmohammadi, Peter Kovesi, Mike Kuhn, Jeff Laake, Frederic Lavancier, Tom Lawrence, Robert Lamb, Jonathan Lee, George Leser, Li Haitao, George Limitsios, Andrew Lister, Ben Madin, Kiran Marchikanti, Jeff Marcus, Robert Mark, Peter McCullagh, Monia Mahling, Jorge Mateu Mahiques, Ulf Mehlig, Sebastian Wastl Meyer, Mi Xiangcheng, Lore De Middeleer, Robin Milne, Enrique Miranda, Jesper Møller, Erika Mudrak, Gopalan Nair, Nicoletta Nava, Linda Stougaard Nielsen, Felipe Nunes, Jens Randel Nyengaard, Jens Oehlschlägel, Thierry Onkelinx, Sean O'Riordan, Evgeni Parilov, Jeff Picka,

Nicolas Picard, Mike Porter, Sergiy Protsiv, Adrian Raftery, Suman Rakshit, Pablo Ramon, Xavier Raynaud, Matt Reiter, Ian Renner, Tom Richardson, Brian Ripley, Ted Rosenbaum, Barry Rowlingson, Jason Rudokas, John Rudge, Christopher Ryan, Farzaneh Safavimanesh, Aila Särkkä, Cody Schank, Katja Schladitz, Sebastian Schutte, Bryan Scott, Olivia Semboli, Vadim Shcherbakov, Shen Guochun, Ida-Maria Sintorn, Yong Song, Malte Spiess, Mark Stevenson, Kaspar Stucki, Michael Sumner, P. Surovy, Ben Taylor, Thordis Linda Thorarinsdottir, Berwin Turlach, Torben Tvedebrink, Kevin Ummer, Medha Uppala, Andrew van Burgel, Tobias Verbeke, Mikko Vihtakari, Alexandre Villers, Fabrice Vinatier, Sasha Voss, Hao Wang, H. Wendrock, Jan Wild, Carl G. Witthoft, Selene Wong, Maxime Woringer, Mike Zamboni and Achim Zeileis.

Author(s)

Adrian Baddeley <Adrian.Baddeley@uwa.edu.au> http://www.maths.uwa.edu.au/~adrian/, Rolf Turner <r.turner@auckland.ac.nz> and Ege Rubak <rubak@math.aau.dk>.

References

Baddeley, A. (2010) *Analysing spatial point patterns in R*. Workshop notes. Version 4.1. CSIRO online technical publication. URL: www.uwa.edu.au/resources/pf16h.html

Baddeley, A. and Turner, R. (2005a) Spatstat: an R package for analyzing spatial point patterns. *Journal of Statistical Software* **12**:6, 1–42. URL: www.jstatsoft.org, ISSN: 1548-7660.

Baddeley, A. and Turner, R. (2005b) Modelling spatial point patterns in R. In: A. Baddeley, P. Gregori, J. Mateu, R. Stoica, and D. Stoyan, editors, *Case Studies in Spatial Point Pattern Modelling*, Lecture Notes in Statistics number 185. Pages 23–74. Springer-Verlag, New York, 2006. ISBN: 0-387-28311-0.

Baddeley, A., Turner, R., Møller, J. and Hazelton, M. (2005) Residual analysis for spatial point processes. *Journal of the Royal Statistical Society, Series B* **67**, 617–666.

Baddeley, A., Rubak, E. and Møller, J. (2011) Score, pseudo-score and residual diagnostics for spatial point process models. *Statistical Science* **26**, 613–646.

Baddeley, A., Turner, R., Mateu, J. and Bevan, A. (2013) Hybrids of Gibbs point process models and their implementation. *Journal of Statistical Software* **55**:11, 1–43. http://www.jstatsoft.org/v55/i11/

Diggle, P.J. (2003) *Statistical analysis of spatial point patterns*, Second edition. Arnold.

Gelfand, A.E., Diggle, P.J., Fuentes, M. and Guttorp, P., editors (2010) *Handbook of Spatial Statistics*. CRC Press.

Huang, F. and Ogata, Y. (1999) Improvements of the maximum pseudo-likelihood estimators in various spatial statistical models. *Journal of Computational and Graphical Statistics* **8**, 510–530.

Waagepetersen, R. An estimating function approach to inference for inhomogeneous Neyman-Scott processes. *Biometrics* **63** (2007) 252–258.

Index