Analysing spatial point patterns in R

Adrian Baddeley
Adrian.Baddeley@csiro.au
adrian@maths.uwa.edu.au

Workshop Notes
February 2008

Abstract

This is a detailed set of notes for a workshop on *Analysing spatial point patterns* that has been held several times in Australia and New Zealand in 2006–2008.

It covers statistical methods that are currently feasible in practice and available in public domain software. Some of these techniques are well established in the applications literature, while some are very recent developments.

The workshop uses the statistical package R and is based on `spatstat`, an add-on library for R for the analysis of spatial data.

Topics covered include: statistical formulation and methodological issues; data input and handling; R concepts such as classes and methods; nonparametric intensity estimates; goodness-of-fit testing for Complete Spatial Randomness; maximum likelihood inference for Poisson processes; model validation for Poisson processes; distance methods and summary functions such as Ripley's K function; non-Poisson point process models; simulation techniques; fitting models using summary statistics; Gibbs point process models; fitting Gibbs models; simulating Gibbs models; validating Gibbs models; multitype and marked point patterns; exploratory analysis of marked point patterns; multitype Poisson process models and maximum likelihood inference; multitype Gibbs process models and maximum pseudolikelihood; and line segment data.

This workshop requires R **version 2.6.0** or later, and `spatstat` **version 1.12-6** or later.

Acknowledgements

The author gratefully acknowledges countless comments and suggestions from workshop participants, and the support of CSIRO Mathematical and Information Sciences, The New Zealand Statistical Association, The University of Waikato, The Statistical Society of Australia and The University of Western Australia.

Contents

1 Introduction **5**

2 Statistical formulation **12**

3 The R system **16**

4 Introduction to spatstat **18**

5 Objects, classes and methods **25**

6 Data input **31**

7 Methods 1: Investigating intensity **36**

8 Defining the window **40**

9 Manipulating point patterns **45**

10 Methods 2: Tests of Complete Spatial Randomness **53**

11 Methods 3: Maximum likelihood for Poisson processes **58**

12 Methods 4: checking a fitted Poisson model **67**

13 Images in spatstat **74**

14 Simple models of non-Poisson patterns **79**

15 Methods 5: Distance methods for point patterns **83**

16 Methods 6: inference using summary statistics **98**

17 Methods 7: adjusting for inhomogeneity **105**

18 Gibbs models **109**

19 Methods 8: fitting Gibbs models **116**

20 Methods 9: validation of fitted Gibbs models **125**

21 Marked point patterns **129**

22 Handling marked point pattern data **133**

23 Methods 10: exploratory tools for marked point patterns **138**

24 Methods 11: multitype Poisson models **151**

25 Methods 12: Gibbs models for multitype point patterns **157**

26 Line segment data **162**

27 Further information on `spatstat` **164**

Bibliography **165**

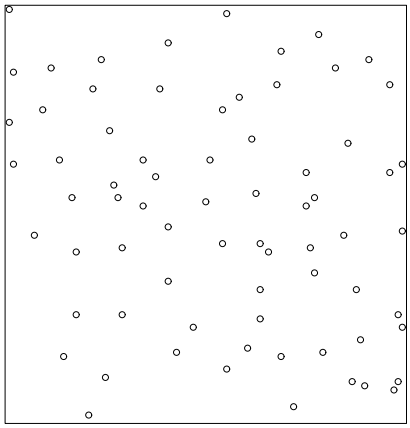
Index **167**

1 Introduction

1.1 Types of data

1.1.1 Points

A **point pattern** dataset gives the locations of objects/events occurring in a study region.

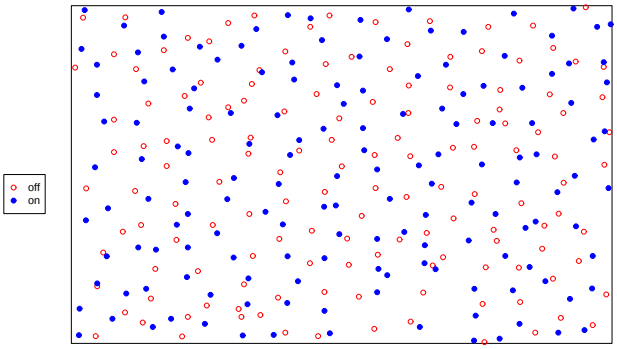


The points could represent trees, animal nests, earthquake epicentres, petty crimes, domiciles of new cases of influenza, galaxies, etc.

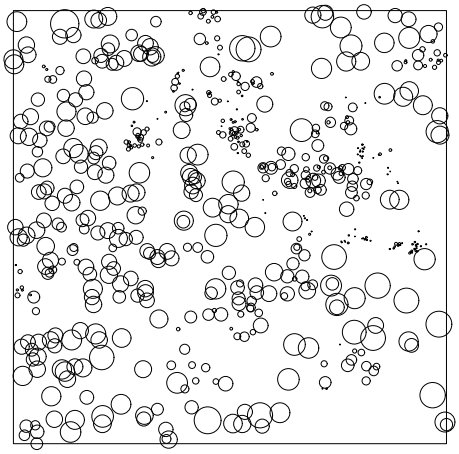
The points might be situated in a region of the two-dimensional (2D) plane, or on the Earth's surface, or a 3D volume, etc. They could be points in space-time (e.g. earthquake epicentre location and time). The software presented here is only applicable to 2D point patterns (but we're working on it).

1.1.2 Marks

The points may have extra information called **marks** attached to them. The mark represents an "attribute" of the point. The mark variable could be *categorical*, e.g. species or disease status:



The mark variable could be *continuous*, e.g. tree diameter:

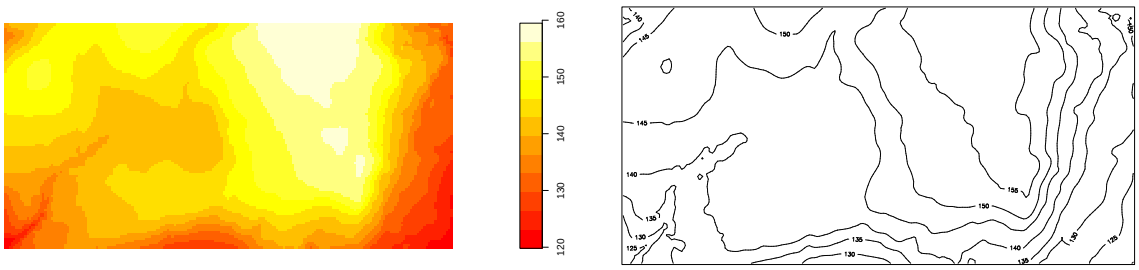


The mark could be multivariate, or even more complicated.

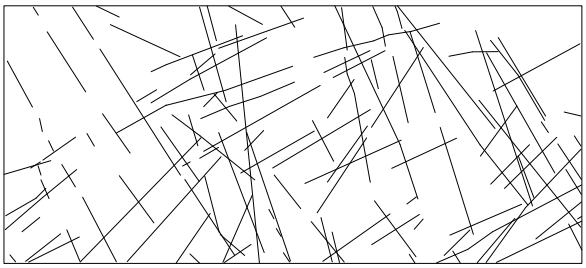
1.1.3 Covariates

Our dataset may also include **covariates** — any data that we treat as explanatory, rather than as part of the 'response'.

Covariate data may be a *spatial function* $Z(u)$ defined at all spatial locations u, e.g. altitude, soil pH, displayed as a pixel image or a contour plot:



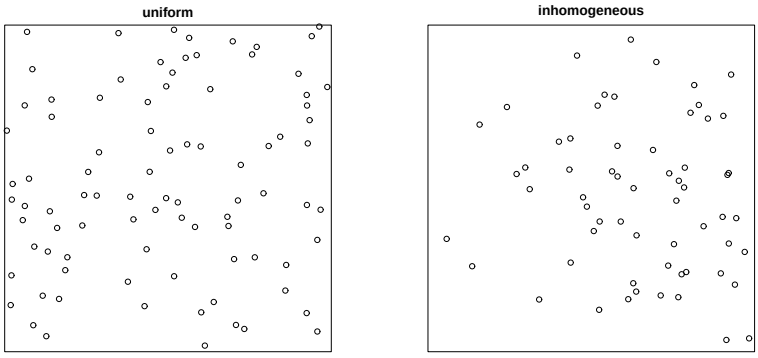
Covariate data may be another *spatial pattern* such as another point pattern, or a line segment pattern, e.g. a map of geological faults:



1.2 Typical scientific questions

1.2.1 Intensity

'Intensity' is the average density of points (expected number of points per unit area). Intensity may be constant ('uniform') or may vary from location to location ('non-uniform' or 'inhomogeneous').



16 Methods 6: inference using summary statistics

Although summary statistics such as the K-function are intended primarily for exploratory purposes, it is also possible to use them as a basis for parameter estimation and hypothesis testing.

16.1 Envelopes and Monte Carlo tests

The graphical comparison of $\widehat{K}$ with $K_{\rm pois}$, etc, can be formalised in terms of hypothesis testing. The null hypothesis is Complete Spatial Randomness (a homogeneous Poisson process) and the alternative comprises all other processes.

16.1.1 Pointwise Monte Carlo test

Following Besag [14] and Ripley [36, 38], formal hypothesis tests are conducted using the *Monte Carlo test* principle [25, 15] rather than the Neyman-Pearson lemma. Suppose the reference curve is the theoretical K function for a completely random (uniform Poisson) point process. Generate M independent simulations of this process inside the study region W. Compute the estimated K functions for each of these realisations, say $\widehat{K}^{(j)}(r)$ for $j = 1, \ldots, M$. Obtain the pointwise upper and lower envelopes of these simulated curves,

$$\begin{aligned} L(r) &= \min_j \widehat{K}^{(j)}(r) \\ U(r) &= \max_j \widehat{K}^{(j)}(r). \end{aligned}$$

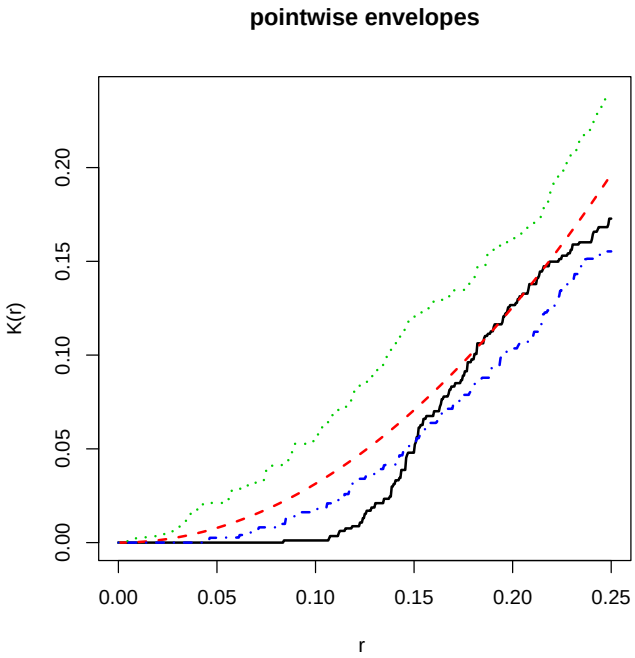
For any fixed value of r, consider the probability that $\widehat{K}(r)$ lies outside the envelope $[L(r), U(r)]$ for the simulated curves. If the data came from a uniform Poisson process, then $\widehat{K}(r)$ and $\widehat{K}^{(1)}(r), \ldots, \widehat{K}^{(M)}(r)$ are statistically equivalent and independent, so this probability is equal to $2/(M+1)$ by symmetry. That is, the test which rejects the null hypothesis of a uniform Poisson process when $\widehat{K}(r)$ lies outside $[L(r), U(r)]$, has exact significance level $\alpha = 2/(M+1)$. Instead of the pointwise maximum and minimum, one could use the pointwise order statistics (the pointwise kth largest and k smallest values) giving a test of exact size $\alpha = 2k/(M+1)$.

16.1.2 Envelopes in `spatstat`

In `spatstat` the function `envelope` computes the pointwise envelopes.

```
> data(cells)
> E <- envelope(cells, Kest, nsim = 39, rank = 1)

> E

Pointwise critical envelopes for K(r)
Obtained from 39 simulations of simulations of CSR
Significance level of pointwise Monte Carlo test: 2/40 = 0.05
Data: cells
Function value object (class 'fv')
for the function r -> K(r)
Entries:
id      label     description
--      -----     -----------
```

```
r       r         distance argument r
obs     obs(r)    function value for data pattern
theo    theo(r)   theoretical value for CSR
lo      lo(r)     lower pointwise envelope of simulations
hi      hi(r)     upper pointwise envelope of simulations
----------------------------------------

Default plot formula:
        . ~ r

Recommended range of argument r: [0, 0.25]

> plot(E, main = "pointwise envelopes")
```



For example if r had been fixed at $r = 0.10$ we would have rejected the null hypothesis of CSR at the 5% level. The value $M = 39$ is the smallest to yield a two-sided test with significance level 5%.

> Tip: A common and dangerous mistake is to misinterpret the simulation envelopes as "confidence intervals" around $\hat{K}$. They cannot be interpreted as a measure of accuracy of the estimated K function! They are the critical values for a test of the hypothesis that $K(r) = \pi r^2$.

16.1.3 Simultaneous Monte Carlo test

Note that the theory of the Monte Carlo test, as presented above, requires that r be fixed in advance. If we plot the envelope and check whether the empirical K function ever wanders outside the envelope, this is equivalent to choosing the value of r in a data-dependent way, and the true significance level is higher (less 'significant').

To avoid this problem we can construct *simultaneous critical bands* which have the property that, under H_0, the probability that $\widehat{K}$ ever wanders outside the critical bands is exactly 5%.

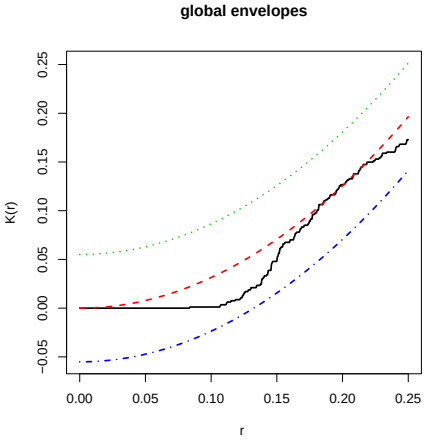
One simple way to achieve this is to compute, for each estimate $\widehat{K}(r)$, its maximum deviation from the Poisson K function, $D = \max_r |\widehat{K}(r) - K_{\rm pois}(r)|$. This is computed for each of the M simulated datasets, and the maximum value $D_{\rm max}$ obtained. Then the upper and lower limits are

$$\begin{aligned} L(r) &= \pi r^2 - D_{\rm max} \\ U(r) &= \pi r^2 + D_{\rm max}. \end{aligned}$$

The estimated K function for the data transgresses these limits if and only if the D-value for the data exceeds $D_{\rm max}$. Under H_0 this occurs with probability $1/(M+1)$. Thus, a test of size 5% is obtained by taking $M = 19$.

```
> E <- envelope(cells, Kest, nsim = 19, rank = 1, global = TRUE)
```

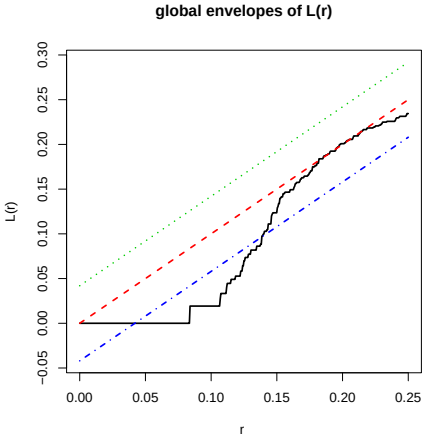
```
> plot(E, main = "global envelopes")
```



A more powerful test is obtained if we (approximately) stabilise the variance, by using the L function in place of K.

```
> E <- envelope(cells, Lest, nsim = 19, rank = 1, global = TRUE)
```

```
> plot(E, main = "global envelopes of L(r)")
```



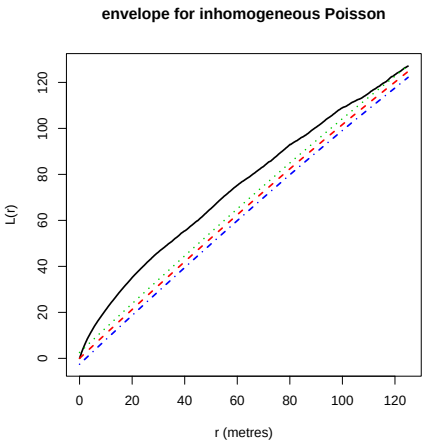
16.1.4 Envelopes for any fitted model

In the explanation above, we assumed that the null hypothesis was CSR (complete spatial randomness, a uniform Poisson process). In fact the Monte Carlo testing rationale can be applied to any point process model serving as a null hypothesis. We simply have to generate simulated realisations from the null hypothesis, and compute the summary function for each simulated realisation.

To simulate from a fitted point process model (object of class `"ppm"`), call the `envelope` function, giving the fitted model as the first argument of `envelope`. Then the simulated patterns will be generated according to this fitted model. The original data point pattern, to which the model was fitted, is stored in the fitted model object; the original data are extracted and the summary function for the data is also computed.

The following code fits an inhomogeneous Poisson process to the Beilschmiedia pattern, then generates simulation envelopes of the L function by simulating from the fitted inhomogeneous Poisson model.

```
> data(bei)
> fit <- ppm(bei, ~elev + grad, covariates = bei.extra)
> E <- envelope(fit, Lest, nsim = 19, global = TRUE, correction = "border")

> plot(E, main = "envelope for inhomogeneous Poisson")
```



16.1.5 Envelopes based on any simulation procedure

Envelopes can also be computed using any user-specified procedure to generate the simulated realisations. This allows us to perform randomisation tests, for example.

The simulation procedure should be encoded as an R expression, which will be evaluated each time a simulation is required. For example if we type

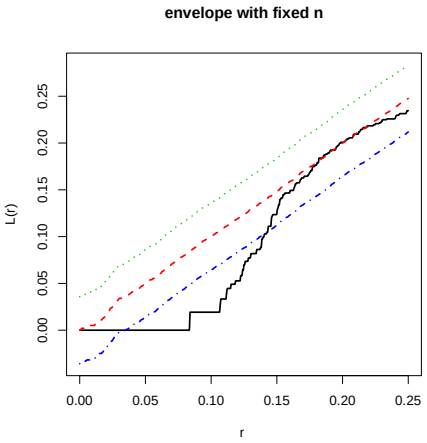
```
> sim <- expression(rpoispp(100))
```

then each time the expression `sim` is evaluated, it will yield a different random outcome of the Poisson process with intensity 100 in the unit square.

This expression should be passed to the `envelope` function as the argument `simulate`.

The following code generates simulation envelopes for the L function based on simulations of CSR which have the same number of points as the data pattern.

```
> data(cells)
> e <- expression(runifpoint(cells$n, cells$window))
> E <- envelope(cells, Lest, nsim = 19, global = TRUE, simulate = e)

> plot(E, main = "envelope with fixed n")
```



16.1.6 Envelopes based on a set of point patterns

Envelopes can also be computed from a user-supplied list of point patterns, instead of the simulated point patterns generated by a chosen simulation procedure.

This improves efficiency and consistency if, for example, we are going to calculate the envelopes of several different summary statistics.

```
> data(cells)
> SimPatList <- list()
> for (i in 1:1000) SimPatList[[i]] <- runifpoint(cells$n)
> EK <- envelope(cells, Kest, simulate = SimPatList, nsim = 1000)
> Ep <- envelope(cells, pcf, simulate = SimPatList, nsim = 1000)
```

16.2 Model-fitting using summary statistics

In the 'method of moments' we estimate a parameter θ by solving

$$\mathbb{E}_\theta[S(\mathbf{X})] = S(\mathbf{x})$$

where $S(\mathbf{x})$ is the observed value of a statistic S for our data $\mathbf{x}$, and the left side is the theoretical mean of S for the model governed by parameter θ.

The analogue for point process models is to fit the model by matching a summary statistic such as the K function to its theoretical value under the model.

16.2.1 Theoretical mean known analytically

In a precious few cases, the K function of a point process is known exactly as an analytic expression in terms of the model parameters. These include many Neyman-Scott processes. For example, the K-function of the Thomas process with parameters $\theta = (\kappa, \mu, \sigma)$ is

$$K_\theta(r) = \pi r^2 + \frac{1}{\kappa}(1 - \exp(-\frac{r^2}{4\sigma^2})). \tag{26}$$

We may thus fit a Thomas model by solving $K_\theta(r) = \widehat{K}(r)$ for some values of r. More efficiently we choose θ to minimise the discrepancy between the two functions over some range $[a, b]$:

$$D = \int_a^b \left| \widehat{K}(r)^q - K_\theta(r)^q \right|^p \, \mathrm{d}r \tag{27}$$

where $0 \le a < b$, and where $p, q > 0$ are indices. This method was originally advocated by Peter Diggle and collaborators, and is now known as the *method of minimum contrast*. See [21].

To fit the Thomas model by minimum contrast to the K function, use `thomas.estK`.

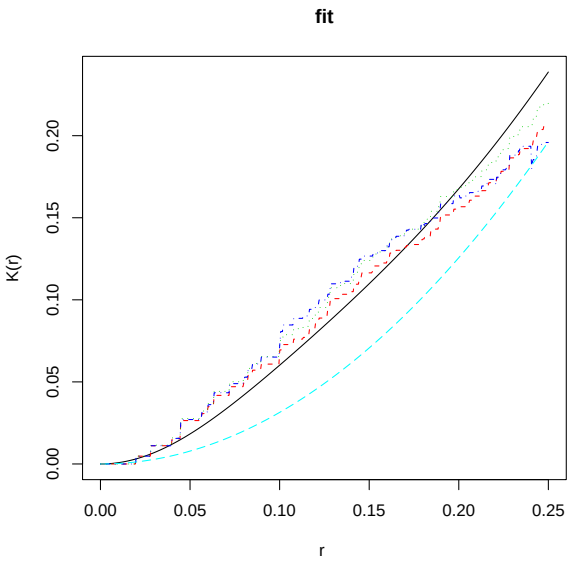
```
> data(redwood)
> fit <- thomas.estK(redwood, c(kappa = 10, sigma2 = 0.1))
```

The second argument to `thomas.estK` gives a set of starting values for the parameters, used in the minimisation search.

The fitted model, `fit`, is an object of class `minconfit`. There are methods for printing and plotting objects of this class.

```
> fit

Minimum contrast fit (object of class "minconfit")
Model: Thomas process
Fitted by matching theoretical K function to Kest(redwood)
Parameters fitted by minimum contrast ($par):
       kappa       sigma2
23.545183910  0.002214530
Derived parameters of Thomas process ($modelpar):
      kappa       sigma          mu
23.54518391  0.04705879  2.63323490
Converged successfully after 139 iterations.
Domain of integration: [ 0 , 0.25 ]
Exponents: p= 2, q= 0.25
```

```
> plot(fit)
```



The plot shows the theoretical K function of the fitted Thomas process (`fit`), three non-parametric estimates of the K function (`iso, trans, border`) and the Poisson K function (`theo`).

Other models can be fitted using `matclust.estK` (Matérn cluster process), `lgcp.estK` (log-Gaussian Cox process), or `mincontrast` (generic fitting algorithm for method of minimum contrast).

16.2.2 Monte Carlo

For the vast majority of point process models, the true K function $K_\theta(r)$ is not known analytically in terms of the parameter θ. In principle we could use Monte Carlo simulation to determine an approximation to $K_\theta(r)$, for any given θ, by generating a large number of simulated realisations of the process with parameter θ, computing the estimated K-function for each realisation, and taking the pointwise sample average. It's possible to do this in `spatstat` using the generic algorithm `mincontrast`. Details are not given here as it is rather fiddly at present, and will change soon.

17 Methods 7: adjusting for inhomogeneity

If a point pattern is known or suspected to be spatially inhomogeneous, then our statistical analysis of the pattern should take account of this inhomogeneity.

17.1 Inhomogeneous K function

There is a modification of the K function that applies to inhomogeneous processes [2]. If $\lambda(u)$ is the true intensity function of the point process $\mathbf{X}$, then the idea is that each point x_i will be weighted by $w_i = 1/\lambda(x_i)$.

The *inhomogeneous K-function* is defined as

$$K_{\mathrm{inhom}}(r) = \mathbb{E}\left[\frac{1}{\lambda(u)}\sum_{x_j \in \mathbf{X}} \frac{1}{\lambda(x_j)} \mathbf{1}\left\{0 < ||u - x_j|| \le r\right\} \;\middle|\; u \in \mathbf{X}\right] \tag{28}$$

assuming that this does not depend on location u. Thus, $\lambda(u)K(r)$ is the expected total 'weight' of all random points within a distance r of the point u, where the 'weight' of a point x_i is $1/\lambda(x_i)$.

If the process is actually homogeneous, then $\lambda(u)$ is constant and $K_{\mathrm{inhom}}(r)$ reduces to the usual K function (21).

It turns out that, for an inhomogeneous Poisson process with intensity function $\lambda(u)$, the inhomogeneous K function is

$$K_{\mathrm{inhom,\ pois}}(r) = \pi r^2 \tag{29}$$

exactly as for the homogeneous case.

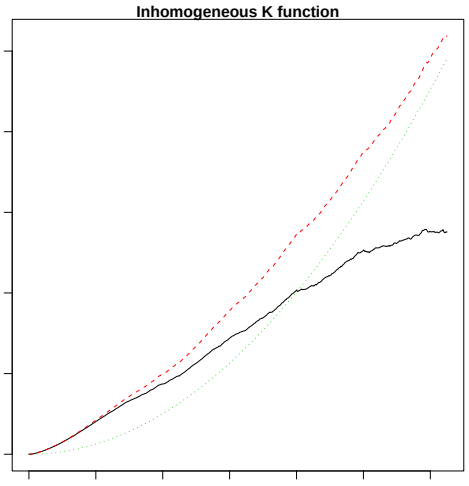
The standard estimators of K can be extended to the inhomogeneous K function:

$$\widehat{K}_{\mathrm{inhom}}(r) = \frac{1}{\mathrm{area}(W)} \sum_i \sum_{j \ne i} \frac{\mathbf{1}\left\{||x_i - x_j|| \le r\right\}}{\widehat{\lambda}(x_i)\widehat{\lambda}(x_j)} e(x_i, x_j; r) \tag{30}$$

where $e(u, v, r)$ is an edge correction weight as before, and $\widehat{\lambda}(u)$ is an estimate of the intensity function $\lambda(u)$.

There remains the question of how to estimate the intensity function $\lambda(u)$. It is usually advisable to obtain the intensity estimate $\widehat{\lambda}(u)$ by fitting a parametric model, to avoid overfitting. Here is an example for the tropical rainforest data, using the covariate data to suggest a model for the intensity.

```
> data(bei)
> fit <- ppm(bei, ~elev + grad, covariates = bei.extra)
> lam <- predict(fit, locations = bei)
> Ki <- Kinhom(bei, lam)
> plot(Ki, main = "Inhomogeneous K function")
```



The plot suggests that, even after accounting for dependence on altitude and slope, the trees still appear to be clustered.

The intensity function $\lambda(u)$ could also be estimated by kernel smoothing the point pattern data. However, notice that the estimator (30) of the inhomogeneous K function depends on the estimated intensity values at the *data points*, $\widehat{\lambda}(x_i)$. These are positively biased estimates of the true values $\lambda(x_i)$. In order to avoid bias, the value $\widehat{\lambda}(x_i)$ should be estimated by kernel smoothing of the point pattern with the point x_i *deleted*. This "leave-one-out" estimator is implemented in `Kinhom` and is invoked when the argument `lambda` is not given:

```
> Ki2 <- Kinhom(bei)
> plot(Ki2, main = "Kinhom using leave-one-out")
```

```
           lty col
bord.modif   1   1
border       2   2
theo         3   3
```

(the smoothing parameter σ can also be controlled.)

The inhomogeneous analogue of the L-function is defined by

$$\widehat{L}_{\mathrm{inhom}}(r) = \sqrt{\frac{\widehat{K}_{\mathrm{inhom}}(r)}{2\pi r}}.$$

This can be computed using `Linhom`. For an inhomogeneous Poisson process, $L_{\mathrm{inhom}}(r) \equiv r$.

The inhomogeneous analogue of the pair correlation function can be defined, similarly to the homogeneous case, as

$$g_{\mathrm{inhom}}(r) = \frac{K'_{\mathrm{inhom}}(r)}{2\pi r}.$$

It has the same interpretation, namely, that $g_{\mathrm{inhom}}(r)$ is the probability of observing a pair of points at certain locations separated by a distance r, divided by the corresponding probability for a Poisson process of the same (inhomogeneous) intensity.

The inhomogeneous pair correlation function is currently computed by calling `Kinhom` followed by `pcf.fv` (which does numerical differentiation):

```
> g <- pcf(Kinhom(bei))
```

17.2 Inhomogeneous cluster models

The inhomogeneous Poisson process was described in Section 11.1. We can also introduce spatial inhomogeneity into any of the non-Poisson models described in Section 14.

In the case of Poisson cluster processes (Section 14.1) we can introduce inhomogeneity in either the parent process or the offspring processes.

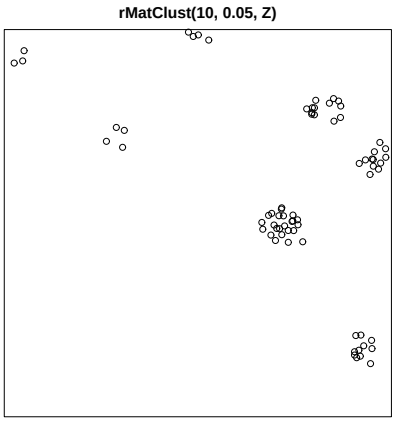
To make the *parents* inhomogeneous, we simply generate the parent points from an inhomogeneous Poisson process with some intensity function $\kappa(u)$.

To make the *clusters* inhomogeneous, we use a clever construction by Waagepetersen [45]. For a parent point at location (x_0, y_0), the offspring are generated from a Poisson process with intensity $\beta(x, y) = \mu(x, y) f(x - x_0, y - y_0)$, where $f(u, v)$ is either the Gaussian probability density (for the Thomas process) or the uniform probability density in a disc (for the Matérn cluster process), and $\mu(x, y)$ is the *reference* or *modulating* intensity. The number of offspring from a given parent (x_0, y_0) is a Poisson random variable with mean

$$B(x_0, y_0) = \int \beta(x, y)\, \mathrm{d}x\, \mathrm{d}y = \int f(x - x_0, y - y_0)\mu(x, y)\, \mathrm{d}x\, \mathrm{d}y.$$

The simulation algorithms `rMatClust` and `rThomas` allow these options. If the parent intensity parameter `kappa` is given as a `function(x,y)` or a pixel image, then the parents are Poisson with inhomogeneous intensity `kappa`. If the offspring mean parameter `mu` is given as a `function(x,y)` or a pixel image, then this determines an inhomogeneous reference density for the clusters.

```
> Z <- as.im(function(x, y) {
+     6 * exp(2 * x - 1)
+ }, owin())
> plot(rMatClust(10, 0.05, Z))
```



17.3 Fitting inhomogeneous models by minimum contrast

Minimum contrast methods can be applied to inhomogeneous point process models.

In principle we could fit any model (homogeneous or inhomogeneous) by the method of minimum contrast using any summary statistic. However, the method works best when we have an exact formula for the true value of the summary function for the model, expressed as a function of the parameters of the model.

Waagepetersen [45] pointed out that, if we take a Thomas process or Matérn cluster process (or in general a Neyman-Scott process) with **homogeneous** parent intensity κ and **inhomogeneous** cluster reference density $\mu(u)$, then the overall intensity of the process is

$$\lambda(u) = \kappa\,\mu(u)$$

and the *inhomogeneous* K-function is the same as it would be if μ were constant.

Thus, we can fit a Thomas process or Matérn cluster process with inhomogeneous clusters as follows:

1. estimate the inhomogeneous intensity $\lambda(u)$ of the process.
2. derive an estimate of the inhomogeneous K-function.
3. use the method of minimum contrast to estimate the parent intensity κ and the cluster scale parameter (Gaussian standard deviation or disc radius), exactly as we would in the homogeneous case.

Here is an application to the rainforest data.

```
> data(bei)
> fit <- ppm(bei, ~elev + grad, covariates = bei.extra)
> lam <- predict(fit, locations = bei)
> Ki <- Kinhom(bei, lam)
> thomas.estK(Ki, c(kappa = 4e-04, sigma2 = 1))

Minimum contrast fit (object of class "minconfit")
Model: Thomas process
Fitted by matching theoretical K function to Ki
Parameters fitted by minimum contrast ($par):
       kappa       sigma2
4.267423e-04 2.941906e+01
Derived parameters of Thomas process ($modelpar):
       kappa        sigma           mu
0.0004267423 5.4239342345           NA
Converged successfully after 113 iterations.
Domain of integration: [ 0 , 125 ]
Exponents: p= 2, q= 0.25
```

18 Gibbs models

One way to construct a statistical model (in any field of statistics) is to write down its probability density. Advantages of doing this are:

- the functional form of the density reflects its probabilistic properties.
- terms or factors in the density often have an interpretation as 'components' of the model.
- it is easy to introduce terms that represent the dependence of the model on covariates, etc.

This approach is useful provided the density *can* be written down, and provided the density is tractable.

Spatial point process models that are constructed by writing down their probability densities are called **'Gibbs processes'**. Good references on Gibbs point processes are [43, 18].

18.1 Probability densities

It is possible to define probability densities for spatial point processes that live inside a bounded window W.

The probability density will be a function $f(\mathbf{x})$ defined for each finite configuration $\mathbf{x} = \{x_1, \ldots, x_n\}$ of points $x_i \in W$ for any $n \geq 0$. Notice that the number of points n is not fixed, and may be zero. Apart from this peculiarity, probability densities for point processes behave much like probability densities in more familiar contexts.

That's all you need to know for applications. **If you're interested in the mathematical technicalities, read on; otherwise, skip to section 18.2.**

A point process $\mathbf{X}$ inside W is defined to have probability density f if and only if, for any nonnegative integrable function h,

$$\mathbb{E}[h(\mathbf{X})] = e^{-|W|} h(\emptyset) f(\emptyset) + e^{-|W|} \sum_{n=1}^{\infty} \frac{1}{n!} \int_W \cdots \int_W h(\{x_1, \ldots, x_n\}) f(\{x_1, \ldots, x_n\}) \, \mathrm{d}x_1 \cdots \mathrm{d}x_n \tag{31}$$

where $|W|$ denotes the area of W.

In particular, the probability that $\mathbf{X}$ contains exactly n points is

$$p_n = \mathbb{P}\{n(\mathbf{X}) = n\} = \frac{e^{-|W|}}{n!} \int_W \cdots \int_W f(\{x_1, \ldots, x_n\}) \, \mathrm{d}x_1 \cdots \mathrm{d}x_n$$

for $n \geq 1$ and $p_0 = \mathbb{P}\{n(\mathbf{X}) = 0\} = e^{-|W|} f(\emptyset)$. Given that there are exactly n points, the conditional joint density of the locations $x_1, \ldots, x_n$ is $f(\{x_1, \ldots, x_n\})/p_n$.

18.2 Poisson processes

The uniform Poisson process with intensity 1 has probability density $f(\mathbf{x}) \equiv 1$.

The uniform Poisson process in W with intensity λ has probability density

$$f(\mathbf{x}) = \alpha \, \lambda^{n(\mathbf{x})} \tag{32}$$

where $n(\mathbf{x})$ is the number of points in the configuration $\mathbf{x}$, and the constant α is

$$\alpha = e^{(1-\lambda)|W|}.$$

The inhomogeneous Poisson process in W with intensity function $\lambda(u)$ has probability density

$$f(\mathbf{x}) = \alpha \prod_{i=1}^{n} \lambda(x_i). \tag{33}$$

where the constant α is

$$\alpha = \exp\left[\int_W (1 - \lambda(u))\,\mathrm{d}u\right].$$

The densities (32) and (33) are products of terms associated with individual points x_i. This reflects the conditional independence property (PP4) of the Poisson process.

18.3 Pairwise interaction models

In order to construct spatial point processes which exhibit interpoint interaction (stochastic dependence between points), we need to introduce terms in the density that depend on more than one point. The simplest are *pairwise interaction models*, which have probability densities of the form

$$f(\mathbf{x}) = \alpha \left[\prod_{i=1}^{n(\mathbf{x})} b(x_i)\right] \left[\prod_{i<j} c(x_i, x_j)\right] \tag{34}$$

where α is a normalising constant, $b(u),\ u \in W$ is the 'first order' term, and $c(u, v),\ u, v \in W$ is the 'second order' or 'pairwise interaction' term. The pairwise interaction term introduces dependence between points. The interaction function must be symmetric, $c(u, v) = c(v, u)$. In principle we are free to choose any functions b and c, provided the resulting density is integrable (the right side of (31) should be finite when $h \equiv 1$).

18.3.1 Hard core process

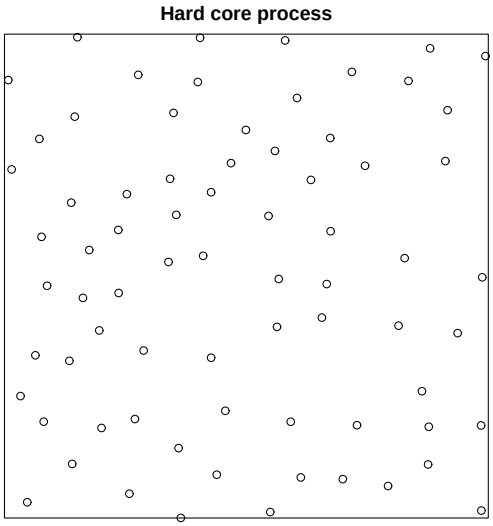
If we take $b(u) \equiv \beta$ and

$$c(u, v) = \begin{cases} 1 & \text{if } ||u - v|| > r \\ 0 & \text{if } ||u - v|| \le r \end{cases} \tag{35}$$

where $||u - v||$ denotes the distance between u and v, and $r > 0$ is a fixed distance, then the density becomes

$$f(\mathbf{x}) = \begin{cases} \alpha\beta^{n(\mathbf{x})} & \text{if } ||x_i - x_j|| > r \text{ for all } i \ne j \\ 0 & \text{otherwise} \end{cases}$$

This is the density of the Poisson process of intensity β in W conditioned on the event that no two points of the pattern lie closer than r units apart. It is known as the (classical) *hard core process*.



18.3.2 Strauss process

Generalising the hard core process, suppose we take $b(u) \equiv \beta$ and

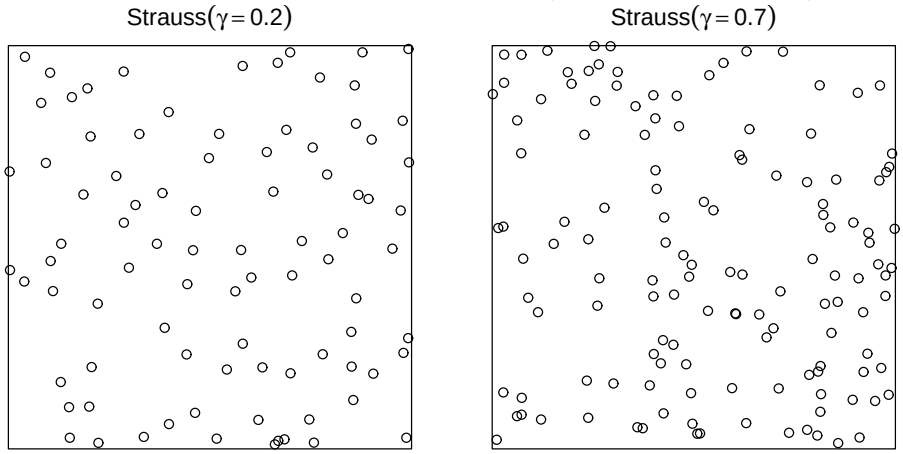
$$c(u,v) = \begin{cases} 1 & \text{if } ||u-v|| > r \\ \gamma & \text{if } ||u-v|| \leq r \end{cases} \tag{36}$$

where γ is a parameter. Then the density becomes

$$f(\mathbf{x}) = \alpha\beta^{n(\mathbf{x})}\gamma^{s(\mathbf{x})} \tag{37}$$

where $s(\mathbf{x})$ is the number of pairs of distinct points in $\mathbf{x}$ that lie closer than r units apart.

The parameter γ controls the 'strength' of interaction between points. If $\gamma = 1$ the model reduces to a Poisson process with intensity β. If $\gamma = 0$ the model is a hard core process. For values $0 < \gamma < 1$, the process exhibits inhibition (negative association) between points.



For $\gamma > 1$, the density (37) is not integrable. Hence the Strauss process is defined only for $0 \leq \gamma \leq 1$ and is a model for inhibition between points. This is typical of most Gibbs models.

18.3.3 Other pairwise interaction models

Other pairwise interactions that are considered in `spatstat` include the *Strauss-hard core* interaction (with hard core distance $h > 0$ and interaction distance $r > h$)

$$c(u,v) = \begin{cases} 0 & \text{if } ||u-v|| \le h \\ \gamma & \text{if } h < ||u-v|| \le r \\ 1 & \text{if } ||u-v|| > r \end{cases},$$

the *soft-core* interaction (with scale $\sigma > 0$ and index $0 < \kappa < 1$)

$$c(u,v) = \left(\frac{\sigma}{||u-v||}\right)^{2/\kappa},$$

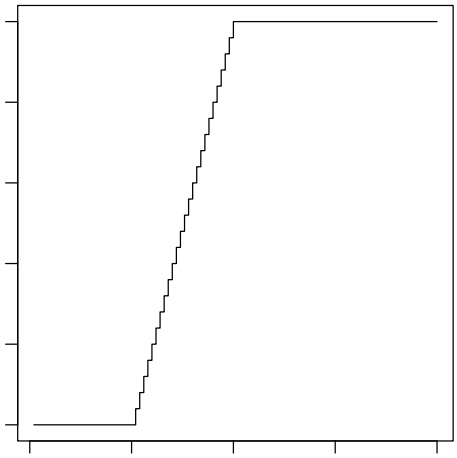
the *Diggle-Gates-Stibbard* interaction (with interaction range ρ)

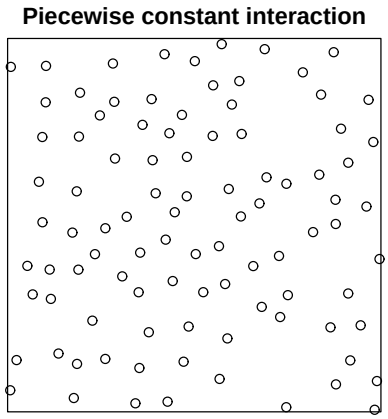
$$c(u,v) = \begin{cases} \sin\left(\frac{\pi||u-v||}{2\rho}\right)^2 & \text{if } ||u-v|| \le \rho \\ 1 & \text{if } ||u-v|| > \rho \end{cases},$$

the *Diggle-Gratton* interaction (with hard core distance δ, interaction distance ρ and index κ)

$$c(u,v) = \begin{cases} 0 & \text{if } ||u-v|| \le \delta \\ \left(\frac{||u-v||-\delta}{\rho-\delta}\right)^{\kappa} & \text{if } \delta < ||u-v|| \le \rho \\ 1 & \text{if } ||u-v|| > \rho \end{cases},$$

and the general *piecewise constant* interaction in which $c(||u-v||)$ is a step function of $||u-v||$.





18.4 Higher-order interactions

There are some useful Gibbs point process models which exhibit interactions of higher order, that is, in which the probability density has contributions from m-tuples of points, where $m > 2$.

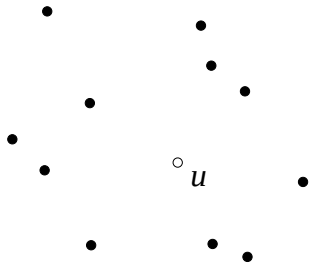
One example is the *area-interaction* or Widom-Rowlinson process [11] with probability density

$$f(\mathbf{x}) = \alpha\beta^{n(\mathbf{x})}\gamma^{-A(\mathbf{x})} \tag{38}$$

where α is the normalising constant, $\beta > 0$ is an intensity parameter, and $\gamma > 0$ is an interaction parameter. Here $A(\mathbf{x})$ denotes the area of the region obtained by drawing a disc of radius r centred at each point x_i, and taking the union of these discs. The value $\gamma = 1$ again corresponds to a Poisson process, while $\gamma < 1$ produces a regular process and $\gamma > 1$ a clustered process. This process has interactions of all orders. It can be used as a model for moderate regularity or clustering.

18.5 Conditional intensity

The main tool for analysing a Gibbs point process is its *conditional intensity* $\lambda(u, \mathbf{X})$. Intuitively this determines the conditional probability of finding a point of the process at the location u given complete information about the rest of the process. For formal definitions see [18]. Informally, the conditional probability of finding a point of the process inside an infinitesimal neighbourhood of the location u, given the complete point pattern at all other locations, is $\lambda(u, \mathbf{X})\,\mathrm{d}u$.



For point processes in a bounded window, the conditional intensity at a location u given the configuration $\mathbf{x}$ is related to the probability density f by

$$\lambda(u, \mathbf{x}) = \frac{f(\mathbf{x} \cup \{u\})}{f(\mathbf{x})} \tag{39}$$

(for $u \notin \mathbf{x}$), the ratio of the probability densities for the configuration $\mathbf{x}$ with and without the point u added.

The homogeneous Poisson process with intensity λ has conditional intensity

$$\lambda(u, \mathbf{x}) = \lambda$$

while the inhomogeneous Poisson process with intensity function $\lambda(u)$ has conditional intensity

$$\lambda(u, \mathbf{x}) = \lambda(u)$$

. The conditional intensity for a Poisson process does not depend on the configuration $\mathbf{x}$, because the points of a Poisson process are independent.

For the general pairwise interaction process (34) the conditional intensity is

$$\lambda(u, \mathbf{x}) = b(u) \prod_{i=1}^{n(\mathbf{x})} c(u, x_i). \tag{40}$$

For the hard core process,

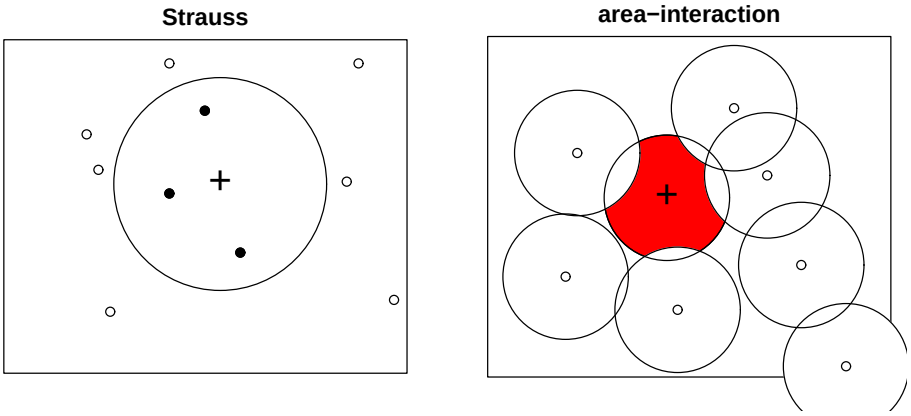
$$\lambda(u, \mathbf{x}) = \begin{cases} \beta & \text{if } ||u - x_i|| > r \text{ for all } i \\ 0 & \text{otherwise} \end{cases} \tag{41}$$

which has the nice interpretation that a point u is either 'permitted' or 'not permitted' depending on whether it satisfies the hard core requirement.

For the Strauss process

$$\lambda(u, \mathbf{x}) = \beta \gamma^{t(u, \mathbf{x})} \tag{42}$$

where $t(u, \mathbf{x}) = s(\mathbf{x} \cup \{u\}) - s(\mathbf{x})$ is the number of points of $\mathbf{x}$ that lie within a distance r of the location u. For $\gamma < 1$, this has the interpretation that a random point is less likely to occur at the location u if there are many points in the neighbourhood.



For the area-interaction process,

$$\lambda(u, \mathbf{x}) = \beta \gamma^{-B(u,\mathbf{x})} \tag{43}$$

where $B(u, \mathbf{x}) = A(\mathbf{x} \cup \{u\}) - A(\mathbf{x})$ is the area of that part of the disc of radius r centred on u that is not covered by discs of radius r centred at the other points $x_i \in \mathbf{x}$. If the points represent trees or plants, we may imagine that each tree takes nutrients and water from the soil inside a circle of radius r. Then we may interpret $B(u, \mathbf{x})$ as the area of the 'unclaimed zone' where a new plant at location u would be able to draw nutrients and water without competition from other plants. For $\gamma < 1$ we can interpret (43) as saying that a random point is less likely to occur when the unclaimed area is small.

The conditional intensity of a point process determines the probability density, through (39). Hence we can use the conditional intensity to define a point process. The conditional intensity is the preferred modelling tool for Gibbs processes: it has a direct interpretation, and it is easier to handle than the probability density.

18.6 Simulating Gibbs models

Gibbs models can be simulated by Markov chain Monte Carlo algorithms. Indeed, MCMC algorithms were invented to simulate Gibbs processes [32, 37].

In brief, these algorithms simulate a Markov chain whose states are point patterns. The chain is designed so that its equilibrium distribution is the distribution of the point process we want to simulate. If the chain were run for an infinite time, the state would converge in distribution to the desired point process. In practice the chain is run for a long finite time. Further details are beyond the scope of this workshop; consult [33, 34] for more information.

Currently `spatstat` offers the function `rmh` which simulates Gibbs processes using the Metropolis-Hastings algorithm.

```
> rmh(model, start, control)
```

- `model` determines the point process model to be simulated (see `help(rmhmodel)`).
- `start` determines the initial state of the Markov chain (see `help(rmhstart)`).
- `control` specifies control parameters for running the Markov chain, such as the number of iteration steps (see `help(rmhcontrol)`).

In the simplest uses of `rmh`, the three arguments are lists of parameter values. To generate a simulated realisation of the Strauss process with parameters $\beta = 2, \gamma = 0.7, r = 0.7$ in a square of side 10,

```
> mo <- list(cif = "strauss", par = c(beta = 2, gamma = 0.2, r = 0.7),
+     w = square(10))
> X <- rmh(model = mo, start = list(n.start = 42), control = list(nrep = 1e+06))
```

The other arguments specify a random initial state of 42 points, and that the algorithm shall be run for a million iterations.

19 Methods 8: fitting Gibbs models

19.1 Maximum pseudolikelihood

Maximum likelihood estimation is intractable for most point process models. At the very least it requires Monte Carlo simulation to evaluate the likelihood (or the score and the Fisher information).

A workable alternative, at least for investigative purposes, is to maximise the log *pseudolikelihood*

$$\log \mathrm{PL}\,(\theta; \mathbf{x}) = \sum_i \log \lambda(x_i; \mathbf{x}) - \int_W \lambda(u, \mathbf{x})\,\mathrm{d}u. \tag{44}$$

You may recognise this as being very similar to the likelihood (4) of the Poisson process. In general it is not a likelihood, but the analogue of the score equation

$$\frac{\partial}{\partial \theta} \log \mathrm{PL}\,(\theta) = 0$$

is an unbiased estimating equation. Thus the maximum pseudolikelihood estimator is asymptotically unbiased, consistent and asymptotically normal under appropriate conditions.

The main advantage of maximum pseudolikelihood is that, at least for popular Gibbs models, the conditional intensity $\lambda(u, \mathbf{x})$ is easily computable, so that the pseudolikelihood is easy to compute and to maximise. The main disadvantage is the bias and inefficiency of maximum pseudolikelihood in small samples.

More computationally-intensive estimation procedures typically use the maximum pseudolikelihood estimate as their initial guess. We are implementing such procedures in `spatstat` as well.

19.2 Fitting Gibbs models in `spatstat`

We have already met the function `ppm` for fitting Poisson point process models. In fact this function will fit a wide class of Gibbs models.

`ppm` contains an implementation of the algorithm of Baddeley and Turner [3] for maximum pseudolikelihood (which extends the Berman-Turner device for Poisson processes to a general Gibbs process). The conditional intensity of the model, $\lambda_\theta(u, \mathbf{x})$, must be loglinear in the parameters θ:

$$\log \lambda_\theta(u, \mathbf{x}) = \theta \cdot S(u, \mathbf{x}), \tag{45}$$

generalising (5), where $S(u, \mathbf{x})$ is a real-valued or vector-valued function of location u and configuration $\mathbf{x}$. Parameters θ appearing in the loglinear form (45) are called 'regular' parameters, and all other parameters are 'irregular' parameters. For example, the Strauss process conditional intensity (42) can be recast as

$$\log \lambda(u, \mathbf{x}) = \log \beta + (\log \gamma) t(u, \mathbf{x})$$

so that $\theta = (\log \beta, \log \gamma)$ are regular parameters, but the interaction distance r is an irregular parameter (technically called a 'bloody nuisance parameter').

In `spatstat` we split the conditional intensity into first-order and higher-order terms:

$$\log \lambda_\theta(u, \mathbf{x}) = \eta \cdot S(u) + \varphi \cdot V(u, \mathbf{x}). \tag{46}$$

The 'first order term' $S(u)$ describes spatial inhomogeneity and/or covariate effects. The 'higher order term' $V(u, \mathbf{x})$ describes interpoint interaction.

The model with conditional intensity (46) is fitted by calling `ppm` in the form

```
ppm(X, ~ terms, V)
```

The first argument `X` is the point pattern dataset. The second argument `~terms` is a model formula, specifying the first order term $S(u)$ in (46), in the manner described in Section 11. Thus the first order term $S(u)$ in (46) may take very general forms.

The third argument `V` is an object of the special class `"interact"` which describes the interpoint interaction term $V(u, \mathbf{x})$ in (46). It may be compared to the 'family' argument which determines the distribution of the responses in a linear model or generalised linear model. Only a limited number of canned interactions are available in `spatstat`, because they must be constructed carefully to ensure that the point process exists.

To fit the Strauss process to the cells data using `ppm`,

```
> data(cells)
> ppm(cells, ~1, Strauss(r = 0.1))

Stationary Strauss process

First order term:
    beta
294.2333

Interaction: Strauss process
interaction distance:        0.1
Fitted interaction parameter gamma:        0.0128

Relevant coefficients:
Interaction
  -4.359277
```

Here `Strauss` is a special function that creates an 'interaction' object (class `"interact"`) describing the interaction structure of the Strauss process. Notice that we had to specify the value of the irregular parameter r (more about that later).

To fit the inhomogeneous Strauss process with conditional intensity

$$\lambda(u, \mathbf{x}) = b(u)\gamma^{t(u,\mathbf{x})}$$

where, say, $b(u)$ is loglinear in the Cartesian coordinates,

$$\log b((x, y)) = \beta_0 + \beta_1 x + \beta_2 y$$

we simply type

```
> ppm(cells, ~x + y, Strauss(r = 0.1))

Nonstationary Strauss process

Trend formula: ~x + y

Fitted coefficients for trend formula:
(Intercept)           x           y
  5.7460724   0.1465176  -0.2724205
```

```
Interaction: Strauss process
interaction distance:        0.1
Fitted interaction parameter gamma:        0.0128

Relevant coefficients:
Interaction
  -4.357253
```

To fit an inhomogeneous Strauss process with log-quadratic first order term,

```
> ppm(cells, ~polynom(x, y, 2), Strauss(r = 0.1))

Nonstationary Strauss process

Trend formula: ~polynom(x, y, 2)

Fitted coefficients for trend formula:
          (Intercept)   polynom(x, y, 2)[x]   polynom(x, y, 2)[y]
             3.019133             11.064005              6.154949
polynom(x, y, 2)[x^2] polynom(x, y, 2)[x.y] polynom(x, y, 2)[y^2]
            -9.853849             -1.761367             -5.579568

Interaction: Strauss process
interaction distance:        0.1
Fitted interaction parameter gamma:        0.0071

Relevant coefficients:
Interaction
  -4.945833
```

19.3 Interpoint interactions

Instead of `Strauss` we may use any of the following functions to create an interaction:

`Poisson()`	the Poisson point process (the default)
`Strauss()`	the Strauss process
`StraussHard()`	the Strauss/hard core point process
`Softcore()`	pairwise interaction, soft core potential
`PairPiece()`	pairwise interaction, piecewise constant
`DiggleGratton()`	Diggle-Gratton potential
`LennardJones()`	Lennard-Jones potential
`Pairwise()`	pairwise interaction, user-supplied potential
`AreaInter()`	area-interaction process
`Geyer()`	Geyer's saturation process
`Saturated()`	Saturated pair model, user-supplied potential
`OrdThresh()`	Ord process, threshold potential
`Ord()`	Ord model, user-supplied potential

(There are two additional ones for multitype point processes, described in section 25.3.2.)

The area-interaction model and the Geyer saturation model are quite handy, as they can be used to model both clustering and regularity.

```
> data(redwood)
> ppm(redwood, ~1, Geyer(r = 0.07, sat = 2))


Stationary Geyer saturation process

First order term:
   beta
17.0143

Interaction: Geyer saturation process
interaction distance:        0.07
 saturation parameter:        2
Fitted interaction parameter gamma:        2.3509

Relevant coefficients:
Interaction
  0.8547814


> ppm(redwood, ~1, AreaInter(r = 0.03))


Stationary Area-interaction process

First order term:
    beta
571.5617

Interaction: Area-interaction process
disc radius:        0.03
Fitted interaction parameter eta:        19.11

Relevant coefficients:
Interaction
   2.950212
```

For more detailed explanation of modelling, see [5].

19.4 Fitted point process models

The result of the `ppm` call is an object of class `"ppm"` ('point process model'). This is very closely analogous to a fitted linear model (`lm`) or fitted generalised linear model (`glm`).

Standard R operations that are defined for fitted point process models (i.e. that have methods for the class `"ppm"`) include:

`print`	print basic information
`summary`	print detailed summary information
`plot`	plot the fitted (conditional) intensity
`predict`	fitted (conditional) intensity
`fitted`	fitted (conditional) intensity at data points
`update`	re-fit the model
`coef`	extract the fitted coefficient vector $\widehat{\theta}$
`vcov`	variance-covariance matrix of $\widehat{\theta}$
`anova`	analysis of deviance
`logLik`	evaluate log-pseudolikelihood

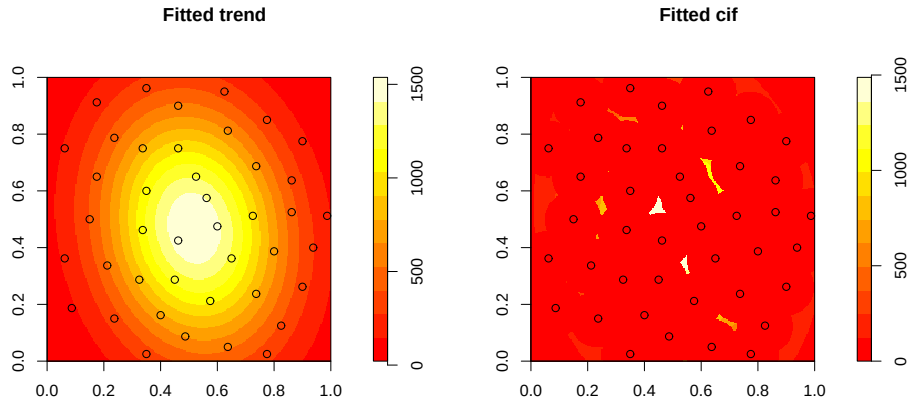
(the methods for `anova` and `vcov` are only available for Poisson models).

Plotting a fitted model generates a series of image and contour plots of

- the fitted first order term $\exp(\hat{\eta} \cdot S(u))$
- the fitted conditional intensity $\lambda_{\hat{\theta}}(u, \mathbf{x})$ evaluated for the data pattern $\mathbf{x}$

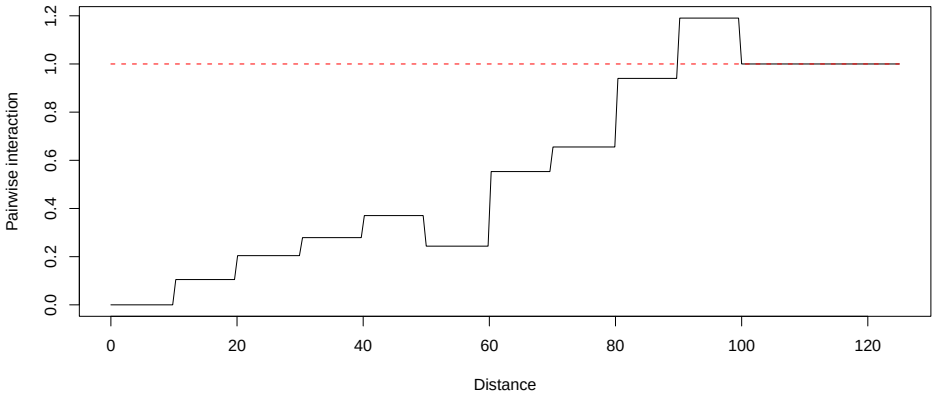
For Poisson models, the two plots are equivalent, and give the fitted intensity function.

```
> fit <- ppm(cells, ~polynom(x, y, 2), Strauss(r = 0.1))
> par(mfrow = c(1, 2))
> plot(fit, how = "image", ngrid = 256)
```



For non-Poisson models, it is also possible to extract and plot the interpoint interaction function, using `fitin`.

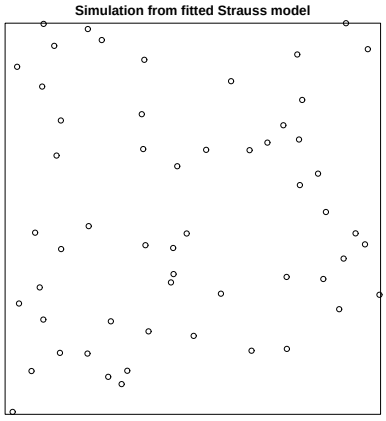
```
> model <- ppm(X, ~1, PairPiece(seq(10, 100, by = 10)))
> f <- fitin(model)
> plot(f)
```



19.5 Simulation from fitted models

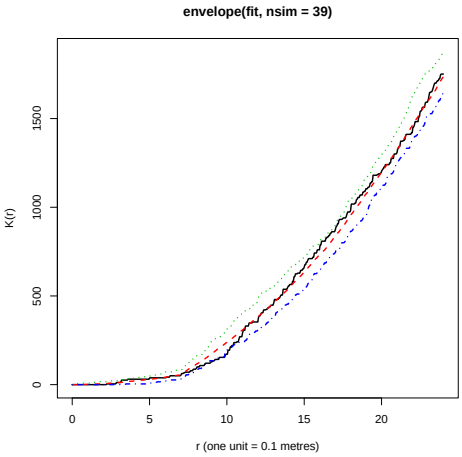
A fitted Gibbs model can also be simulated automatically using `rmh`.

```
> fit <- ppm(swedishpines, ~1, Strauss(r = 7))
> Xsim <- rmh(fit)
> plot(Xsim, main = "Simulation from fitted Strauss model")
```



The `envelope` command will also generate simulation envelopes for a fitted model.

```
> plot(envelope(fit, nsim = 39))
```



19.6 Dealing with nuisance parameters

Irregular parameters, such as the interaction radius r in the Strauss process, cannot be estimated directly using `ppm`. Indeed the statistical theory for estimating such parameters is unclear.

For some special cases, a maximum likelihood estimator of the nuisance parameter is available. For example, for the 'hard core process' (Strauss process with interaction parameter $\gamma = 0$) with interaction radius r, the maximum likelihood estimator is the minimum nearest-neighbour distance. Thus the following is a reasonable approach to the `cells` dataset:

```
> rhat <- min(nndist(cells))
> rhat <- rhat * 0.99999
> ppm(cells, ~1, Strauss(r = rhat))

Stationary Strauss process

First order term:
    beta
168.2692

Interaction: Strauss process
interaction distance:        0.0836293018068393
Fitted interaction parameter gamma:        0

Relevant coefficients:
Interaction
  -19.29955
```

The analogue of profile likelihood, *profile pseudolikelihood*, provides a general solution which may or may not perform well. If $\theta = (\phi, \eta)$ where ϕ denotes the nuisance parameters and η the regular parameters, define the profile log pseudolikelihood by

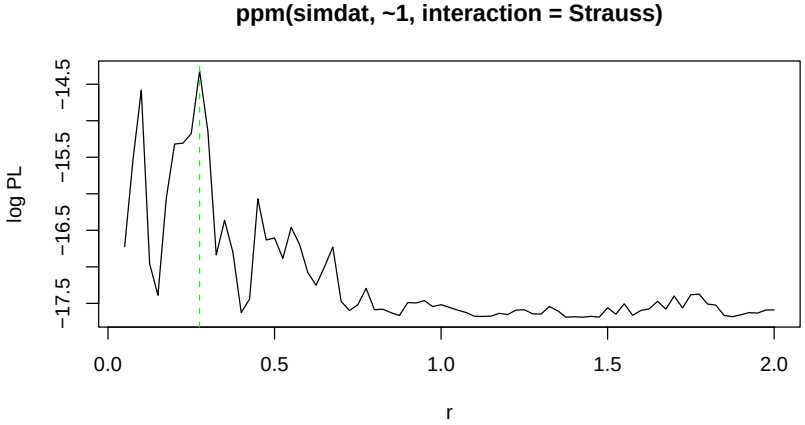
$$\mathrm{PLP}(\phi, \mathbf{x}) = \max_{\eta} \log \mathrm{PL}\left((\phi, \eta); \mathbf{x}\right).$$

The right hand side can be computed, for each fixed value of ϕ, by the algorithm `ppm`. Then we just have to maximise $\mathrm{PLP}(\phi)$ over ϕ. This is done by the command `profilepl`:

```
> data(simdat)
> df <- data.frame(r = seq(0.05, 2, by = 0.025))
> pfit <- profilepl(df, Strauss, simdat, ~1)

> pfit

Profile log pseudolikelihood values
for model:        ppm(simdat, ~1, interaction = Strauss)
fitted with rbord= 2
Interaction: Strauss
 with irregular parameter 'r' in [0.05, 2]
Optimum value of irregular parameter: r = 0.275
```

The result is an object of class `profilepl` containing the profile log pseudolikelihood function, the optimised value of the irregular parameter r, and the final fitted model. To plot the profile log pseudolikelihood,

```
> plot(pfit)
```



To extract the final fitted model,

```
> pfit$fit

Stationary Strauss process

First order term:
    beta
2.583110

Interaction: Strauss process
interaction distance:        0.275
Fitted interaction parameter gamma:        0.5631

Relevant coefficients:
Interaction
 -0.5743608
```

There is a `summary` method for these objects as well.

19.7 Improvements over maximum pseudolikelihood

Maximum pseudolikelihood is quick and dirty. There are statistically more efficient alternatives, but they are computationally intensive.

Currently we have implemented the easiest of these alternatives, the Huang-Ogata [27] one-step approximation to maximum likelihood. Starting from the maximum pseudolikelihood estimate $\hat{\theta}_{PL}$, we simulate M independent realisations of the model with parameters $\hat{\theta}_{PL}$, evaluate the canonical sufficient statistics, and use them to form estimates of the score and Fisher information at $\theta = \hat{\theta}_{PL}$. Then we take one Newton-Raphson step, updating the value of θ. The rationale is that the log-likelihood is approximately quadratic in a neighbourhood of the maximum pseudolikelihood estimator, so that one Newton-Raphson step is almost enough.

To use the Huang-Ogata method instead of maximum pseudolikelihood, add the argument `method="ho"`.

```
> fit <- ppm(simdat, ~1, Strauss(r = 0.275), method = "ho")

> fit

Stationary Strauss process

First order term:
    beta
2.500546

Interaction: Strauss process
interaction distance:        0.275
Fitted interaction parameter gamma:        0.6951

Relevant coefficients:
Interaction
 -0.3637451

> vcov(fit)

             [,1]        [,2]
[1,]  0.01070257 -0.01264063
[2,] -0.01264063  0.03635432
```

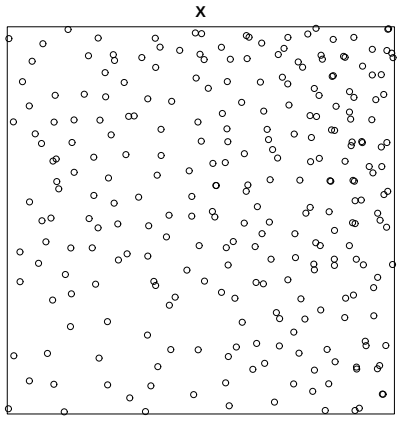
For models fitted by Huang-Ogata, the variance-covariance matrix returned by `vcov` is computed from the simulations.

20 Methods 9: validation of fitted Gibbs models

Goodness-of-fit testing and model validation for Poisson models were described in Section 12. Checking a fitted Gibbs point process model is more difficult. There is little theory available to support goodness-of-fit tests and the like.

As an example, consider the following data:

```
> data(residualspaper)
> X <- residualspaper$Fig4b
> plot(X)
```



We fit a Strauss process model with a log-quadratic intensity term:

```
> fit <- ppm(X, ~polynom(x, y, 2), Strauss(0.05), correction = "isotropic")
```

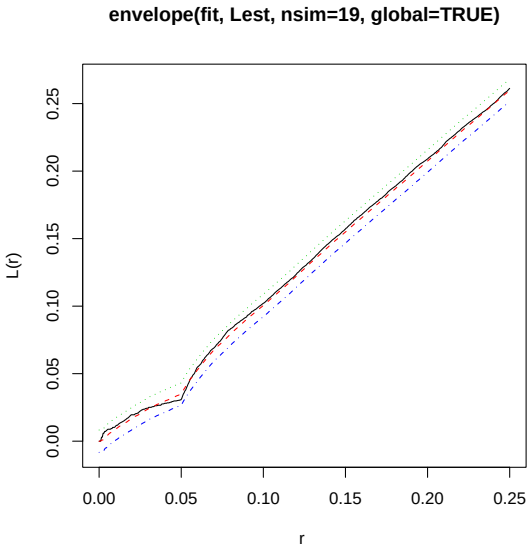
The question is how to confirm or validate this model.

20.1 Goodness-of-fit testing for Gibbs processes

For a fitted Gibbs process, no theory is available to support the χ^2 goodness-of-fit test or the Kolmogorov-Smirnov test. The predicted mean number of points in a given region is not known in closed form for a Gibbs process. Thus, the appropriate test statistic for a χ^2 test is not even available in closed form, let alone the null distribution of this statistic.

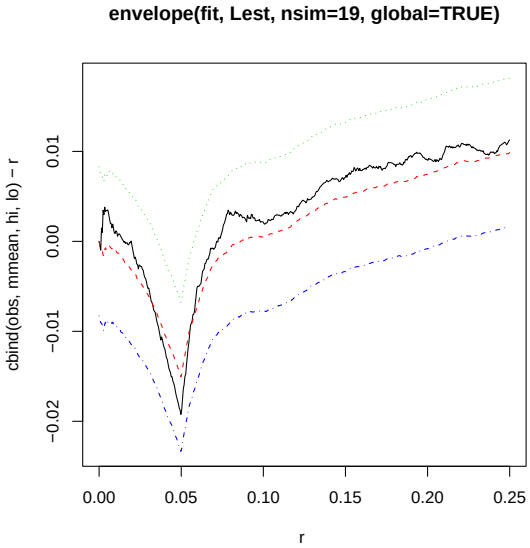
Instead, goodness-of-fit for fitted Gibbs models often relies on the summary functions K and G. The command `envelope` will accept as its first argument a fitted Gibbs model, and will simulate from this model to determine the critical envelope.

```
> plot(envelope(fit, Lest, nsim = 19, global = TRUE))
```



Let's subtract the theoretical Poisson value $L(r) = r$ to get a more readable plot:

```
> plot(envelope(fit, Lest, nsim = 19, global = TRUE), . - r ~ r)
```



This is fairly consistent with a Strauss process.

20.2 Residuals for Gibbs processes

Residuals for a general Gibbs model were defined only recently [6, 1]. The total residual in a region $B \subset \mathbb{R}^2$ is defined as

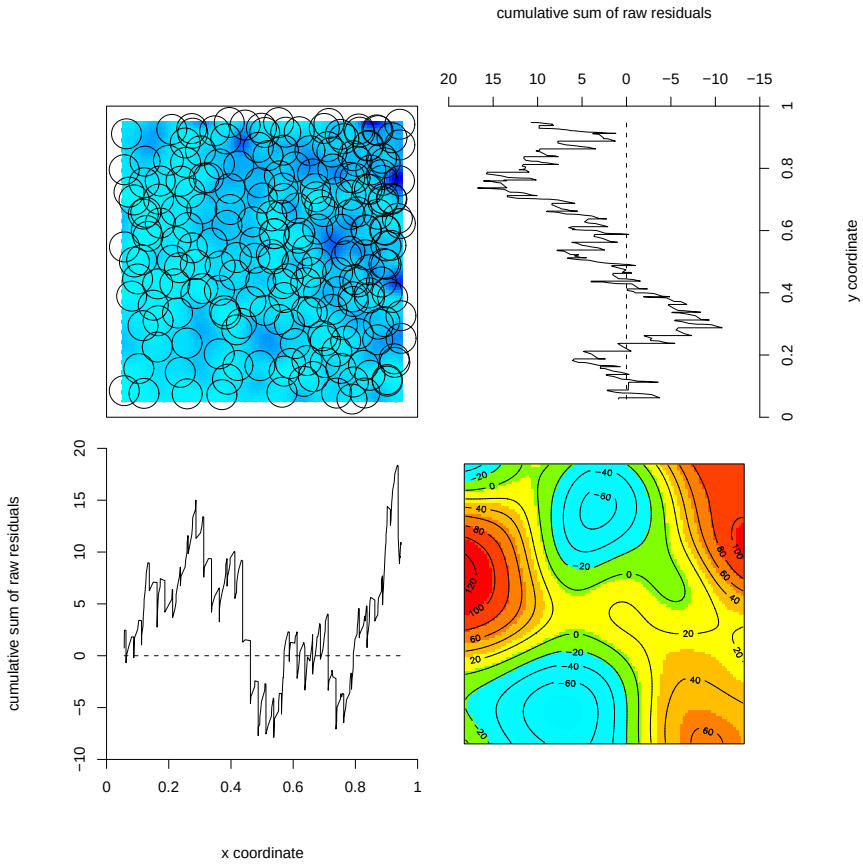
$$R(B) = n(\mathbf{x} \cap B) - \int_B \widehat{\lambda}(u, \mathbf{x})\, \mathrm{d}u \tag{47}$$

where again $n(\mathbf{x} \cap B)$ is the observed number of points in the region B, and $\widehat{\lambda}(u, \mathbf{x})$ is the **conditional** intensity of the fitted model, *evaluated for the data point pattern* $\mathbf{x}$. If the fitted model is correct, the residuals have mean zero.

This definition is similar to the definition of residuals for Poisson processes (Section 12.2) except that the intensity $\widehat{\lambda}(u)$ of the fitted Poisson process has been replaced by the *conditional* intensity $\widehat{\lambda}(u, \mathbf{x})$ of the fitted Gibbs process evaluated for the data point pattern $\mathbf{x}$.

Residuals for Gibbs processes can be plotted as explained in Section 12.2.

```
> diagnose.ppm(fit)
```

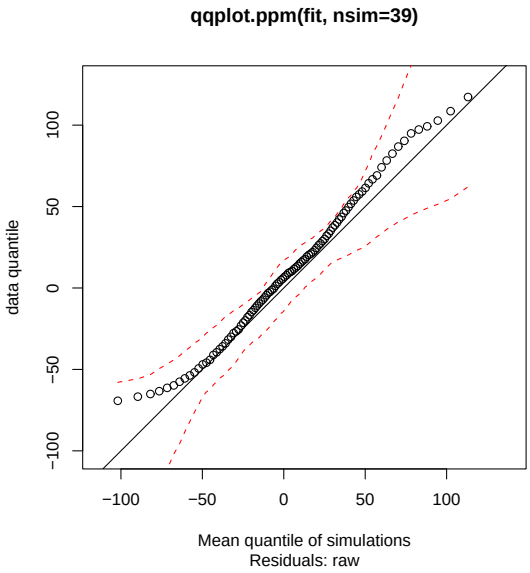


At the time of writing, `spatstat` does not yet display 2σ significance bands for the lurking variable plots when the fitted model is not Poisson. The interpretation of the lurking variable

plots is a little more difficult without the significance bands. One tends to place a little more emphasis on the smoothed residual field.

Interaction between points in a point process corresponds roughly to the distribution of the responses in loglinear regression. To validate the interaction terms in a point process model, we should plot the distribution of the residuals.

```
> qqplot.ppm(fit, nsim = 39)
```



This shows a Q–Q plot of the smoothed residuals, with pointwise 5% critical envelopes from simulations of the fitted model. This suggests that the Strauss model is reasonable.

These validation techniques generalise and unify many existing exploratory methods. For particular models of interpoint interaction, the Q–Q plot is closely related to the summary functions F, G and K. See [6].

21 Marked point patterns

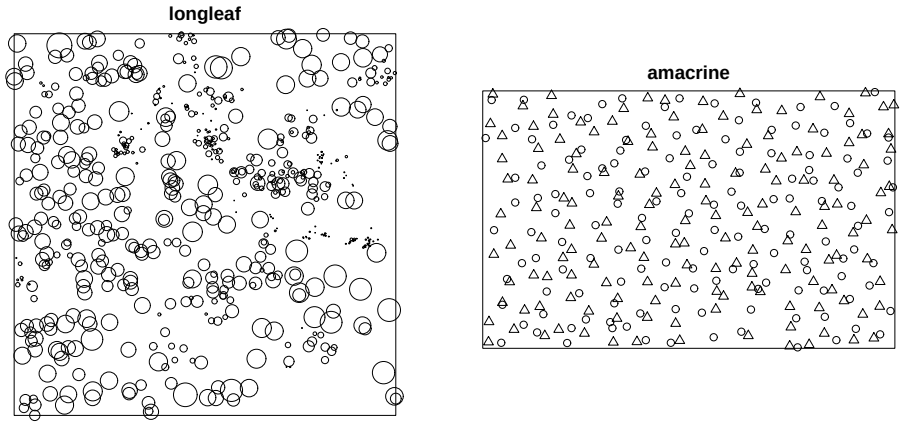
21.1 Marked point patterns

Each point in a spatial point pattern may carry additional information called a 'mark'. For example, points which are classified into two or more different types (on/off, case/control, species, colour, etc) may be regarded as marked points, with a mark which identifies which type they are. Data recording the locations and heights of trees in a forest can be regarded as a marked point pattern where the mark attached to a tree's location is the tree height.

In our current implementation, the mark attached to each point must be a *single* value (which may be numeric, character, complex, logical, or factor). Many of the functions in `spatstat` handle marked point patterns in which the mark attached to each point is either

- a **continuous variate** or "real number". An example is the Longleaf Pines dataset (`longleaf`) in which each tree is marked with its diameter at breast height. The `marks` component must be a `numeric` vector such that `marks[i]` is the mark value associated with the `i`th point. We say the point pattern has *continuous marks*.

- a **categorical variate**. An example is the Amacrine Cells dataset (`amacrine`) in which each cell is identified as either "on" or "off". Such point patterns may be regarded as consisting of points of different "types". The `marks` component must be a `factor` such that `marks[i]` is the label or type of the `i`th point. We call this a *multitype point pattern* and the levels of the factor are the possible types.



Note that, in some other packages, a point pattern dataset consisting of points of two different types (A and B say) is represented by two datasets, one representing the points of type A and another containing the points of type B. In `spatstat` we take a different approach, in which all the points are collected together in one point pattern, and the points are then labelled by the type to which they belong. An advantage of this approach is that it is easy to deal with multitype point patterns with more than 2 types. For example the classic Lansing Woods dataset represents the positions of trees of 6 different species. This is available in `spatstat` as a single dataset, a marked point pattern, with the marks having 6 levels.

21.2 Formulation

A mark variable may be interpreted as an additional coordinate for the point: for example a point process of earthquake epicentre locations (longitude, latitude), with marks giving the

occurrence time of each earthquake, can alternatively be viewed as a point process in space-time with coordinates (longitude, latitude, time).

A marked point process of points in space S with marks belonging to a set M is mathematically defined as a point process in the cartesian product $S \times M$. The space M of possible marks may be 'anything'. In current applications, typically the mark is either a categorical variable (so that the points are grouped into 'types') or a real number. Multivariate marks consisting of several such variables are also common.

A marked point pattern is an unordered set

$$\mathbf{y} = \{(x_1, m_1), \ldots, (x_n, m_n)\}, \quad x_i \in W, \quad m_i \in M$$

where x_i are the locations and m_i are the corresponding marks.

21.3 Methodological issues

21.3.1 Should the data be treated as a marked point process?

In a marked point process the points are random. Treating the data as a point process is inappropriate if the locations are fixed, or if the locations are not part of the 'response'.

Example 16 *Today's maximum temperatures at 25 Australian cities are displayed on a map.*

This is not a point process in any useful sense. The cities are fixed locations. The temperatures are observations of a spatial variable at a fixed set of locations. See the R packages `sp`, `spdep`, `spgwr` for suitable methods.

Example 17 *A mineral exploration dataset records the map coordinates where 15 core samples were drilled, and for each core sample, the assayed concentration of iron in the sample.*

This should *not* be treated as a point process. The core sample locations were chosen by a geologist, and are part of the experimental design. The main interest is in the iron concentration at these locations. This should probably be analysed as a geostatistical dataset. See the R packages `geoR`, `geoRglm` for suitable methods.

21.3.2 Joint vs. conditional analysis

There are more choices for analysis (and more traps) when marks are present. Schematically, if we write X for the points and M for the marks, then a statistical model for the marked point pattern could be formulated in several ways:

- $[X]$ $[M|X]$ — 'conditional on locations' — points X are first generated according to a spatial point process, then marks M are 'assigned' to the points by a random mechanism $[M|X]$;
- $[M]$ $[X|M]$ — 'conditional on marks' or 'split by marks' — marks M are first generated according to some random mechanism $[M]$, then they are placed at certain locations X by point process(es) $[X|M]$;
- $[X, M]$ — 'joint' — marked points are generated according to a marked point process.

These approaches typically lead to different stochastic models and have different inferential interpretations. Correspondingly, there are different null hypotheses that can be tested:

- *random labelling:* given the locations X, the marks are conditionally independent and identically distributed;
- *independence of components:* the sub-processes $\mathbf{X}_m$ of points of each mark m, are independent point processes;
- *complete spatial randomness and independence (CSRI):* the locations $\mathbf{X}$ are a uniform Poisson point process, and the marks are independent and identically distributed. (This implies both random labelling and independence of components).

These null hypotheses are not equivalent.

The properties of random labelling and independence of components are not equivalent. For example, take a point process $\mathbf{X}$ where nearest neighbour distances are always larger than a threshold r, and attach random marks to the points. The resulting marked point process cannot be generated using the independence construction, because if points with different marks are independent, they can come arbitrarily close to one another.

Example 18 (Ant nests data) *Two species of ants build nests in a desert. We want to investigate ecological interaction between the species, and between different nests of the same species. The locations of all nests are mapped, and marked by the species.*

These data can be analysed as a marked point process consisting of two different types of points. The 'mark' attached to each point is its species (a categorical variable). The most natural kind of modelling and analysis is either joint $[X, M]$ or split by species $[M]$ $[X|M]$. We could also treat one of the species as a covariate and analyse the other species conditional on it.

Example 19 *Trees in an orchard are examined and their disease status (infected/not infected) is recorded. We are interested in the spatial characteristics of the disease, such as contagion between neighbouring trees.*

These data probably should *not* be treated as a point process. The response is 'disease status'. We can think of disease status as a label applied to the trees after their locations have been determined. Since we are interested in the spatial correlation of disease status, the tree locations are effectively fixed covariate values. It would probably be best to treat these data as a discrete random field (of disease status values) observed at a finite known set of sites (the trees).

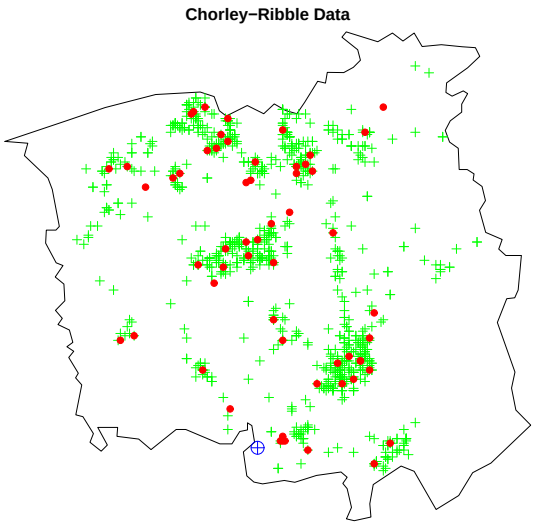
21.3.3 Grey areas

There are some 'grey areas' which permit several alternative choices of analysis. It could be appropriate either to analyse the locations and marks jointly (denoted $[X, M]$), or to analyse the marks conditional on the locations ($[M|X]$) or to analyse the locations given the marks ($[X|M]$).

One grey area occurs when the locations are random, but may be ancillary for the parameters of interest.

Example 20 *Case-control study of cancer [20, 24]. The domicile locations of all new cases of a rare cancer are mapped. To allow for spatial variation in the density of the susceptible population, domicile locations are recorded for a random sample of (matched) controls.*

This can be analysed either as a marked point pattern (where the mark is the case/control label) or, by conditioning on locations, as a random field of case/control values attached to the known domicile locations.



22 Handling marked point pattern data

This section explains how to create a marked point pattern dataset in `spatstat`, and how to manipulate it.

22.1 Creating datasets

In `spatstat` version 1, each point in a point pattern can be marked with a *single* value (i.e. one mark value per point). The marks are stored in a vector, of the same length as the number of points. The marks can be of any atomic type: numeric, integer, character, factor, logical or complex.

A marked point pattern dataset can be created using any of the following tools:

`ppp`	create point pattern dataset
`as.ppp`	convert other data to point pattern
`superimpose`	combine several point patterns into a marked point pattern
`marks`	extract marks from a point pattern
`marks<-`	attach marks to a point pattern
`%mark%`	attach marks to a point pattern
`unmark`	delete marks from a point pattern
`scanpp`	read point pattern data from text file
`clickppp`	create a pattern using point-and-click on the screen

The command `ppp` can be used to create a marked point pattern dataset from raw data. The syntax is

```
> ppp(x, y, ..., marks = m)
```

where `x`, `y` and `m` are vectors of equal length containing the (x, y) coordinates and the corresponding mark values, and `...` are arguments that determine the window for the point pattern.

> Tip: If the marks are intended to be a categorical variable (representing the types in a multitype point pattern),
>
> - ensure that `m` is stored as a `factor` in R.
> - when the point pattern `X` has been created, check that it is multitype using `is.multitype(X)`.
> - check that the factor levels are as you intended, using `levels(m)` or `levels(marks(X))` where `X` is the marked point pattern. If the factor levels are character strings, they will be sorted into alphabetical order by default.
> - be careful when performing equality/inequality comparisons involving a factor. Particular danger occurs when the factor levels are strings that represent integers.

The command `as.ppp` will convert data in another format (for example, a 2-column or 3-column matrix or data frame) to a point pattern object of class `"ppp"`. The third column of a matrix or data frame will be interpreted as containing the marks.

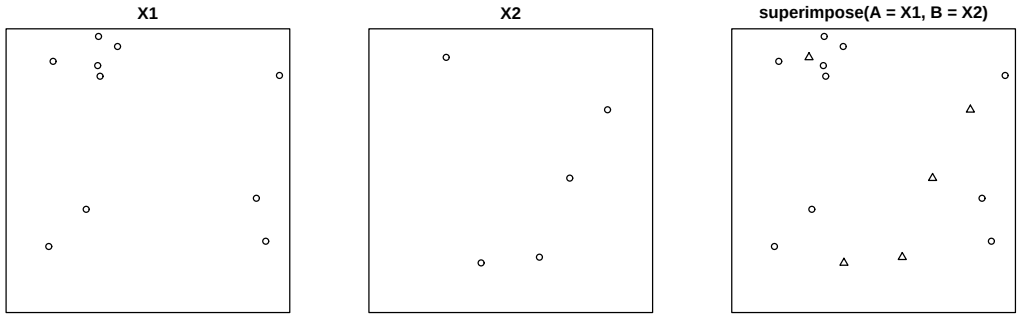
```
> mydata <- data.frame(x = runif(10), y = runif(10), m = sample(letters[1:3],
+     10, replace = TRUE))
> as.ppp(mydata, square(1))
```

```
marked planar point pattern: 10 points
multitype, with levels = a        b        c
window: rectangle = [0, 1] x [0, 1] units
```

If point pattern data are stored in a text file, the command `scanpp` will read the data and create a point pattern object of class `"ppp"`. The argument `multitype=TRUE` will ensure that the mark values are interpreted as a factor.

```
> X <- scanpp("myfile.txt", window = square(1), multitype = TRUE)
```

The command `superimpose` combines several point patterns within the same window. It can be used to create a multitype point pattern, if you have already created separate point patterns containing the points of each type. Suppose `X1` and `X2` are unmarked point patterns Then `superimpose(A=X1, B=X2)` will create a multitype point pattern by attaching the mark A to each point of `X1`, attaching the mark B to each point of `X2`, and combining the points.



Marks can be attached to an existing point pattern `X` using the function `marks<-` as in

```
> marks(X) <- m
```

or using the binary operator `%mark%`,

```
> Y <- X %mark% m
```

These are convenient when you want to assign new marks to a dataset that are computed using another variable, or perhaps to randomise the marks in a dataset.

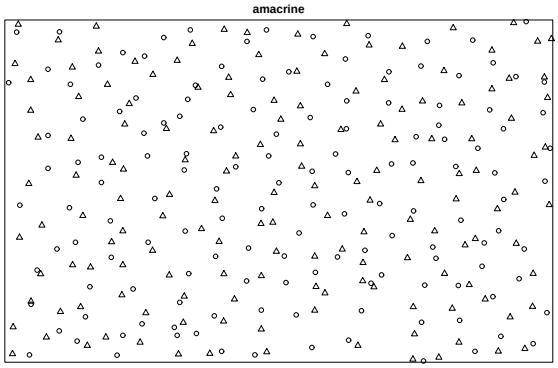
A multitype point pattern can also be created interactively using `clickppp`, using the argument `types` to specify the possible types.

22.2 Inspecting a marked point pattern

Basic tools for inspecting a marked point pattern include the `print`, `plot` and `summary` methods.

```
> data(amacrine)
> amacrine

marked planar point pattern: 294 points
multitype, with levels = off       on
window: rectangle = [0, 1.6012] x [0, 1] units (one unit = 662 microns)

> summary(amacrine)
```

```
Marked planar point pattern: 294 points
Average intensity 184 points per square unit (one unit = 662 microns)
Multitype:
    frequency proportion intensity
off       142      0.483      88.7
on        152      0.517      94.9

Window: rectangle = [0, 1.6012] x [0, 1] units
Window area =  1.60121 square units
Unit of length: 662 microns

> plot(amacrine)

off  on
  1   2
```



You can also convert a marked point pattern into a data frame for closer inspection of the coordinates and mark values:

```
> as.data.frame(amacrine)

         x      y marks
1   0.0224 0.0243    on
2   0.0243 0.1028    on
3   0.1626 0.1477    on
........
```

The marks can be extracted using the function `marks`:

```
> data(longleaf)
> m <- marks(longleaf)
```

Beware the possibility that two points with different marks may occupy the same spatial location. This is not currently detected by `ppp` since, for a marked point pattern, the function `duplicated.ppp` regards two points as identical only when their coordinates **and** mark values are identical. To detect duplication of the spatial locations, use `duplicated(unmark(X))`.

Further tools are presented in the next section.

22.3 Manipulating data

22.3.1 Manipulating marks

The following tools can manipulate the marks in a point pattern:

`marks`	extract marks
`marks<-`	attach marks to a point pattern
`%mark%`	attach marks to a point pattern
`unmark`	remove marks from point pattern

For example, the Lansing Woods data are tree locations marked by diameter at breast height (dbh) in centimetres. To convert the marks from diameters to circular areas,

```
> data(lansing)
> d <- marks(lansing)
> a <- (pi/4) * d^2
> marks(lansing) <- a
```

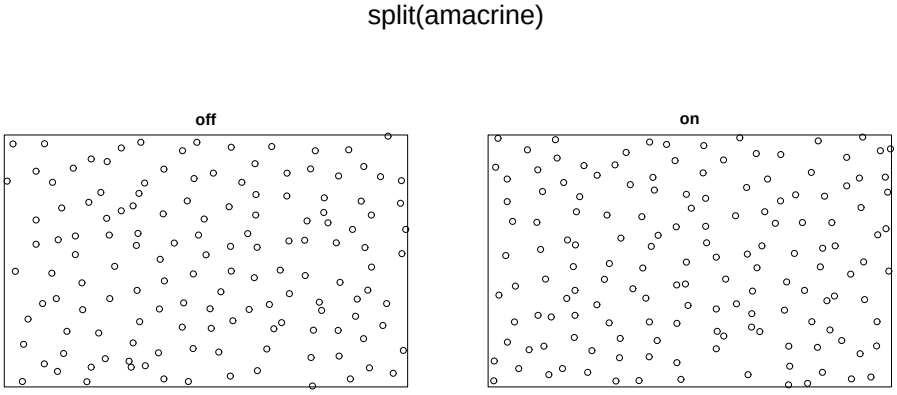
22.3.2 Separating points of different types

A *multitype* point pattern can be separated into the sub-patterns of points of each type, using the `split` command.

```
> data(amacrine)
> Y <- split(amacrine)
```

In fact `split` is a generic function and the commands above invoke the `split` method for the class of point patterns, `split.ppp`. The result `Y` is a list of point patterns, with names that correspond to the type labels. This list also belongs to the class `"splitppp"` which can be plotted automatically:

```
> plot(split(amacrine))
```

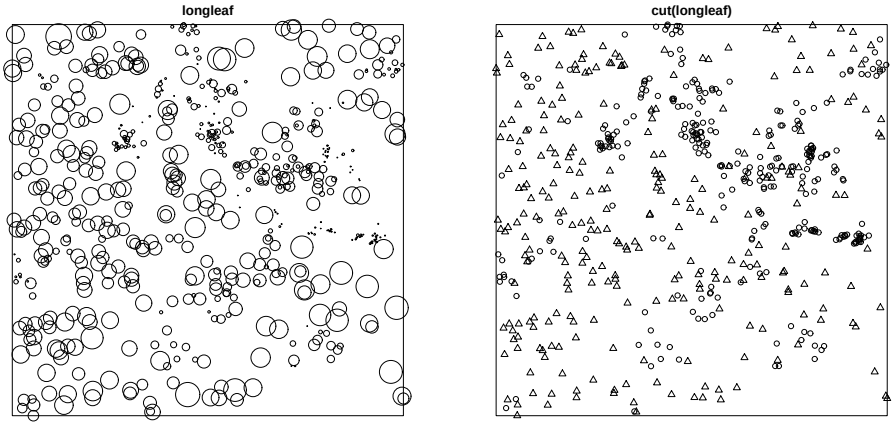


22.3.3 Cutting the numerical scale into bands

For a point pattern with *numeric* marks, the marks can be converted to a factor, using a method for the generic function `cut`. The user specifies a series of cut-points on the numerical scale; all mark values between two cut-points are given the same label.

For example, the Longleaf Pines data are the locations of trees marked with their diameter at breast height, dbh, in centimetres. By convention we define "adult" trees to be those with dbh greater than 30 centimetres. To obtain the bivariate point pattern of adult and juvenile trees,

```
> data(longleaf)
> longleaf

marked planar point pattern: 584 points
marks are numeric, of type 'double'
window: rectangle = [0, 200] x [0, 200] metres

> X <- cut(longleaf, breaks = c(0, 30, 80), labels = c("juvenile",
+     "adult"))
> X

marked planar point pattern: 584 points
multitype, with levels = juvenile        adult
window: rectangle = [0, 200] x [0, 200] metres

> par(mfrow = c(1, 2))
> plot(longleaf)

       0       20       40       60       80
0.000000 1.722522 3.445045 5.167567 6.890090

> plot(X, main = "cut(longleaf)")

juvenile    adult
       1        2

> par(mfrow = c(1, 1))
```



23 Methods 10: exploratory tools for marked point patterns

This section covers some tools for exploratory data analysis of marked point patterns. Most of the tools have been developed for the special case of *multitype* point patterns (i.e. where the marks are categorical).

23.1 Intensity

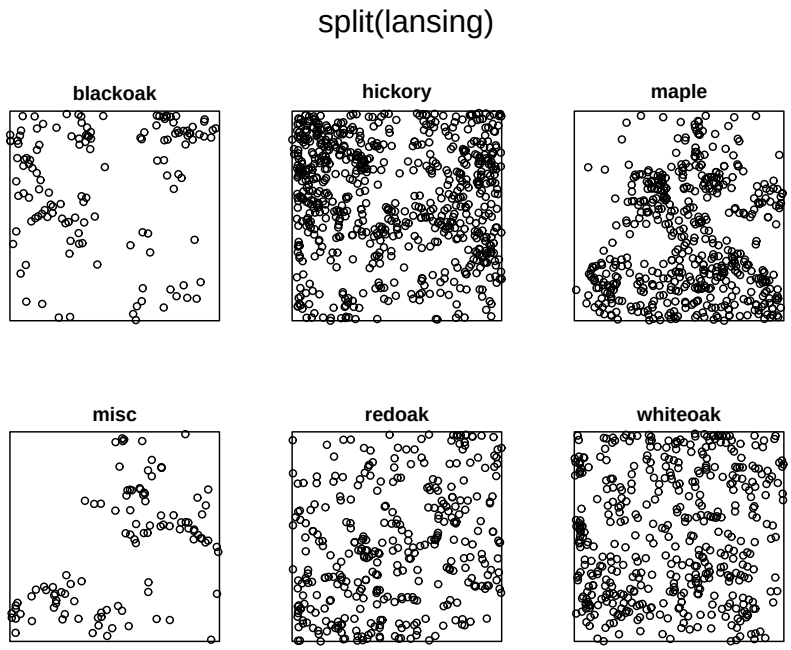
The Lansing Woods data give the locations of 6 species of trees in a forest in Michigan. Elementary estimates of the frequency distribution of species, and the intensity of each species, are available from `summary.ppp`.

```
> data(lansing)
> summary(lansing)
```

```
Marked planar point pattern: 2251 points
Average intensity 2250 points per square unit (one unit = 924 feet)

*Pattern contains duplicated points*
Multitype:
         frequency proportion intensity
blackoak       135     0.0600       135
hickory        703     0.3120       703
maple          514     0.2280       514
misc           105     0.0466       105
redoak         346     0.1540       346
whiteoak       448     0.1990       448

Window: rectangle = [0, 1] x [0, 1] units
Window area =  1 square unit
Unit of length: 924 feet
```

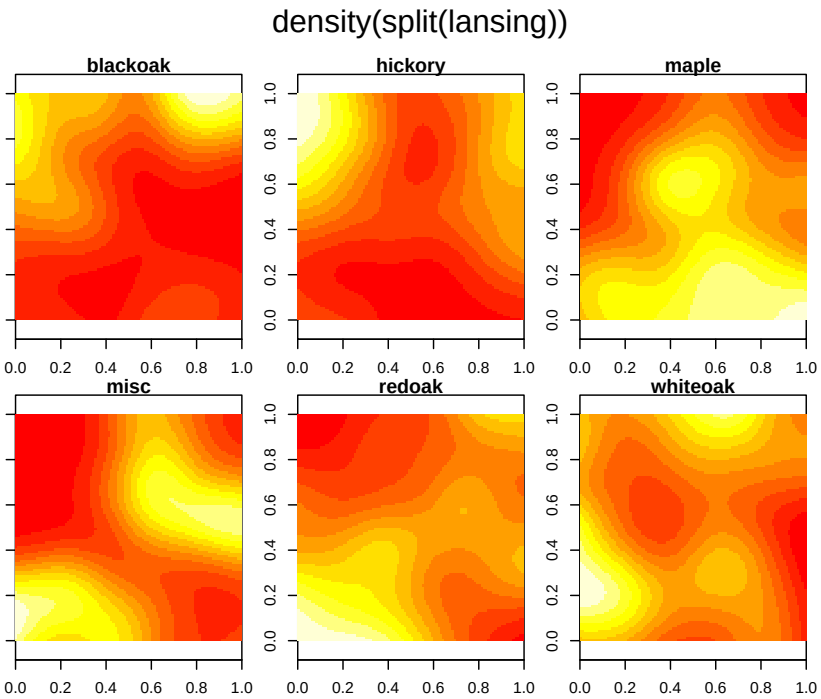
It's sensible to examine the sub-patterns of different types separately, using `split.ppp`.

```
> plot(split(lansing))
```



It would be useful to compute and plot a separate estimate of intensity for each type of tree. This is possible using the functions `density.splitppp` and `plot.listof`. They are invoked simply by typing

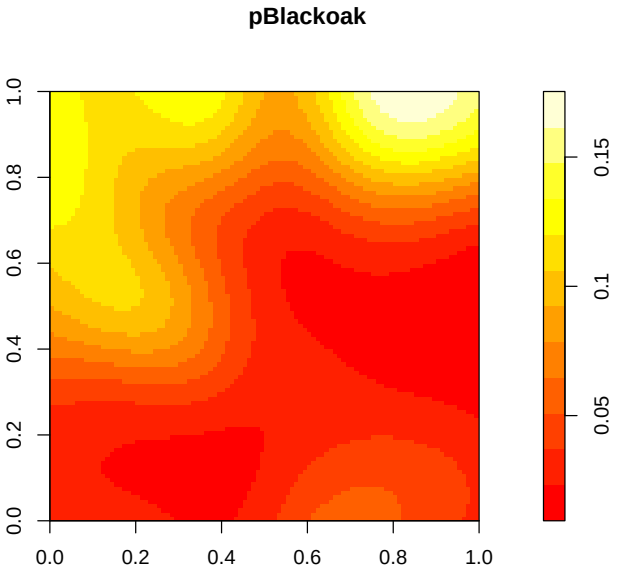
```
> plot(density(split(lansing)), ribbon = FALSE)
```



The relative proportions of intensity can then be computed using `eval.im`:

```
> Y <- density(split(lansing))
> attach(Y)
```

```
> pBlackoak <- eval.im(blackoak/(blackoak + hickory + maple + misc +
+     redoak + whiteoak))
> plot(pBlackoak)
> detach(Y)
```

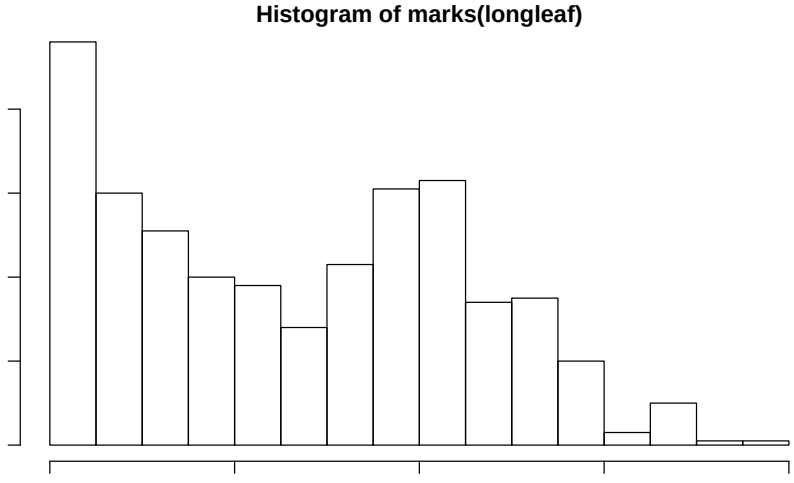


Parametric estimates of intensity can be obtained using `ppm`, fitting a Poisson model with an intensity function that may depend on location and/or on the marks. See below.

23.2 Numeric marks: distribution and trend

For a point pattern with marks that are numeric (real numbers or integers) or logical values, the mark values can be extracted using the `marks` function and inspected using the histogram or kernel density estimate:

```
> data(longleaf)
> hist(marks(longleaf))
```

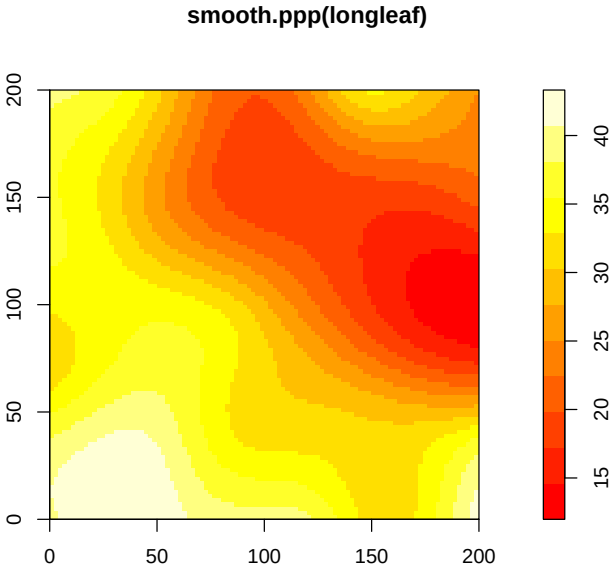


To assess spatial trend in the marks, one way is to form a kernel regression smoother. The smoothed mark value at location $u \in \mathbb{R}^2$ is

$$\widehat{m}(u) = \frac{\sum_i m_i \kappa(u - x_i)}{\sum_i \kappa(u - x_i)}$$

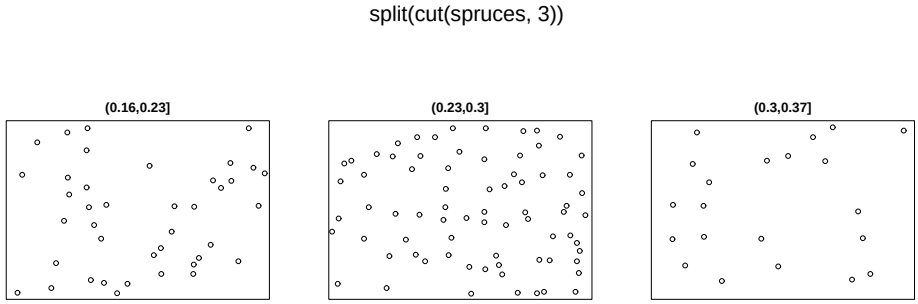
where k is the smoothing kernel, and m_i is the mark value at data point x_i. This is computed by `smooth.ppp`:

```
> plot(smooth.ppp(longleaf))
```



You can also use `cut.ppp` followed by `split.ppp` to look for spatial inhomogeneity of the marks:

```
> data(spruces)
> plot(split(cut(spruces, 3)))
```



23.3 Simple summaries of neighbouring marks

We are often interested in the marks that are attached to the close neighbours of a typical point.

For a multitype point pattern, the function `marktable` compiles a contingency table of the marks of all points within a given radius of each data point:

```
> data(amacrine)
> M <- marktable(amacrine, R = 0.1)
> M[1:10, ]
```

```
     mark
point off on
   1    1  1
   2    2  2
   3    4  3
   4    3  1
   5    4  1
   6    2  3
   7    3  2
   8    1  1
   9    3  1
   10   3  2
```

More general summaries of the marks of neighbours can be obtained using the function `markstat`. For example, to compute the average diameter of the 5 closest neighbours of each tree in the Longleaf Pines dataset,

```
> md <- markstat(longleaf, mean, N = 5)
> md[1:10]
```

```
 [1] 43.40 43.40 48.58 21.70 48.38 53.32 40.28 29.82 24.92 21.70
```

23.4 Summary functions

The summary functions F, G, J and K (and other functions derived from K, such as L and the pair correlation function) have been extended to multitype point patterns.

Assume the multitype point process $\mathbf{X}$ is stationary. Let $\mathbf{X}_j$ denote the sub-pattern of points of type j, with intensity λ_j. Then

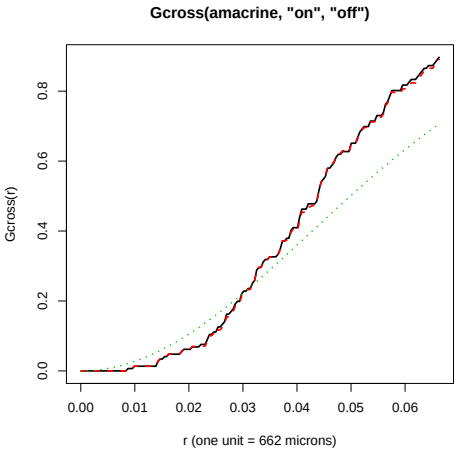
- $F_j(r)$ is the empty space function for $\mathbf{X}_j$
- $G_{ij}(r)$ is the distribution function of the distance from a point of type i to the nearest point of type j
- $K_{ij}(r)$ is $1/\lambda_j$ times the expected number of points of type j within a distance r of a typical point of type i.
- J_{ij} is defined as

$$J_{ij}(r) = \frac{1 - G_{ij}(r)}{1 - F_j(r)}.$$

The functions G_{ij}, K_{ij}, J_{ij} are called "cross-type" or "i-to-j" summary functions. They are computed in `spatstat` by `Gcross`, `Kcross` and `Jcross`.

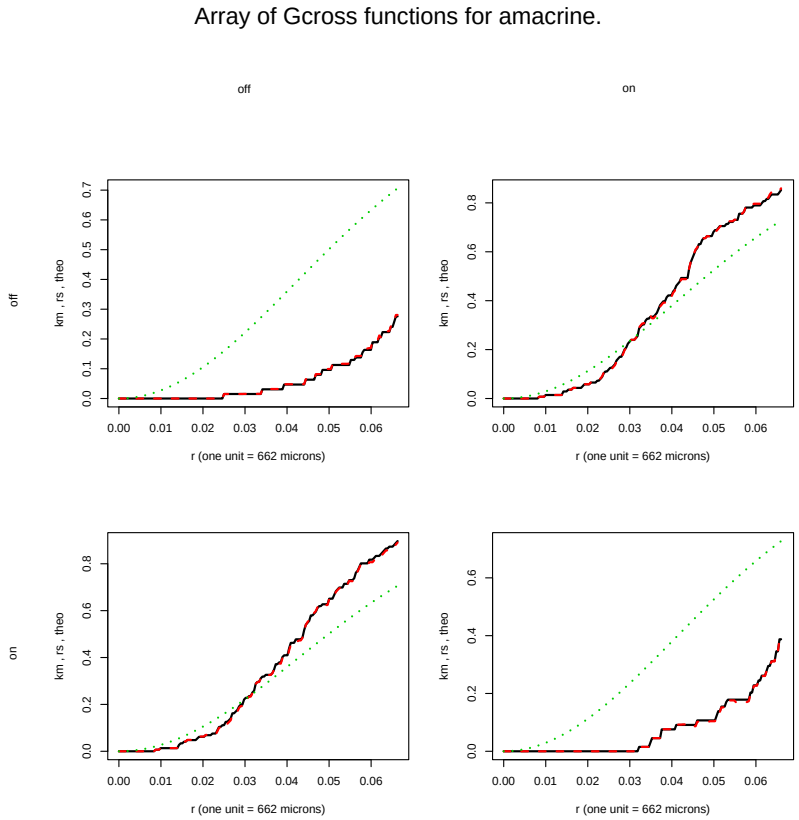
```
> data(amacrine)
> amacrine
```

```
> plot(Gcross(amacrine, "on", "off"))
```



The command `alltypes` enables the user to compute the cross-type summary functions between all pairs of types simultaneously. For example, to compute $G_{ij}(r)$ for all i and j in the amacrine cells data, we would use `alltypes(amacrine, "G")`. The result is automatically displayed as an array of plot panels.

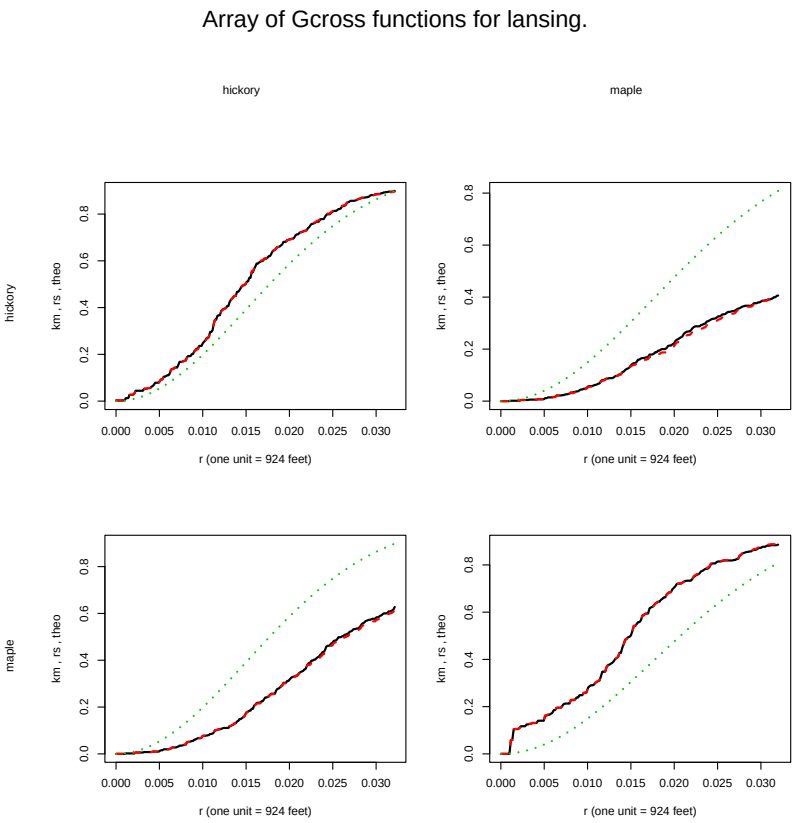
```
> plot(alltypes(amacrine, "G"))
```



The result of `alltypes` is a 'function array' (object of class `"fasp"`) which can be indexed by row and column subscripts. If the point pattern has a large number of possible types, you can compute the array of all possible pairwise G functions, then use the subscript operator to inspect a subset of the array.

```
> data(lansing)
> a <- alltypes(lansing, "G")
```

```
> plot(a[2:3, 2:3])
```



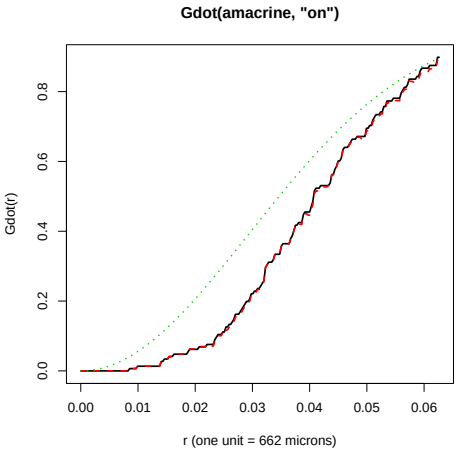
Also defined are the "i-to-any" summaries

- $G_{i\bullet}(r)$, the distribution function of the distance from a point of type i to the nearest other point of any type;
- $K_{i\bullet}(r)$ is $1/\lambda$ times the expected number of points of any type within a distance r of a typical point of type i. Here $\lambda = \sum_j \lambda_j$ is the intensity of the entire process $\mathbf{X}$.
- $J_{i\bullet}$ defined by

$$J_{i\bullet}(r) = \frac{1 - G_{i\bullet}}{1 - F(r)}$$

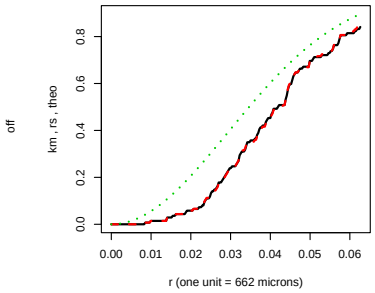
These are computing by `Gdot`, `Kdot` and `Jdot` respectively, or using `alltypes`.

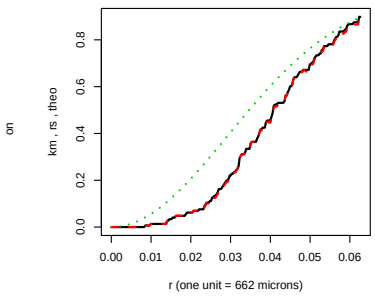
```
> plot(Gdot(amacrine, "on"))
```




```
> plot(alltypes(amacrine, "Gdot"))
```

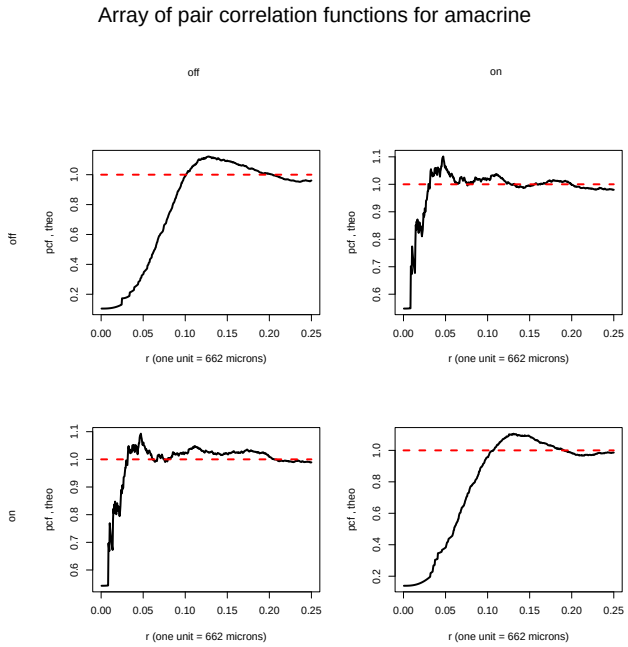
Array of Gdot functions for amacrine.





The pair correlation functions corresponding to the K-functions can also be computed, using `pcf.fasp`.

```
> K <- alltypes(amacrine, "K")
> P <- pcf(K, method = "b", spar = 1)
> plot(P, lwd = 2)
```



23.5 Mark correlation function

The mark correlation function $\rho_f(r)$ of a stationary marked point process $\mathbf{Y}$ is a measure of the dependence between the marks of two points of the process a distance r apart [42]. It is informally defined as

$$\rho_f(r) = \frac{\mathbb{E}[f(M_1, M_2)]}{\mathbb{E}[f(M, M')]}$$

where M_1, M_2 are the marks attached to two points of the process separated by a distance r, while M, M' are independent realisations of the marginal distribution of marks.

Here f is any function $f(m_1, m_2)$ with two arguments which are possible marks of the pattern, and which returns a nonnegative real value. Common choices of f are:

- for continuous real-valued marks, $f(m_1, m_2) = m_1 m_2$;
- for categorical marks (multitype point patterns), $f(m_1, m_2) = \mathbf{1}\{m_1 = m_2\}$;
- for marks taking values in $[0, 2\pi]$, $f(m_1, m_2) = \sin(m_1 - m_2)$.

Note that $\rho_f(r)$ is not a "correlation" in the usual statistical sense. It can take any nonnegative real value. The value 1 suggests "lack of correlation": under random labelling, $\rho_f(r) \equiv 1$. The interpretation of values larger or smaller than 1 depends on the choice of function f.

The mark correlation function is computed in `spatstat` by `markcorr`. It has the syntax

```
> markcorr(X, f)
```

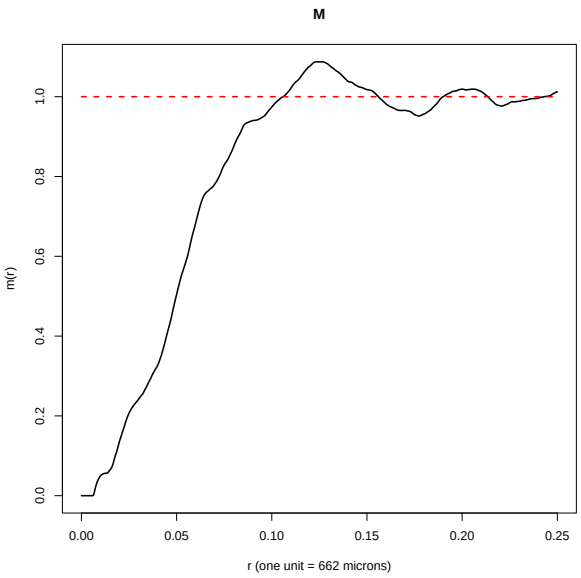
where `X` is a point pattern and `f` is an R language function. For example, for the `amacrine` data, the natural function f is $f(m_1, m_2) = \mathbf{1}\{m_1 = m_2\}$ which we encode as

```
> eqfun <- function(m1, m2) {
+     m1 == m2
+ }
```

Then simply

```
> M <- markcorr(amacrine, eqfun, correction = "translate", method = "density",
+     kernel = "epanechnikov")
```

```
> plot(M)
```



23.6 Randomisation tests

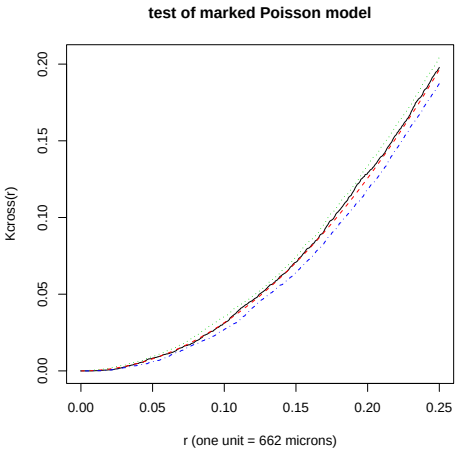
Simulation envelopes of summary functions can be used to test various null hypotheses for marked point patterns.

23.6.1 Poisson null

The null hypothesis of a homogeneous Poisson marked point process can be tested by direct simulation, using `envelope` as before. For example, using the cross-type K function as the test statistic,

```
> data(amacrine)
> E <- envelope(amacrine, Kcross, nsim = 39, i = "on", j = "off")
```

```
> plot(E, main = "test of marked Poisson model")
```



Notice that the arguments `i` and `j` here do not match any of the formal arguments of `envelope`, so they are passed to `Kcross`. This has the effect of calling `Kcross(X, i="on", j="off")` for each of the simulated point patterns `X`. Each simulated pattern is generated by the homogeneous Poisson point process with intensities estimated from the dataset `amacrine`.

23.6.2 Independence of components

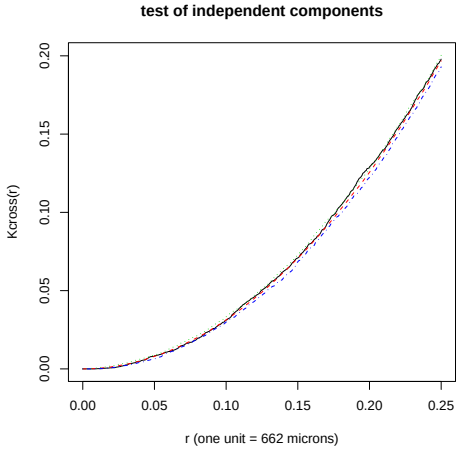
It's also possible to test other null hypotheses by a randomisation test. We discussed two popular null hypotheses:

- *random labelling:* given the locations X, the marks are conditionally independent and identically distributed;
- *independence of components:* the sub-processes $\mathbf{X}_m$ of points of each mark m, are independent point processes.

In a randomisation test of the independence-of-components hypothesis, the simulated patterns `X` are generated from the dataset by splitting the data into sub-patterns of points of one type, and randomly shifting these sub-patterns, independently of each other. The shifting is performed by `rshift`:

```
> E <- envelope(amacrine, Kcross, nsim = 39, i = "on", j = "off",
+     simulate = expression(rshift(amacrine, radius = 0.25)))
```

```
> plot(E, main = "test of independent components")
```



The independence-of-components hypothesis seems to be accepted in this example.

Under the independence hypothesis,

$$\begin{aligned} K_{ij}(r) &= \pi r^2 \\ G_{ij}(r) &= F_j(r) \\ J_{ij}(r) &\equiv 1. \end{aligned}$$

while the "i-to-any" functions have complicated values. Thus, we would normally use K_{ij} or J_{ij} to construct a test statistic for independence of components.

23.6.3 Random labelling

In a randomisation test of the random labelling null hypothesis, the simulated patterns `X` are generated from the dataset by holding the point locations fixed, and randomly resampling the marks, either with replacement (independent random sampling) or without replacement (randomly permuting the marks). The resampling operation is performed by `rlabel`.

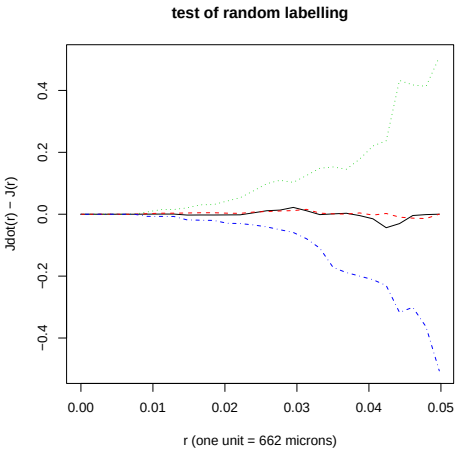
Under random labelling,

$$\begin{aligned} J_{i\bullet}(r) &= J(r) \\ K_{i\bullet}(r) &= K(r) \\ G_{i\bullet}(r) &= G(r) \end{aligned}$$

(where G, K, J are the summary functions for the point process without marks) while the other, cross-type functions have complicated values. Thus, we would normally use something like $K_{i\bullet}(r) - K(r)$ to construct a test statistic for random labelling.

To do this, cook up a little function to evaluate $J_{i\bullet}(r) - J(r)$:

```
> Jdif <- function(X, ..., i) {
+     Jidot <- Jdot(X, ..., i = i)
+     J <- Jest(X, ...)
+     dif <- eval.fv(Jidot - J)
+     return(dif)
+ }
> E <- envelope(amacrine, Jdif, nsim = 39, i = "on", simulate = expression(rlabel(amacrin

> plot(E, main = "test of random labelling")
```



The random labelling hypothesis also seems to be accepted.

24 Methods 11: multitype Poisson models

This section covers multitype Poisson process models: basic properties, simulation, and fitting models to data.

24.1 Theory

24.1.1 Complete spatial randomness and independence

A *uniform Poisson marked point process* in $\mathbb{R}^2$ with marks in $\mathcal{M}$ can be defined in the following equivalent ways.

- randomly marked Poisson process (Poisson $[X]$, iid $[M|X]$): a Poisson point process of locations $\mathbf{X}$ with intensity β is first generated. Then each point x_i is labelled with a random mark m_i, independently of other points, with distribution $\mathbb{P}\{M_i = m\} = p_m$ for $m \in \mathcal{M}$.

- superposition of independent Poisson processes (iid $[M]$, Poisson $[X|M]$): for each possible mark $m \in \mathcal{M}$, a Poisson process $\mathbf{X}_m$ is generated, with intensity β_m. The points of $\mathbf{X}_m$ are tagged with the mark m. Then the processes $\mathbf{X}_m$ with different marks $m \in \mathcal{M}$ are superimposed, to yield a marked point process.

- Poisson marked point process (jointly Poisson $[X, M]$): a Poisson process on $\mathbb{R}^2 \times \mathcal{M}$ is generated, with intensity function $\lambda(u, m) = \beta_m$ at location u and mark m.

These constructions are *equivalent* when $\beta_m = p_m\beta$. See the lovely book by Kingman [28].

Since the established term CSR ('complete spatial randomness') is used to refer to the uniform Poisson point process, I propose that the uniform *marked* Poisson point process should be called 'complete spatial randomness and independence' (CSRI).

24.1.2 Inhomogeneous Poisson marked point processes

A *inhomogeneous* Poisson marked point process $\mathbf{Y}$ with 'joint' intensity $\lambda(u, m)$ for locations u and mark values m is simply defined as an inhomogeneous Poisson point process on $\mathbb{R}^2 \times \mathcal{M}$ with intensity function $\lambda(u, m)$.

Let's restrict attention to the case of categorical marks, where $\mathcal{M}$ is finite. Then the process $\mathbf{Y}$ has the following properties:

- The locations $\mathbf{X}$, obtained by removing the marks, constitute an inhomogeneous Poisson process in $\mathbb{R}^2$ with intensity function

$$\beta(u) = \sum_m \lambda(u, m).$$

- Conditional on the locations $\mathbf{X}$, the marks attached to the points are independent. For a point x_i the conditional distribution of the mark m_i is $\mathbb{P}\{M_i = m\} = \lambda(x_i, m)/\beta(x_i)$.

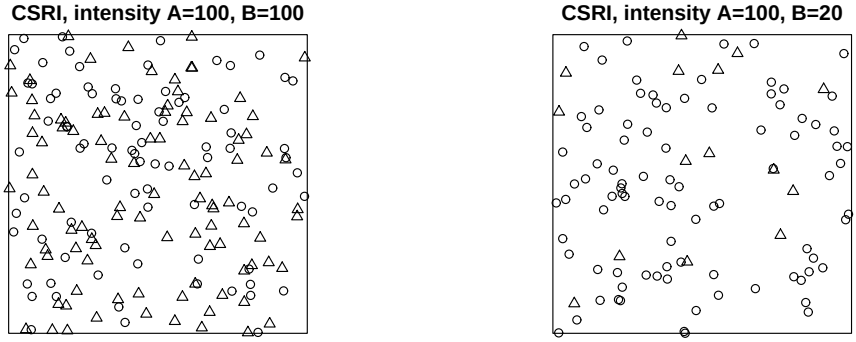
- The sub-process $\mathbf{X}_m$ of points with mark m, is an inhomogeneous Poisson point process with intensity $\beta_m(u) = \lambda(u, m)$.

- The sub-processes $\mathbf{X}_m$ of points with different marks m are independent processes.

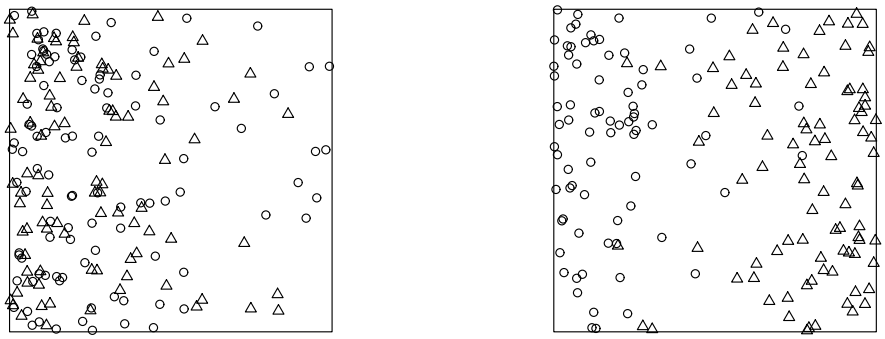
24.2 Simulation

Realisations of Poisson marked point processes can be generated using `rmpoispp`. The first argument of this command specifies the intensity or intensity function $\lambda(u, m)$. It can be a constant, a vector of constants, or an R function.

```
> par(mfrow = c(1, 2))
> Xunif <- rmpoispp(100, types = c("A", "B"), win = square(1))
> plot(Xunif, main = "CSRI, intensity A=100, B=100")
> Xunif <- rmpoispp(c(100, 20), types = c("A", "B"), win = square(1))
> plot(Xunif, main = "CSRI, intensity A=100, B=20")
> par(mfrow = c(1, 1))
```




```
> X1 <- rmpoispp(function(x, y, m) {
+     300 * exp(-3 * x)
+ }, types = c("A", "B"))
> lamb <- function(x, y, m) {
+     ifelse(m == "A", 300 * exp(-4 * x), 300 * exp(-4 * (1 - x)))
+ }
> X2 <- rmpoispp(lamb, types = c("A", "B"))
> par(mfrow = c(1, 2))
> plot(X1, main = "")
> plot(X2, main = "")
> par(mfrow = c(1, 1))
```



24.3 Fitting Poisson models

Poisson marked point process models may be fitted to point pattern data using `ppm`. Currently the methods are only available for multitype point processes (categorical marks).

24.3.1 Probability densities

Let $W \subset \mathbb{R}^2$ be the study region, and $\mathcal{M}$ the (finite) set of possible marks. Then a marked point pattern is a set

$$\mathbf{y} = \{(x_1, m_1), \ldots, (x_n, m_n)\}, \quad x_i \in W, \quad m_i \in \mathcal{M}, \quad n \geq 0$$

of pairs (x_i, m_i) of locations x_i with marks m_i. It can be viewed as a point pattern in the Cartesian product $W \times \mathcal{M}$.

The probability density of a marked point process is a function $f(\mathbf{y})$ defined for all marked point patterns $\mathbf{y}$ including the empty pattern $\emptyset$.

The process with probability density $f(\mathbf{y}) \equiv 1$ is the uniform Poisson marked point process with intensity 1 **for each mark**. That is, for this model, the sub-process of points with mark $m_i = m$ is a uniform Poisson process with intensity 1. If the marks are removed, we obtain a Poisson point process with intensity equal to $|\mathcal{M}|$, the number of possible types.

The uniform Poisson marked point process with intensity $\lambda(u, m) = \beta_m$ has probability density

$$\begin{aligned} f(\mathbf{y}) &= \exp\left(\sum_{m \in \mathcal{M}} (1 - \beta_m)|W|\right) \prod_{i=1}^{n(\mathbf{y})} \beta_{m_i} \\ &= \exp\left(\sum_{m \in \mathcal{M}} (1 - \beta_m)|W|\right) \prod_{m \in \mathcal{M}} \beta_m^{n_m(\mathbf{y})} \end{aligned}$$

where $n_m(\mathbf{y})$ is the number of points in $\mathbf{y}$ having mark value m.

The inhomogeneous Poisson marked point process with intensity function $\lambda(u, m)$, at location $u \in W$ and mark $m \in \mathcal{M}$, has probability density

$$f(\mathbf{y}) = \exp\left(\sum_{m\in\mathcal{M}} \int_W (1 - \lambda(u,m)\,\mathrm{d}u\right) \prod_{i=1}^{n(\mathbf{y})} \lambda(x_i, m_i). \tag{48}$$

24.3.2 Maximum likelihood

For the multitype Poisson process with intensity function $\lambda(u,m)$ at location $u \in W$ and mark $m \in \mathcal{M}$, the loglikelihood is, up to a constant,

$$\log L = \sum_{i=1}^{n} \log \lambda(x_i, m_i) - \sum_{m\in\mathcal{M}} \int_W \lambda(u,m)\,\mathrm{d}u. \tag{49}$$

where m_i is the mark attached to data point x_i. This is formally equivalent to the loglikelihood of a Poisson loglinear regression, so the Berman-Turner algorithm can again be used to maximise the loglikelihood.

24.3.3 Model-fitting in `spatstat`

Poisson marked point process models are fitted to data using `ppm`.

The trend formula in the call to `ppm` may involve the reserved name `marks` as a variable. This refers to the marks of the points. Since the marks are categorical, `marks` is treated as a `factor` variable for modelling purposes.

To fit the homogeneous multitype Poisson process (CSRI), equation (50), we call

```
> ppm(X, ~marks)
```

The formula `~marks` indicates that the trend depends only on the marks, and not on spatial location; since `marks` is a factor, the trend has a separate constant value for each level of `marks`. This is the model (50).

Note that if we had typed

```
> ppm(X, ~1)
```

this would have fitted the special case of CSRI where the intensities β_m are equal, $\beta_m \equiv \alpha$ say, for all possible marks. That model is only appropriate if we believe that all mark values are equally likely.

For the Lansing Woods data, the minimal model that makes sense is (50), so we call

```
> ppm(lansing, ~marks)

Stationary multitype Poisson process
Possible marks:
blackoak hickory maple misc redoak whiteoak

Trend formula: ~marks

Intensities:
beta_blackoak  beta_hickory    beta_maple     beta_misc   beta_redoak
          135           703           514           105           346
beta_whiteoak
          448
```

Since `lansing` is a multitype point pattern (its marks are categorical), the variable `marks` in the formula is a factor. The model has one parameter/coefficient for each level of the factor, i.e. one coefficient for each type of point. In other words, this is the homogeneous Poisson marked point process with intensity β_m for points of mark m.

You'll notice that the parameter estimates $\widehat{\beta}_m$ coincide with those obtained from `summary.ppp` above. That is a consequence of the fact that the maximum likelihood estimates (obtained by `ppm`) are also the method-of-moments estimates (obtained by `summary.ppp`).

A more complicated example is

```
> ppm(lansing, ~marks + x)

Nonstationary multitype Poisson process
Possible marks:
blackoak hickory maple misc redoak whiteoak

Trend formula: ~marks + x

Fitted coefficients for trend formula:
  (Intercept)  markshickory    marksmaple     marksmisc   marksredoak
   4.94294727    1.65008211    1.33694849   -0.25131442    0.94116400
markswhiteoak             x
   1.19951845   -0.07581624
```

This is the marked Poisson process whose intensity function $\lambda((x, y, m))$ at location (x, y) and mark m satisfies

$$\log \lambda((x, y, m)) = \alpha_m + \beta x$$

where $\alpha_1, \ldots, \alpha_6$ and β are parameters. The intensity is loglinear in x, with a different intercept for each mark, but the same slope ("parallel loglinear regression"). In the printout above, the fitted slope parameter β is $\hat{\beta}$ =-0.07581624. As discussed in Section 11.3 on page 61, the fitted coefficients α_m for the categorical mark are interpreted in the light of the 'contrasts' in force. The default is the treatment contrasts, and the first level of the mark is `blackoak`, so in this case the fitted coefficient for `m=blackoak` is 4.942947, while the fitted coefficient for `m=hickory` is $4.942947 + 1.650082 = 6.593029$ and so on.

```
> ppm(lansing, ~marks * x)

Nonstationary multitype Poisson process
Possible marks:
blackoak hickory maple misc redoak whiteoak

Trend formula: ~marks * x

Fitted coefficients for trend formula:
    (Intercept)    markshickory      marksmaple       marksmisc     marksredoak
      5.2378062       1.4424915       0.6795604      -0.8482907       0.6916392
  markswhiteoak               x  markshickory:x    marksmaple:x     marksmisc:x
      1.0901772      -0.7063987       0.4511157       1.3243326       1.2138278
  marksredoak:x markswhiteoak:x
      0.5380413       0.2421379
```

The symbol `*` here is an 'interaction' in the usual sense for linear models. The fitted model is the marked Poisson process with

$$\log \lambda((x, y, m)) = \alpha_m + \beta_m x$$

where $\alpha_1, \ldots, \alpha_6$ and $\beta_1, \ldots, \beta_6$ are parameters. The intensity is loglinear in x with a different slope and intercept for each mark.

The result of `ppm` is again an object of class `"ppm"` representing a fitted point process model. To plot the fitted intensity and conditional intensity of the fitted model, use `plot.ppm`. For a multitype point process you will get a separate plot for each possible mark value.

More complicated examples are:

```
> ppm(lansing, ~marks * polynom(x, y, 2))
> ppm(lansing, ~marks * harmonic(x, y, 2))
```

25 Methods 12: Gibbs models for multitype point patterns

Gibbs point process models (section 18) are also available for marked point processes, and can be fitted to data using `ppm`. Currently the methods are only implemented for *multitype* point processes (categorical marks), so we restrict attention to this case.

25.1 Gibbs models

Much of the theory of Gibbs models described in Section 18 carries over immediately to *multitype* point processes.

25.1.1 Conditional intensity

The conditional intensity $\lambda(u, \mathbf{X})$ of an (unmarked) point process $\mathbf{X}$ at a location u was defined in section 18.5. Roughly speaking $\lambda(u, \mathbf{x})\,\mathrm{d}u$ is the conditional probability of finding a point near u, given that the rest of the point process $\mathbf{X}$ coincides with $\mathbf{x}$.

For a marked point process $\mathbf{Y}$ the conditional intensity is a function $\lambda((u, m), \mathbf{Y})$ giving a value at a location u for each possible mark m. For a *finite* set of marks M, we can interpret $\lambda((u, m), \mathbf{y})\,\mathrm{d}u$ as the conditional probability finding a point *with mark* m near u, given the rest of the marked point process.

The conditional intensity is related to the probability density $f(\mathbf{y})$ by

$$\lambda((u, m), \mathbf{y}) = \frac{f(\mathbf{y} \cup \{u\})}{f(\mathbf{y})}$$

for $(u, m) \notin \mathbf{y}$.

For Poisson processes, the conditional intensity $\lambda((u, m), \mathbf{y})$ coincides with the intensity function $\lambda(u, m)$ and does not depend on the configuration $\mathbf{y}$. For example, the homogeneous Poisson multitype point process or "CSRI" (Section 24.1.1) has conditional intensity

$$\lambda((u, m), \mathbf{y}) = \beta_m \tag{50}$$

where $\beta_m \geq 0$ are constants which can be interpreted in several equivalent ways (section 18.5). The sub-process consisting of points of type m *only* is Poisson with intensity β_m. The process obtained by ignoring the types, and combining all the points, is Poisson with intensity $\beta = \sum_m \beta_m$. The marks attached to the points are i.i.d. with distribution $p_m = \beta_m/\beta$.

25.1.2 Pairwise interactions

A *multitype* pairwise interaction process is a Gibbs process with probability density of the form

$$f(\mathbf{y}) = \alpha \left[\prod_{i=1}^{n(\mathbf{y})} b_{m_i}(x_i) \right] \left[\prod_{i<j} c_{m_i, m_j}(x_i, x_j) \right] \tag{51}$$

where $b_m(u), m \in \mathcal{M}$ are functions determining the 'first order trend' for points of each type, and $c_{m,m'}(u, v), m, m' \in \mathcal{M}$ are functions determining the interaction between a pair of points of given types m and m'. The interaction functions must be symmetric, $c_{m,m'}(u, v) = c_{m,m'}(v, u)$ and $c_{m,m'} \equiv c_{m',m}$. The conditional intensity is

$$\lambda((u, m); \mathbf{y}) = b_m(u) \prod_{i=1}^{n(\mathbf{y})} c_{m,m_i}(u, x_i). \tag{52}$$

25.1.3 Pairwise interactions not depending on marks

The simplest examples of multitype pairwise interaction processes are those in which the interaction term $c_{m,m'}(u,v)$ does not depend on the marks m, m'. For example, we can take any of the interaction functions $c(u,v)$ described in section 18.3 and use it to construct a marked point process.

Such processes can be constructed equivalently as follows [8]:

- an *unmarked* Gibbs process is generated with first order term $b(u) = \sum_{m\in\mathcal{M}} b_m(u)$ and pairwise interaction $c(u,v)$.
- each point x_i of this unmarked process is labelled with a mark m_i with probability distribution $\mathbb{P}\{m_i = m\} = b_i(x_i)/b(x_i)$ independent of other points.

If additionally the intensity functions are constant, $b_m(u) \equiv \beta_m$, then such a point process has the random labelling property.

25.1.4 Mark-dependent pairwise interactions

Various complex kinds of behaviour can be created by postulating a pairwise interaction that does depend on the marks.

A simple example is the *multitype hard core process* in which $\beta_m(u) \equiv \beta$ and

$$c_{m,m'}(u,v) = \begin{cases} 1 & \text{if } ||u-v|| > r_{m,m'} \\ 0 & \text{if } ||u-v|| \le r_{m,m'} \end{cases} \tag{53}$$

where $r_{m,m'} = r_{m',m} > 0$ is the hard core distance for type m with type m'. In this process, two points of type m and m' respectively can never come closer than the distance $r_{m,m'}$.

By setting $r_{m,m'} = \infty$ for a particular pair of marks m, m' we effectively remove the interaction term between points of these types. If there are only two types, say $\mathcal{M} = \{1,2\}$, then setting $r_{1,2} = \infty$ implies that the sub-processes $\mathbf{X}_1$ and $\mathbf{X}_2$, consisting of points of types 1 and 2 respectively, are independent point processes. In other words the process satisfies the independence-of-components property.

The *multitype Strauss process* has pairwise interaction term

$$c_{m,m'}(u,v) = \begin{cases} 1 & \text{if } ||u-v|| > r_{m,m'} \\ \gamma_{m,m'} & \text{if } ||u-v|| \le r_{m,m'} \end{cases} \tag{54}$$

where $r_{m,m'} > 0$ are interaction radii as above, and $\gamma_{m,m'} \ge 0$ are interaction parameters.

In contrast to the unmarked Strauss process, which is well-defined only when its interaction parameter γ is between 0 and 1, the multitype Strauss process allows some of the interaction parameters $\gamma_{m,m'}$ to exceed 1 for $m \ne m'$, provided one of the relevant types has a hard core ($\gamma_{m,m} = 0$ or $\gamma_{m',m'} = 0$).

If there are only two types, say $\mathcal{M} = \{1,2\}$, then setting $\gamma_{1,2} = 1$ implies that the sub-processes $\mathbf{X}_1$ and $\mathbf{X}_2$, consisting of points of types 1 and 2 respectively, are independent Strauss processes.

The *multitype Strauss-hard core process* has pairwise interaction term

$$c_{m,m'}(u,v) = \begin{cases} 0 & \text{if } ||u-v|| < h_{m,m'} \\ \gamma_{m,m'} & \text{if } h_{m,m'} \le ||u-v|| \le r_{m,m'} \\ 1 & \text{if } ||u-v|| > r_{m,m'} \end{cases} \tag{55}$$

where $r_{m,m'} > 0$ are interaction distances and $\gamma_{m,m'} \ge 0$ are interaction parameters as above, and $h_{m,m'}$ are hard core distances satisfying $h_{m,m'} = h_{m',m}$ and $0 < h_{m,m'} < r_{m,m'}$.

25.2 Pseudolikelihood for multitype Gibbs processes

Models can be fitted by maximum pseudolikelihood. For a multitype Gibbs point process with conditional intensity $\lambda((u,m);\mathbf{y})$, the log pseudolikelihood is

$$\log \mathsf{PL} = \sum_{i=1}^{n(\mathbf{y})} \log \lambda((x_i,m_i);\mathbf{y}) - \sum_{m\in\mathcal{M}} \int_W \lambda((u,m);\mathbf{y})\,\mathrm{d}u. \tag{56}$$

The pseudolikelihood can be maximised using an extension of the Berman-Turner device [3].

25.3 Fitting Gibbs models to multitype data

Marked point process models may be fitted to point pattern data using `ppm`. Currently the methods are only available for multitype point processes (categorical marks).

25.3.1 Interactions not depending on marks

The model-fitting function `ppm` expects an argument `interaction` that specifies the interpoint interaction structure of the point process. The default is 'no interaction', corresponding to a Poisson process.

On page 118 there is a list of interpoint interactions for modelling *unmarked* point patterns. These interactions can also be used, without modification, to fit models to *multitype* point patterns.

For example

```
> ppm(lansing, ~marks, Strauss(0.07))
```

fits a multitype version of the Strauss process (section 18.3.2) in which the conditional intensity is

$$\lambda((u,m),\mathbf{y}) = \beta_m \gamma^{t(u,\mathbf{y})}.$$

Here β_m are constants which account for the unequal abundance of the different species of tree. The other quantities are the same as in (42). The interaction between two trees is assumed to be the same for all species, and is controlled by the interaction parameter γ and interaction radius $r = 0.07$. For example, this includes the case $\gamma = 0$ where no two trees (whatever species they belong to) come closer than 0.07 units apart, a 'multitype hard core process'.

25.3.2 Interactions depending on marks

There are two additional interpoint interactions defined in `spatstat` for multitype point patterns:

`MultiStrauss`	the multitype Strauss process
`MultiStraussHard`	multitype hybrid hard core / Strauss process

In these models, the interaction between two points depends on the types of the points as well as their separation. For example, in the multitype Strauss process, for each pair of types i and j there is an interaction radius r_{ij} and interaction parameter γ_{ij}.

To fit the stationary multitype Strauss process to the dataset `betacells`:

```
> data(betacells)
> r <- 30 * matrix(c(1, 2, 2, 1), nrow = 2, ncol = 2)
> ppm(betacells, ~1, MultiStrauss(c("off", "on"), r), rbord = 60)
```

```
Stationary Multitype Strauss process
Possible marks:
off on

First order terms:
    beta_off      beta_on
0.0001373652 0.0001373652

Interaction: Pairwise interaction family
Interaction:         Multitype Strauss process
2 types of points
Possible types:
[1] "off" "on"
Interaction radii:
    off on
off  30 60
on   60 30
Fitted interaction parameters gamma_ij:
       off     on
off 0.0000 0.8303
on  0.8303 0.0000

Relevant coefficients:
markoffxoff  markoffxon   markonxon
-17.2378706  -0.1860184 -17.2138383
```

To fit a nonstationary multitype Strauss process with log-cubic polynomial trend:

```
> ppm(betacells, ~polynom(x, y, 3), MultiStrauss(c("off", "on"),
+     r), rbord = 60)
```

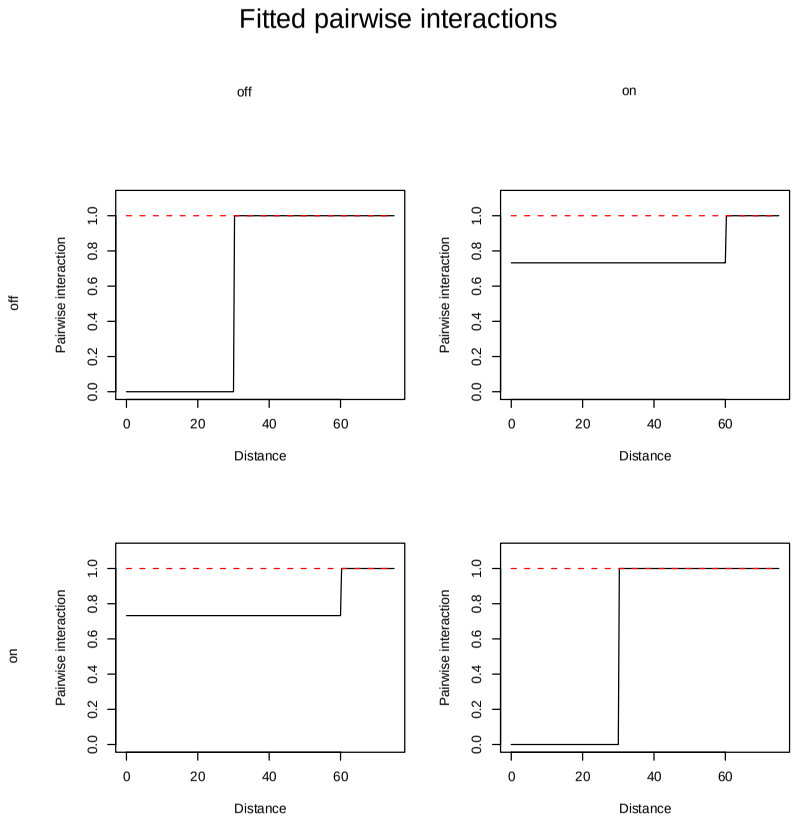
For more detailed explanation and examples of modelling and the interpretation of model formulae for point processes, see [5].

25.3.3 Plotting the fitted interaction

The fitted pairwise interaction in a point process model can be plotted using `fitin`. The value returned by `fitin` is a function array (class `"fasp"`).

```
> model <- ppm(betacells, ~polynom(x, y, 3), MultiStrauss(c("off",
+     "on"), r), rbord = 60)
```

```
> plot(fitin(model))
```

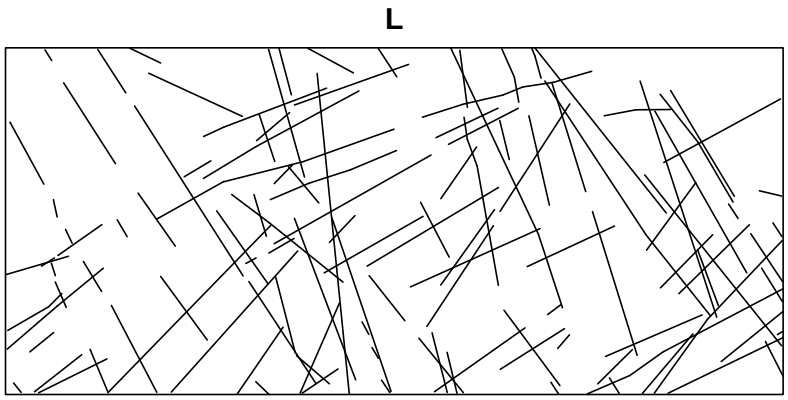


26 Line segment data

`spatstat` also has some facilities for handling spatial patterns of *line segments*.

For example, the `copper` dataset in `spatstat` contains a dataset `copper$Lines` that records the locations of geological faults in a survey region.

```
> data(copper)
> L <- copper$Lines
> L <- rotate(L, pi/2)
> plot(L)
```



A spatial pattern of line segments is represented by an object of class `"psp"`. It consists of a list of line segments (given by the coordinates of their two endpoints), and a window in which the line segments were observed. The line segments may also carry marks.

Objects of class `"psp"` can be created by the function `psp` or obtained by converting other data using the function `as.psp`.

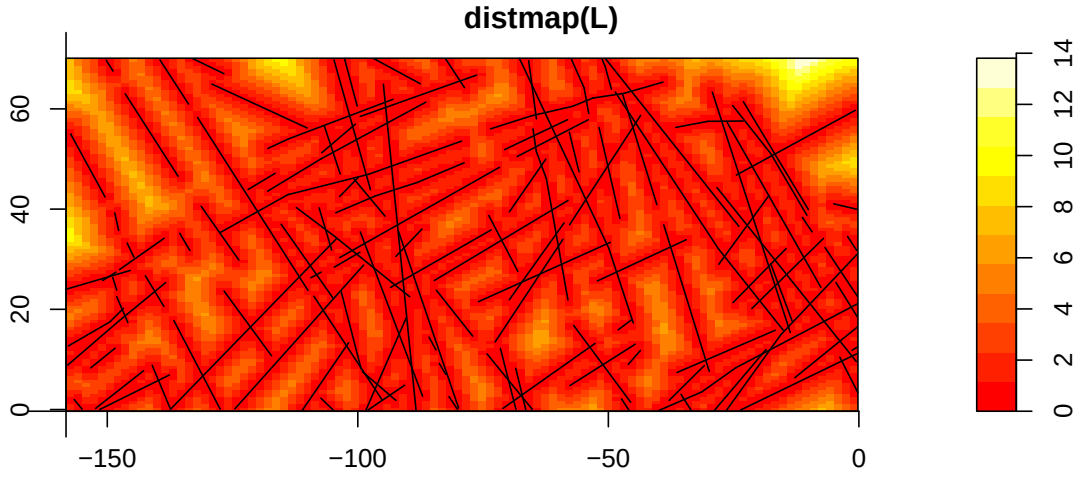
Capabilities available for this class include:

`[.psp`	subset operator (also performs clipping)
`marks.psp`	extract marks
`endpoints.psp`	extract midpoints of line segments
`midpoints.psp`	compute midpoints of line segments
`lengths.psp`	compute lengths of line segments
`angles.psp`	compute angles of orientation for line segments
`rotate.psp`	rotate a line segment pattern
`shift.psp`	shift a line segment pattern
`affine.psp`	apply affine transformation
`pairdist.psp`	distances between line segments
`crossdist.psp`	distances between line segments
`nndist.psp`	closest distances between line segments
`density.psp`	kernel-smoothed intensity image
`crossing.psp`	find intersection points between line segments
`selfcrossing.psp`	find intersection points between line segments
`unitname.psp`	determine units of length
`rescale.psp`	change units of length
`rshift.psp`	apply random shift to each line segment

There are also the usual methods

`plot.psp`	plot a line segment pattern
`print.psp`	print information on a line segment pattern
`summary.psp`	compute summary of a line segment pattern

```
> summary(L)

146 line segments
Lengths:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
 0.09242  6.61400 12.18000 15.02000 19.95000 65.48000
Total length: 2192.57251480451 km
Length per unit area: 0.196937548404655
Angles (radians):
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.008107 0.549500 1.747000 1.378000 2.113000 2.912000
Window: polygonal boundary
single connected closed polygon with 4 vertices
enclosing rectangle: [-158.23, -0.19] x [-0.335, 70.11] km
Window area =  11133.3 square km
Unit of length: 1 km

> plot(distmap(L))
> plot(L, add = TRUE)
```



27 Further information on `spatstat`

Help files

For information on a particular command in `spatstat`, consult the online help file by typing `help(`*command*`)`. The help files are detailed and extensive. The complete manual is over 500 pages.

For examples of the use of a particular command, read the examples section in the help file, or type `example(`*command*`)` to see the examples executed.

Quick reference

Type `help(spatstat)` for a quick-reference overview of all the functions available in the package.

For a demonstration of many of the capabilities of `spatstat`, type `demo(spatstat)`.

For a visual display of all the datasets supplied in `spatstat`, type `demo(data)`.

Website

The website `www.spatstat.org` contains information on recent updates to the package, frequently-asked questions, bug fixes, literature and other developments.

Modelling

For examples on fitting point process models, see [5].

Citation

If you use `spatstat` in a research publication, it would be much appreciated if you could cite the paper [4], or mention `spatstat` in the acknowledgements.

In doing so, you will help us to justify the expenditure of time and effort on maintaining and developing the package.

Citation details are also available in the package by typing `citation(package="spatstat")`.

Queries and requests

If you have difficulty in getting the package to do what you want, or if you have a suggestion for additional features that could be added, please contact the package authors, `adrian@maths.uwa.edu.au` and `r.turner@auckland.ac.nz`, or email the R special interest group in spatial and geographical statistics, `r-sig-geo@stat.math.ethz.ch`.

References

[1] A. Baddeley, J. Møller, and A.G. Pakes. Properties of residuals for spatial point processes. *Annals of the Institute of Statistical Mathematics*, 2007. To appear. Accepted for publication 6 July 2007.

[2] A. Baddeley, J. Møller, and R. Waagepetersen. Non- and semiparametric estimation of interaction in inhomogeneous point patterns. *Statistica Neerlandica*, 54(3):329–350, November 2000.

[3] A. Baddeley and R. Turner. Practical maximum pseudolikelihood for spatial point patterns (with discussion). *Australian and New Zealand Journal of Statistics*, 42(3):283–322, 2000.

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

[5] A. Baddeley and R. Turner. 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*, number 185 in Lecture Notes in Statistics, pages 23–74. Springer-Verlag, New York, 2006. ISBN: 0-387-28311-0.

[6] A. Baddeley, R. Turner, J. Møller, and M. Hazelton. Residual analysis for spatial point processes (with discussion). *Journal of the Royal Statistical Society, series B*, 67(5):617–666, 2005.

[7] A.J. Baddeley. Spatial sampling and censoring. In O.E. Barndorff-Nielsen, W.S. Kendall, and M.N.M. van Lieshout, editors, *Stochastic Geometry: Likelihood and Computation*, chapter 2, pages 37–78. Chapman and Hall, London, 1998.

[8] A.J. Baddeley and J. Møller. Nearest-neighbour Markov point processes and random sets. *International Statistical Review*, 57:89–121, 1989.

[9] A.J. Baddeley, R.A. Moyeed, C.V. Howard, and A. Boyde. Analysis of a three-dimensional point pattern with replication. *Applied Statistics*, 42(4):641–668, 1993.

[10] A.J. Baddeley and B.W. Silverman. A cautionary example on the use of second-order methods for analyzing point patterns. *Biometrics*, 40:1089–1094, 1984.

[11] A.J. Baddeley and M.N.M. van Lieshout. Area-interaction point processes. *Annals of the Institute of Statistical Mathematics*, 47:601–619, 1995.

[12] M. Bell and G. Grunwald. Mixed models for the analysis of replicated spatial point patterns. *Biostatistics*, 5:633–648, 2004.

[13] M. Berman and T.R. Turner. Approximating point process likelihoods with GLIM. *Applied Statistics*, 41:31–38, 1992.

[14] J. Besag and P.J. Diggle. Simple Monte Carlo tests for spatial pattern. *Applied Statistics*, 26:327–333, 1977.

[15] J.E. Besag and P. Clifford. Generalized Monte Carlo significance tests. *Biometrika*, 76:633–642, 1989.

[16] D.R. Brillinger. Comparative aspects of the study of ordinary time series and of point processes. In P.R. Krishnaiah, editor, *Developments in Statistics*, pages 33–133. Academic Press, 1978.

[17] N.A.C. Cressie. *Statistics for Spatial Data*. John Wiley and Sons, New York, 1991.

[18] D.J. Daley and D. Vere-Jones. *An Introduction to the Theory of Point Processes*. Springer Verlag, New York, 1988.

[19] P.J. Diggle. *Statistical analysis of spatial point patterns*. Academic Press, London, 1983.

[20] P.J. Diggle. A point process modelling approach to raised incidence of a rare phenomenon in the vicinity of a prespecified point. *Journal of the Royal Statistical Society, series A*, 153:349–362, 1990.

[21] P.J. Diggle. *Statistical Analysis of Spatial Point Patterns*. Arnold, second edition, 2003.

[22] P.J. Diggle, N. Lange, and F. M. Benes. Analysis of variance for replicated spatial point patterns in clinical neuroanatomy. *Journal of the American Statistical Association*, 86:618–625, 1991.

[23] P.J. Diggle, J. Mateu, and H.E. Clough. A comparison between parametric and non-parametric approaches to the analysis of replicated spatial point patterns. *Advances in Applied Probability (SGSA)*, 32:331–343, 2000.

[24] P.J. Diggle and B. Rowlingson. A conditional approach to point process modelling of elevated risk. *Journal of the Royal Statistical Society, series A (Statistics in Society)*, 157(3):433–440, 1994.

[25] A.C.A. Hope. A simplified Monte Carlo significance test procedure. *Journal of the Royal Statistical Society, series B*, 30:582–598, 1968.

[26] C.V. Howard, S. Reid, A.J. Baddeley, and A. Boyde. Unbiased estimation of particle density in the tandem-scanning reflected light microscope. *Journal of Microscopy*, 138:203–212, 1985.

[27] F. Huang and Y. Ogata. Improvements of the maximum pseudo-likelihood estimators in various spatial statistical models. *Journal of Computational and Graphical Statistics*, 8(3):510–530, 1999.

[28] J.F.C. Kingman. *Poisson Processes*. Oxford University Press, 1993.

[29] G.M. Laslett. Censoring and edge effects in areal and line transect sampling of rock joint traces. *Mathematical Geology*, 14:125–140, 1982.

[30] P.A.W. Lewis. Recent results in the statistical analysis of univariate point processes. In P.A.W. Lewis, editor, *Stochastic point processes*, pages 1–54. Wiley, New York, 1972.

[31] J.K. Lindsey. *The analysis of stochastic processes using GLIM*. Springer, Berlin, 1992.

[32] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[33] J. Møller and R.P. Waagepetersen. *Statistical Inference and Simulation for Spatial Point Processes*. Chapman and Hall/CRC, Boca Raton, 2003.

[34] J. Møller and R.P. Waagepetersen. Modern statistics for spatial point processes. Research Report R-2006-12, Department of Mathematical Sciences, Aalborg University, April 2006. Submitted for publication.

[35] Y. Ogata. Statistical models for earthquake occurrences and residual analysis for point processes. *Journal of the American Statistical Association*, 83:9–27, 1988.

[36] B.D. Ripley. Modelling spatial patterns (with discussion). *Journal of the Royal Statistical Society, series B*, 39:172–212, 1977.

[37] B.D. Ripley. Simulating spatial patterns: dependent samples from a multivariate density. *Applied Statistics*, 28:109–112, 1979.

[38] B.D. Ripley. *Spatial Statistics*. John Wiley and Sons, New York, 1981.

[39] B.D. Ripley. *Statistical Inference for Spatial Processes*. Cambridge University Press, 1988.

[40] A. Särkkä. *Pseudo-likelihood approach for pair potential estimation of Gibbs processes*. Number 22 in Jyväskylä Studies in Computer Science, Economics and Statistics. University of Jyväskylä, 1993.

[41] D. Stoyan and P. Grabarnik. Second-order characteristics for stochastic structures connected with Gibbs point processes. *Mathematische Nachrichten*, 151:95–100, 1991.

[42] D. Stoyan and H. Stoyan. *Fractals, Random Shapes and Point Fields*. John Wiley and Sons, Chichester, 1995.

[43] M.N.M. van Lieshout. *Markov Point Processes and their Applications*. Imperial College Press, 2000.

[44] M.N.M. van Lieshout and A.J. Baddeley. A nonparametric measure of spatial interaction in point patterns. *Statistica Neerlandica*, 50:344–361, 1996.

[45] R. Waagepetersen. An estimating function approach to inference for inhomogeneous Neyman-Scott processes. Submitted for publication, 2006.

Index

analysis of deviance, 65

binary mask, 26, 42

circular windows, 40
classes, 25
    in R, 25
    in spatstat, 25
`clickppp`, 24
complete spatial randomness, 53
    and independence, 130, 151
    definition, 53
    Kolmogorov-Smirnov test, 56
    quadrat counting test, 55
conditional intensity, 113
    for marked point processes, 157
contrasts, 61, 155
covariate effects, 8
covariates, 6, 15, 61
    in `ppm`, 61
Cox process, 80
CSRI, 130, 151
    conditional intensity, 157
    fitting to data, 154
    simulating, 152

data entry, 31
    at the terminal, 31
    basic, 31, 32
    checking, 34
    from file, 32
    marked point patterns, 133
    marks, 32
    point-and-click, 24
datasets
    inspecting, 19
    provided in spatstat, 24
dispatching, 25
distance methods, 83
distances
    empty space, 83, 84
    nearest neighbour, 83, 90
    pairwise, 83, 92
`distmap`, 83

edge effects, 85
empty space distances, 83, 84
empty space function, 85
envelopes, 98
    and Monte Carlo tests, 98
    for any fitted model, 101
    for any simulation procedure, 101
    in `spatstat`, 98
    of summary functions, 98
exploratory data analysis, 20
    for marked point patterns, 138

fitted model, 119
    goodness-of-fit, 67, 125
    interpretation of coefficients, 61
    methods for, 63
    residuals, 68, 127
    simulation of, 66
fitting models
    by Huang-Ogata method, 124
    maximum pseudolikelihood, 116
    to marked point patterns, 154, 159
    via summary statistics, 98, 102
`fv`, 30

geometrical transformations, 49
Gibbs models, 109
    area-interaction, 112
    Diggle-Gates-Stibbard, 112
    Diggle-Gratton, 112
    fitting, 116
      by Huang-Ogata method, 124
      maximum pseudolikelihood, 116
      `ppm`, 116
    fitting to marked point patterns, 159
    goodness-of-fit, 125
    hard core process, 110
    in spatstat, 118
    infinite order interaction, 112
    multitype, 157
      maximum pseudolikelihood, 159
    multitype pairwise interaction, 157
    pairwise interaction, 112
    residuals, 127
    simulation, 114
    simulation of fitted model, 121
    soft core, 112
    Strauss process, 111
    Strauss-hard core, 112

goodness-of-fit, 67
  for fitted Gibbs model, 125
  for Poisson models, 67

hard core process, 110
  multitype, 158
Huang-Ogata method, 124

`im`, 25, 74
images, 74
  computing with, 78
  creating, 74
    from raw data, 74
  exploratory inspection of, 76
  extracting subset, 77
  plotting, 76
  returned by a function, 75
independence of components, 130, 148
intensity
  function, 37
    kernel estimator, 37
  homogeneous, 36
  inhomogeneous, 37
  investigation of, 36
  measure, 37
  of marked point process, 138
interaction, 7, 10
  distance methods, 83
  in `spatstat`, 118
  multitype, 157, 159
    in `spatstat`, 159
  plotting a fitted interaction, 160
  Q–Q plot, 73
  simple models, 79
  summary functions, 83

K function, 21, 92
  for multitype point pattern, 142
  inhomogeneous, 105
kernel estimator of intensity, 37, 38
kernel smoothing of marks, 140
Kolmogorov-Smirnov test
  of CSR, 56
  of inhomogeneous Poisson, 68

line segments, 162
lurking variable plot, 70

mark correlation function, 146
marked point patterns
  cutting marks into bands, 136
  data entry, 133
  exploratory data analysis, 138
  exploring marks, 140
  inspecting, 134
  joint and conditional analysis, 130
  manipulating, 136
  methodological issues, 130
  model-fitting, 154, 159
  probabilistic formulation, 129
  randomisation tests, 130
  separating into types, 136
  summary functions, 142
marked point process
  intensity, 138
marks, 5, 14, 129
  categorical, 33
  data entry, 31, 32
  exploratory data analysis, 140
  manipulating, 136
  operations on, 48
  smoothing, 140
  spatial trend in, 140
  versus covariates, 14
`markstat`, 142
`marktable`, 141
Matern cluster process, 79
maximum likelihood, 58
maximum pseudolikelihood, 116, 159
  for multitype Gibbs models, 159
  improvements over, 124
methods, 25
  default method, 27
  dispatch, 25
minimum contrast, 98, 102
model validation, 67, 125
Monte Carlo test, 98
  pointwise, 98
  simultaneous, 99
multitype hard core process, 158
multitype point pattern, 9, 10, 21, 33
multitype point patterns
  separating into types, 136
  summary functions, 142
multitype Strauss process, 158

nearest neighbour distances, 83, 90
`nndist`, 83
nuisance parameters, 122

owin, 25, 40

pairdist, 83
pairwise distances, 83, 92
pairwise interaction process, 110
point pattern, 5
  marked, 129
  marks, 5, 14
  multitype, 9, 10
  needs window, 47
  point process model for, 12
  standard model, 13
point process, 12
point process models
  area-interaction, 112
  Diggle-Gates-Stibbard, 112
  Diggle-Gratton, 112
  Gibbs, 109
  hard core, 110
  infinite order interaction, 112
  pairwise interaction, 110, 112
  soft core, 112
  Strauss, 111
  Strauss-hard core, 112
Poisson cluster processes, 79
Poisson models
  fitting, 59
  goodness-of-fit, 67
  homogeneous, 53
  inhomogeneous, 58
  log-likelihood, 59
  marked, 151
  maximum likelihood, 58
  residuals, 68
Poisson point process
  homogeneous
    definition, 53
    simulation, 53
  inhomogeneous
    definition, 58
    fitting, 59
    likelihood, 59
    motivation, 58
    simulation, 58
Poisson-derived models, 79
polygonal windows, 26, 41
ppm, 63, 119
  marked Gibbs point process models, 159
  marked Poisson point process models, 154
  methods for, 63
ppp, 25
  combining several, 49
  extracting subset, 47
  format, 45
  geometrical transformations, 49
  in arbitrary window, 44
  manipulating, 45
  needs window, 47
  operations on, 47
  ways to make, 35
probability density, 109
profile pseudolikelihood, 122
pseudolikelihood, 116
  profile pseudolikelihood, 122

quadrat counting, 20, 37
quadrat counting test
  of CSR, 55
quadrat test
  of inhomogeneous Poisson, 67

R, 16
  contributed packages, 17
  where to get, 16
random labelling, 130, 149
random thinning, 58
randomisation tests, 130, 147
  for marked point patterns, 147
rectangular windows, 26, 40
residuals, 68, 127
  for fitted Gibbs model, 127
  for Poisson models, 68
  lurking variable plot, 70
  Q–Q plot, 72
  smoothed residual field, 70
return value, 28
rpoispp, 53, 58
runifpoint, 54

sequential models, 81
simulation
  of fitted Gibbs model, 121
  of fitted Poisson model, 66
smoothed residual field, 70
spatstat, 18, 164
  citing, 18
  getting started, 18

installing, 18
`split`, 23
standard model, 13
Strauss process, 111
fitting to data, 117
multitype, 158
summary functions, 83
and Monte Carlo tests, 98
critique, 96
edge effects, 85
envelopes, 98
F, 85
for multitype point patterns, 142
G, 90
inference using, 98
inhomogeneous K, 105
J, 95
K, 92
L, 93
mark correlation, 146
model-fitting with, 102
pair correlation, 93

tests
χ^2 quadrat counting, 55
Kolmogorov-Smirnov, 56, 68
Monte Carlo, 98
thinning, 80
Thomas process, 79
tips, 25, 29, 34, 48, 84, 87, 99, 133
treatment contrasts, 61

`unitname`, 35
units of length, 35

validation, 67, 125

windows, 40
binary mask, 26, 42
circular, 40
needed in any point pattern, 47
operations on, 44
polygonal, 26, 41
rectangular, 26, 40
returned by functions, 43

χ^2 quadrat counting test, 55