Package ‘spatstat’

November 20, 2024

Version 3.3-0

Date 2024-11-20

Title Spatial Point Pattern Analysis, Model-Fitting, Simulation, Tests

Maintainer Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>

Depends R (>= 3.5.0), spatstat.data (>= 3.1-4), spatstat.univar (>= 3.1-1), spatstat.geom (>= 3.3-4), spatstat.random (>= 3.3-2), spatstat.explore (>= 3.3-3), spatstat.model (>= 3.3-3), spatstat.linnet (>= 3.2-2), utils

Imports spatstat.utils (>= 3.1-1)

Description Comprehensive open-source toolbox for analysing Spatial Point Patterns. Focused mainly on two-dimensional point patterns, including multitype/marked points, in any spatial region. Also supports three-dimensional point patterns, space-time point patterns in any number of dimensions, point patterns on a linear network, and patterns of other geometrical objects. Supports spatial covariate data such as pixel images.
Contains over 3000 functions for plotting spatial data, exploratory data analysis, model-fitting, simulation, spatial sampling, model diagnostics, and formal inference.
Data types include point patterns, line segment patterns, spatial windows, pixel images, tessellations, and linear networks.
Exploratory methods include quadrat counts, K-functions and their simulation envelopes, nearest neighbour distance and empty space statistics, Fry plots, pair correlation function, kernel smoothed intensity, relative risk estimation with cross-validated bandwidth selection, mark correlation functions, segregation indices, mark dependence diagnostics, and kernel estimates of covariate effects. Formal hypothesis tests of random pattern (chi-squared, Kolmogorov-Smirnov, Monte Carlo, Diggle-Cressie-Loosmore-Ford, Dao-Genton, two-stage Monte Carlo) and tests for covariate effects (Cox-Berman-Waller-Lawson, Kolmogorov-Smirnov, ANOVA) are also supported.
Parametric models can be fitted to point pattern data using the functions ppm(), kppm(), slrm(), dppm() similar to glm(). Types of models include Poisson, Gibbs and Cox point processes, Neyman-Scott cluster processes, and determinantal point processes. Models may involve dependence on covariates, inter-point interaction, cluster formation and dependence on marks. Models are fitted by maximum likelihood, logistic regression, minimum contrast, and composite likelihood methods.
A model can be fitted to a list of point patterns (replicated point pattern data) using the function mppm(). The model can include random effects and fixed effects depending on the experimental design, in addition to all the features listed above.

Fitted point process models can be simulated, automatically. Formal hypothesis tests of a fitted model are supported (likelihood ratio test, analysis of deviance, Monte Carlo tests) along with basic tools for model selection (stepwise(), AIC()) and variable selection (sdr). Tools for validating the fitted model include simulation envelopes, residuals, residual plots and Q-Q plots, leverage and influence diagnostics, partial residuals, and added variable plots.

License GPL (>= 2)

URL http://spatstat.org/

NeedsCompilation yes

ByteCompile true

BugReports https://github.com/spatstat/spatstat/issues

Author Adrian Baddeley [aut, cre] (<https://orcid.org/0000-0001-9499-8382>),
Rolf Turner [aut] (<https://orcid.org/0000-0001-5521-5218>),
Ege Rubak [aut] (<https://orcid.org/0000-0002-6675-533X>)

Repository CRAN

Date/Publication 2024-11-20 01:10:02 UTC

Contents

`spatstat-package` *The Spatstat Package*

Description

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

Details

spatstat is a family of R packages for the statistical analysis of spatial data. Its main focus is the analysis of spatial patterns of points in two-dimensional space.

spatstat is designed to support a complete statistical analysis of spatial data. It supports

- creation, manipulation and plotting of point patterns;

- exploratory data analysis;
- spatial random sampling;
- simulation of point process models;
- parametric model-fitting;
- non-parametric smoothing and regression;
- formal inference (hypothesis tests, confidence intervals);
- 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 point process models (by Waagepetersen's two-step fitting procedure and minimum contrast, composite likelihood, or Palm likelihood)
- determinantal point process models (by Waagepetersen's two-step fitting procedure and minimum contrast, composite likelihood, or Palm 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 read that document, you can either

- visit https://cran.r-project.org/package=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`.

Once you have installed **spatstat**, start R and type `library(spatstat)`. Then type `beginner` for a beginner's introduction, or `demo(spatstat)` for a demonstration of the package's capabilities.

For a complete course on **spatstat**, and on statistical analysis of spatial point patterns, read the book by Baddeley, Rubak and Turner (2015). Other recommended books on spatial point process methods are Diggle (2014), Gelfand et al (2010) and Illian et al (2008).

The **spatstat** package includes over 50 datasets, which can be useful when learning the package. Type `demo(data)` to see plots of all datasets available in the package. Type `vignette('datasets')` for detailed background information on these datasets, and plots of each dataset.

For information on converting your data into **spatstat** format, read Chapter 3 of Baddeley, Rubak and Turner (2015). This chapter is available free online, as one of the sample chapters at the book companion website, `https://book.spatstat.org/`.

Structure of the spatstat family

The original **spatstat** package grew to be very large. It has now been divided into several **sub-packages**:

- **spatstat.utils** containing basic utilities
- **spatstat.sparse** containing linear algebra utilities
- **spatstat.data** containing datasets
- **spatstat.univar** containing functions for estimating probability distributions of random variables
- **spatstat.geom** containing functionality for geometrical operations, and defining the main classes of spatial objects
- **spatstat.explore** containing the main functions for exploratory analysis of spatial data
- **spatstat.model** containing the main functions for parametric statistical modelling and analysis, and formal inference, for spatial data
- **spatstat.linnet** containing functions for spatial data on a linear network
- **spatstat**, which simply loads the other sub-packages listed above, and provides documentation.

The breakup has been done in such a way that the user should not notice any difference. Source code that worked with the old **spatstat** package should work with the new **spatstat** family. Code that is documented in our books, journal articles and vignettes should still work.

When you install **spatstat**, the sub-packages listed above are also installed. Then if you load the **spatstat** package by typing `library(spatstat)`, the other sub-packages listed above will automatically be loaded or imported.

This help file covers all the functionality and datasets that are provided in the sub-packages listed above.

Extension packages

Additionally there are several **extension packages:**

- **spatstat.gui** for interactive graphics
- **spatstat.local** for local likelihood (including geographically weighted regression)
- **spatstat.Knet** for additional, computationally efficient code for linear networks
- **spatstat.sphere** (under development) for spatial data on a sphere, including spatial data on the earth's surface

The extension packages must be installed separately and loaded explicitly if needed. They also have separate documentation.

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 book (Baddeley, Rubak and Turner, 2015) was published. It is accessible as `vignette('updates')`.

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)` or `?name` where `name` is the name of the function or dataset.

CONTENTS:

I.	Creating and manipulating data
II.	Exploratory Data Analysis
III.	Model fitting (Cox and cluster models)
IV.	Model fitting (Poisson and Gibbs models)
V.	Model fitting (determinantal point processes)
VI.	Model fitting (spatial logistic regression)
VII.	Simulation
VIII.	Tests and diagnostics
IX.	Documentation

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)
rStraussHard	simulate Strauss-hard core process (perfect simulation)
rDiggleGratton	simulate Diggle-Gratton process (perfect simulation)
rDGS	simulate Diggle-Gates-Stibbard process (perfect simulation)
rPenttinen	simulate Penttinen 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 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 display a 2D point pattern:

`plot.ppp`	plot a point pattern (e.g. `plot(X)`)
`spatstat.gui::iplot`	plot a point pattern interactively
`persp.ppp`	perspective plot of marked point pattern

To manipulate a 2D point pattern:

`edit.ppp`	interactive text editor
`[.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
`densityHeat.ppp`	diffusion 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
`uniquemap.ppp`	map duplicated points to unique points
`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
ellipse	make an elliptical 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
as.data.frame.owin	convert window to data frame

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
raster.xy	raster x and 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
inradius	radius of incircle
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
triangulate.owin	decompose into triangles
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
clickdist	measure distance between two points clicked by user

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
blurHeat	apply diffusion blur to image
Smooth.im	apply Gaussian blur to image
SmoothHeat.im	apply diffusion 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
im.apply	evaluate a function of several 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
transmat	convert matrix of pixel values to a different indexing convention
rnoise	random pixel noise

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
subset.psp	extract subset of 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
psp2mask	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
selfcut.psp	cut segments where they cross
crossing.psp	find crossing points between two line segment patterns
extrapolate.psp	extrapolate line segments to infinite lines
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
polartess	tessellation using polar coordinates

quantess	quantile tessellation
venn.tess	Venn diagram tessellation
dirichlet	compute Dirichlet-Voronoi tessellation of points
delaunay	compute Delaunay triangulation of points
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
rpoislinetess	generate tessellation using Poisson line process
tile.areas	area of each tile in tessellation
bdist.tiles	boundary distance for each tile in tessellation
connected.tess	find connected components of tiles
shift.tess	shift a tessellation
rotate.tess	rotate a tessellation
reflect.tess	reflect about the origin
flipxy.tess	reflect about the diagonal
affine.tess	apply affine transformation

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
spatstat.gui::iplot.linnet	interactively plot 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
vertices.linnet	nodes of network
joinVertices	join existing vertices in a network
insertVertices	insert new vertices at positions along a network
addVertices	add new vertices, extending a network
thinNetwork	remove vertices or lines from a network
repairNetwork	repair internal format
pixellate.linnet	approximate by pixel image

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 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
subset.hyperframe	method for subset
head.hyperframe	first few rows of hyperframe
tail.hyperframe	last few rows of hyperframe

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
spatstat.gui::istat(X)	Interactive exploratory analysis
spatstat.gui::View.ppp(X)	spreadsheet-style viewer

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
relriskHeat	diffusion estimate of relative risk
Smooth.ppp	spatial interpolation of marks
SmoothHeat.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.CvL	Cronie-Van Lieshout bandwidth selection for density estimation
bw.scott	Scott's rule of thumb for density estimation
bw.abram	Abramson's rule for adaptive bandwidths
bw.relrisk	cross-validated bandwidth selection for relrisk
bw.relriskHeatppp	cross-validated bandwidth selection for relriskHeat.ppp
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 effect of two covariates
spatialcdf	Spatial cumulative distribution function
roc	Receiver operating characteristic curve

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
pool.fv	pool several estimates 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
densityHeat.ppp	diffusion 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
localKcross,localKdot	local counterparts of Kcross, Kdot
localLcross,localLdot	local counterparts of Lcross, Ldot
localKcross.inhom,localLcross.inhom	local counterparts of Kcross.inhom, Lcross.inhom

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
markcrosscorr	mark cross-correlation function
markvario	mark variogram
markmarkscatter	mark-mark scatterplot
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
density.lpp	kernel smoothing estimator of intensity
densityHeat.lpp	diffusion kernel estimate
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 (COX AND 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
summary.kppm	Summarise 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
coef.kppm	Extract trend coefficients
formula.kppm	Extract trend formula
parameters	Extract all model parameters
clusterfield.kppm	Compute offspring density
clusterradius.kppm	Radius of support of offspring density
Kmodel.kppm	K function of fitted model
pcfmodel.kppm	Pair correlation of fitted model

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

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 Matérn Cluster process model
matclust.estpcf	fit the Matérn 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 Poisson 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

parameters	Extract all model parameters
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. For variable selection, see sdr.

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
HierHard()	Hierarchical multiype hard core process
HierStrauss()	Hierarchical multiype Strauss process
HierStraussHard()	Hierarchical multiype Strauss-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
Penttinen()	Penttinen pairwise interaction
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
fitted.lppm	fitted intensity values
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 (DETERMINANTAL POINT PROCESS MODELS)

Code for fitting *determinantal point process models* has recently been added to **spatstat**.

For information, see the help file for dppm.

VI. 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. For variable selection, see sdr.

VII. 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
rHardcore	simulate hard core process (perfect simulation)
rStrauss	simulate Strauss process (perfect simulation)

rStraussHard	simulate Strauss-hard core process (perfect simulation)
rDiggleGratton	simulate Diggle-Gratton process (perfect simulation)
rDGS	simulate Diggle-Gates-Stibbard process (perfect simulation)
rPenttinen	simulate Penttinen 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
bits.envelope	critical envelope for balanced two-stage Monte Carlo test
qqplot.ppm	diagnostic plot for interpoint interaction
scan.test	spatial scan statistic/test
studpermu.test	studentised permutation test

segregation.test	test of segregation of types

VIII. TESTS AND DIAGNOSTICS

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
anova.ppm	Analysis of Deviance for point process models

More recently-developed tests:

dg.test	Dao-Genton test
bits.test	Balanced independent two-stage test
dclf.progress	Progress plot for DCLF test
mad.progress	Progress plot for MAD test

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
dffit.ppm	Effect change diagnostic

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

IX. DOCUMENTATION

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

The book Baddeley, Rubak and Turner (2015) is a complete course on analysing spatial point patterns, with full details about **spatstat**.

Older material (which is now out-of-date but is freely available) includes Baddeley and Turner (2005a), a brief overview of the package in its early development; Baddeley and Turner (2005b), a more detailed explanation of how to fit point process models to data; and Baddeley (2010), a complete set of notes from a 2-day workshop on the use of **spatstat**.

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

Licence

Acknowledgements

Kasper Klitgaard Berthelsen, Ottmar Cronie, Tilman Davies, Yongtao Guan, Ute Hahn, Abdollah Jalilian, Marie-Colette van Lieshout, Greg McSwiggan, Tuomas Rajala, Suman Rakshit, Dominic Schuhmacher, Rasmus Waagepetersen and Hangsheng Wang made substantial contributions of code.

Additional contributions and suggestions from Monsuru Adepeju, Corey Anderson, Ang Qi Wei, Ryan Arellano, Jens Åström, Robert Aue, Marcel Austenfeld, Sandro Azaele, Malissa Baddeley,

Guy Bayegnak, Colin Beale, Melanie Bell, Thomas Bendtsen, Ricardo Bernhardt, Andrew Bevan, Brad Biggerstaff, Anders Bilgrau, Leanne Bischof, Christophe Biscio, Roger Bivand, Jose M. Blanco Moreno, Florent Bonneu, Jordan Brown, Ian Buller, Julian Burgos, Simon Byers, Ya-Mei Chang, Jianbao Chen, Igor Chernayavsky, Y.C. Chin, Bjarke Christensen, Lucía Cobo Sanchez, Jean-François Coeurjolly, Kim Colyvas, Hadrien Commenges, Rochelle Constantine, Robin Corria Ainslie, Richard Cotton, Marcelino de la Cruz, Peter Dalgaard, Mario D'Antuono, Sourav Das, Peter Diggle, Patrick Donnelly, Ian Dryden, Stephen Eglen, Ahmed El-Gabbas, Belarmain Fandohan, Olivier Flores, David Ford, Peter Forbes, Shane Frank, Janet Franklin, Funwi-Gabga Neba, Oscar Garcia, Agnes Gault, Jonas Geldmann, 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, Maximilian Hesselbarth, Paul Hewson, Hamidreza Heydarian, Kassel Hingee, Kurt Hornik, Philipp Hunziker, Jack Hywood, Ross Ihaka, Čenk Içös, Aruna Jammalamadaka, Robert John-Chandran, Devin Johnson, Mahdieh Khanmohammadi, Bob Klaver, Lily Kozmian-Ledward, Peter Kovesi, Mike Kuhn, Jeff Laake, Robert Lamb, Frédéric Lavancier, Tom Lawrence, Tomas Lazauskas, Jonathan Lee, George Leser, Angela Li, Li Haitao, George Limitsios, Andrew Lister, Nestor Luambua, Bethany Macdonald, Ben Madin, Martin Maechler, Daniel Manrique-Castaño, Kiran Marchikanti, Jeff Marcus, Robert Mark, Peter McCullagh, Monia Mahling, Jorge Mateu Mahiques, Ulf Mehlig, Frederico Mestre, Sebastian Wastl Meyer, Mi Xiangcheng, Lore De Middeleer, Robin Milne, Enrique Miranda, Jesper Møller, Annie Mollié, Ines Moncada, Mehdi Moradi, Virginia Morera Pujol, Erika Mudrak, Gopalan Nair, Nader Najari, Nicoletta Nava, Linda Stougaard Nielsen, Felipe Nunes, Jens Randel Nyengaard, Jens Oehlschlägel, Thierry Onkelinx, Sean O'Riordan, Evgeni Parilov, Jeff Picka, Nicolas Picard, Tim Pollington, Mike Porter, Sergiy Protsiv, Adrian Raftery, Suman Rakshit, Ben Ramage, Pablo Ramon, Xavier Raynaud, Nicholas Read, Matt Reiter, Ian Renner, Tom Richardson, Brian Ripley, Yonatan Rosen, Ted Rosenbaum, Barry Rowlingson, Jason Rudokas, Tyler Rudolph, John Rudge, Christopher Ryan, Farzaneh Safavimanesh, Aila Särkkä, Cody Schank, Katja Schladitz, Sebastian Schutte, Bryan Scott, Olivia Semboli, François Sémécurbe, Vadim Shcherbakov, Shen Guochun, Shi Peijian, Harold-Jeffrey Ship, Tammy L Silva, Ida-Maria Sintorn, Yong Song, Malte Spiess, Mark Stevenson, Kaspar Stucki, Jan Sulavik, Michael Sumner, P. Surovy, Ben Taylor, Thordis Linda Thorarinsdottir, Leigh Torres, Berwin Turlach, Torben Tvedebrink, Kevin Ummer, Medha Uppala, Andrew van Burgel, Tobias Verbeke, Mikko Vihtakari, Alexendre Villers, Fabrice Vinatier, Maximilian Vogtland, Sasha Voss, Sven Wagner, Hao Wang, H. Wendrock, Jan Wild, Carl G. Witthoft, Selene Wong, Maxime Woringer, Luke Yates, Mike Zamboni and Achim Zeileis.

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>, Rolf Turner <rolfturner@posteo.net> and Ege Rubak <rubak@math.aau.dk>.

References

Baddeley, A. (2010) *Analysing spatial point patterns in R*. Workshop notes, Version 4.1. Online technical publication, CSIRO. https://research.csiro.au/software/wp-content/uploads/sites/6/2015/02/Rspatialcourse_CMIS_PDF-Standard.pdf

Baddeley, A., Rubak, E. and Turner, R. (2015) *Spatial Point Patterns: Methodology and Applications with R*. Chapman and Hall/CRC Press.

Baddeley, A. and Turner, R. (2005a) Spatstat: an R package for analyzing spatial point patterns. *Journal of Statistical Software* **12**:6, 1–42. DOI: 10.18637/jss.v012.i06.

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. https://www.jstatsoft.org/v55/i11/

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

Diggle, P.J. (2014) *Statistical Analysis of Spatial and Spatio-Temporal Point Patterns*, Third edition. Chapman and Hall/CRC.

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.

Illian, J., Penttinen, A., Stoyan, H. and Stoyan, D. (2008) *Statistical Analysis and Modelling of Spatial Point Patterns.* Wiley.

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

`beginner` *Print Introduction For Beginners*

Description

Prints an introduction for beginners to the `spatstat` package, or another specified package.

Usage

```
beginner(package = "spatstat")
```

Arguments

`package`	Name of package.

Details

This function prints an introduction for beginners to the **spatstat** package.

The function can be executed simply by typing `beginner` without parentheses.

If the argument `package` is given, then the function prints the beginner's help file `BEGINNER.txt` from the specified package (if it has one).

Value

Null.

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>

and Rolf Turner <rolfturner@posteo.net>

See Also

latest.news

Examples

```
beginner
```

---

bugfixes   *List Recent Bug Fixes*

Description

List all bug fixes in a package, starting from a certain date or version of the package. Fixes are sorted alphabetically by the name of the affected function. The default is to list bug fixes in the latest version of the **spatstat** family of packages.

Usage

```
bugfixes(sinceversion = NULL, sincedate = NULL,
         package = spatstat.family(), show = TRUE)
```

Arguments

sinceversion	Earliest version of package for which bugs should be listed. A character string. The default is the current installed version.
sincedate	Earliest release date of package for which bugs should be listed. A character string or a date-time object.
package	The name of the package (or packages) for which bugs are to be listed. A character string or character vector.
show	Logical value indicating whether to display the bug table on the terminal.

Details

Bug reports are extracted from the NEWS file of the specified package. Only those after a specified date, or after a specified version of the package, are retained. The bug reports are then sorted alphabetically, so that all bugs affecting a particular function are listed consecutively. Finally the table of bug reports is displayed (if `show=TRUE`) and returned invisibly.

The argument `sinceversion` should be a character string like `"1.2-3"`. The default is the current installed version of the package.

The argument `sincedate` should be a character string like `"2015-05-27"`, or a date-time object.

If `sinceversion="all"` or `sincedate="all"` then all recorded bugs will be listed.

The special options `sinceversion="book"` and `sincedate="book"` are interpreted to mean `sincedate="2015-06-05"`, which gives all bugs reported after publication of the book by Baddeley, Rubak and Turner (2015).

Typing `bugfixes` without parentheses will display a table of all bugs that were fixed in the current installed version of **spatstat** and its sub-packages.

By default, bugs in the *extension* packages **spatstat.local**, **spatstat.Knet**, **spatstat.gui** are *not* reported. To include these bugs as well, set `package=spatstat.family(TRUE,TRUE)`.

Value

(Invisibly) a data frame, belonging to the class `"bugtable"`, which has a `print` method.

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>.

References

Baddeley, A., Rubak, E. and Turner, R. (2015) *Spatial Point Patterns: Methodology and Applications with R*. Chapman and Hall/CRC Press.

See Also

`latest.changes`, `latest.news`, `news`.

Examples

```
bugfixes
## show all bugs reported after publication of the spatstat book
if(interactive()) bugfixes(sinceversion="book")
```

`foo` *Foo is Not a Real Name*

Description

The name `foo` is not a real name: it is a place holder, used to represent the name of any desired thing.

The functions defined here simply print an explanation of the placeholder name `foo`.

Usage

```
foo()

## S3 method for class 'foo'
plot(x, ...)
```

Arguments

`x`	Ignored.
`...`	Ignored.

Details

The name `foo` is used by computer scientists as a *place holder*, to represent the name of any desired object or function. It is not the name of an actual object or function; it serves only as an example, to explain a concept.

However, many users misinterpret this convention, and actually type the command `foo` or `foo()`. Then they email the package author to inform them that `foo` is not defined.

To avoid this correspondence, we have now defined an object called `foo`.

The function `foo()` prints a message explaining that `foo` is not really the name of a variable.

The function can be executed simply by typing `foo` without parentheses.

Value

Null.

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>, Rolf Turner <rolfturner@posteo.net> and Ege Rubak <rubak@math.aau.dk>.

See Also

beginner

Examples

```
foo
```

---

latest.changes *List Recent Significant Changes to a Function*

Description

List all the changes to a particular function in a package, starting from a certain date or version of the package. The default is to list changes in the latest version of the **spatstat** family of packages.

Usage

```
latest.changes(x,
        sinceversion = NULL, sincedate = NULL,
        package = spatstat.family(), show = TRUE)
```

Arguments

`x`	Character string giving the name of the function of interest, or a search pattern to be matched. A character vector is permitted.
`sinceversion`	Earliest version of `package` for which changes should be listed. A character string. Default is the most recent version.
`sincedate`	Earliest release date of `package` for which changes should be listed. A character string or a date-time object. Default is the date of the most recent version.
`package`	The name of the package (or packages) for which changes are to be listed. A character string or character vector.
`show`	Logical value indicating whether to display the table of changes on the terminal.

Details

Details of changes are extracted from the NEWS file of the specified `package` under the heading 'Significant User-Visible Changes'. All entries for which the first line matches `x` are selected. The table of changes is displayed (if `show=TRUE`) and returned invisibly.

The argument `sinceversion` should be a character string like `"1.2-3"`. The default is the version string of the most recent version.

The argument `sincedate` should be a character string like `"2015-05-27"`, or a date-time object. The default is the date of the most recent version.

If `sinceversion="all"` or `sincedate="all"` then all recorded changes will be listed.

The special options `sinceversion="book"` and `sincedate="book"` are interpreted to mean `sincedate="2015-06-05"`, which gives all changes reported after publication of the book by Baddeley, Rubak and Turner (2015).

By default, changes in the *extension* packages **spatstat.local**, **spatstat.Knet**, **spatstat.gui** are *not* reported. To include these changes as well, set `package=spatstat.family(TRUE,TRUE)`.

Value

(Invisibly) a data frame, belonging to the class "changetable", which has a print method.

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>.

References

Baddeley, A., Rubak, E. and Turner, R. (2015) *Spatial Point Patterns: Methodology and Applications with R*. Chapman and Hall/CRC Press.

See Also

bugfixes, latest.news, news.

Examples

```
latest.changes("plot.symbolmap")
```

---

latest.news — *Print News About Latest Version of Package*

Description

Prints the news documentation for the current version of spatstat or another specified package.

Usage

```
latest.news(package = spatstat.family(), doBrowse=FALSE, major=TRUE)
```

Arguments

package	Name of package for which the latest news should be printed. A character string, or vector of character strings.
doBrowse	Logical value indicating whether to display the results in a browser window instead of printing them.
major	Logical value. If TRUE (the default), print all information for the current major version "x.y". If FALSE, print only the information for the current minor version "x.y-z".

Details

This function prints the news documentation about changes in the current installed version of a package.

By default, it prints the latest news about all the sub-packages in the **spatstat** family.

The function can be called simply by typing its name without parentheses (see the Examples).

If `major=FALSE`, only information for the current minor version `"x.y-z"` will be printed. If `major=TRUE` (the default), all information for the current major version `"x.y"` will be printed, encompassing versions `"x.y-0"`, `"x.y-1"`, up to `"x.y-z"`.

If `package` is given, then the function reads the news for the specified package from its `NEWS` file (if it has one) and prints only the entries that refer to the current version of the package.

To see the news for all previous versions as well as the current version, use the R utility `news`. See the Examples.

Value

Null.

Author(s)

Adrian Baddeley `<Adrian.Baddeley@curtin.edu.au>`, Rolf Turner `<rolfturner@posteo.net>` and Ege Rubak `<rubak@math.aau.dk>`.

See Also

`spatstat.family` lists the packages in the **spatstat** family.

`bugfixes` lists bug fixes.

`news`

Examples

```
if(interactive()) {

  # current news
  latest.news

  # all news
  # news(package="spatstat")

}
```

`spatstat.family` *Names of All Packages in the Spatstat Family*

Description

Provides the names of all the packages belonging to the **spatstat** family of packages.

Usage

```
spatstat.family(subpackages=TRUE, extensions=FALSE)
```

Arguments

`subpackages`	Logical value specifying whether to include sub-packages.
`extensions`	Logical value specifying whether to include extension packages.

Details

This function returns a character vector containing the names of the packages that belong to the **spatstat** family.

By default, only the sub-packages are listed, and not the extension packages.

A "sub-package" is a package which is implicitly loaded or imported when the command `library(spatstat)` is issued. Currently the sub-packages are:

- `spatstat.utils`
- `spatstat.data`
- `spatstat.univar`
- `spatstat.sparse`
- `spatstat.geom`
- `spatstat.random`
- `spatstat.explore`
- `spatstat.model`
- `spatstat.linnet`
- `spatstat`

An "extension package" is a package which must be loaded explicitly. The extension packages are:

- `spatstat.gui`
- `spatstat.local`
- `spatstat.Knet`

Value

Character vector of package names.

Author(s)

Adrian Baddeley <Adrian.Baddeley@curtin.edu.au>, Rolf Turner <rolfturner@posteo.net> and Ege Rubak <rubak@math.aau.dk>.

See Also

latest.news

Index