9

Poisson Models

Part III deals with statistical modelling and inference for point pattern data, starting in this chapter with Poisson point process models.

9.1 Introduction

Poisson point processes were introduced briefly in Chapter 5. In this chapter we show how to use them as statistical models for point pattern data. We show how to formulate a Poisson model appropriate to the data, fit the model to the data, and interpret the results. Validation of a fitted model is discussed in chapters 10 and 11.

The key property of a Poisson process is that the random points are *independent* of each other. This property greatly simplifies the statistical analysis, and gives access to a wide variety of powerful statistical techniques.

A Poisson process is completely described by its intensity function $\lambda(u)$. Statistical models of Poisson point processes are easy to build: they are simply models of the intensity, that is, a model is just an equation stating the form of the function $\lambda(u)$. This also means that there is a close relationship between techniques for estimating an intensity function (Chapter 6) and methods for fitting a Poisson point process model.

Why Poisson?

The huge literature on spatial analysis contains many other techniques for analysing spatial point pattern data. Many of these techniques appear at first sight to have nothing to do with Poisson point processes — and even seem to have nothing to do with point processes at all.

In fact, several popular techniques are *equivalent* to fitting a Poisson point process model with a specific form of the intensity — namely, where the intensity is a loglinear function of the covariates. Such techniques include logistic regression [4, 658, 604, 326, 397, 116] and maximum entropy [248, 246, 252, 546].

Logistic regression is commonly used in GIS to predict the probability of occurrence of mineral deposits [4, 152], archaeological finds [604, 397], landslides [153, 288], and other events which can be treated as points at the scale of interest. The study region is divided into pixels; in each pixel the presence or absence of any data points is recorded; then logistic regression [458, 240, 344] is used to predict the probability of the presence of a point as a function of predictor variables.

Logistic regression implicitly assumes that the presence/absence values for different pixels are independent [458, 240, 344]. This implies that the underlying point process is Poisson (as explained in Chapter 5). Furthermore, the *logistic* relationship implies that the intensity of the Poisson process is a *loglinear* function of the covariates and the parameters. Thus, logistic regression is essentially equivalent to fitting a Poisson point process model with *loglinear* intensity [34, 691].

The principle of maximum entropy [248] is often used in ecology, for example, to study the influence of habitat variables on the spatial distribution of animals or plants [246, 252, 546]. Con-

SAMPLE

ceptually we consider all possible spatial distributions, and find the spatial distribution which maximises a quantity called entropy, subject to constraints implied by the observed data. Surprisingly, the solution is a probability density which is a *loglinear* function of the covariates. An analogy could be drawn with the stretching of a string: a string may take on any shape, but if we demand that the string be stretched as tight as possible, it will take up a straight line. Thus, again, this analysis principle is equivalent to fitting a Poisson point process model with loglinear intensity [568].

Advantages of statistical modelling

Statistical modelling is a much more powerful way to analyse data than simply computing summary statistics. Formulating a statistical model, and fitting it to the data, allows us to adjust for effects that might otherwise distort the analysis (such as uneven distribution of survey effort, and spatially varying population density) by including terms that represent these effects. By fitting different models that include or omit a particular term, and comparing the models, we can decide which variables have a statistically significant influence on the intensity.

A great advantage of statistical modelling is that the assumptions of the analysis are openly stated, rather than implicitly imposed. All model assumptions are 'on the table' and are amenable to criticism. Chapter 11 provides tools for criticising each individual model assumption of a Poisson point process model, and for identifying weaknesses of the data analysis.

9.2 Poisson point process models

Poisson processes were introduced briefly in Sections 5.3 and 5.4 of Chapter 5. Here we discuss using Poisson processes as statistical models for data analysis of spatial point patterns.

9.2.1 Characteristic Properties of the Poisson Process

We start with a detailed statement of the characteristic properties of the Poisson point process. Apart from clarifying our understanding of the model, this is a useful checklist of properties that need to be verified before the analyst can be satisfied that a real point pattern dataset is well described by a Poisson point process.

Homogeneous Poisson process (CSR)

The *homogeneous Poisson process* with intensity $\lambda > 0$ (also called *'complete spatial randomness', CSR*) has the properties that

(PP1) **Poisson counts:** the number $n(\mathbf{X} \cap B)$ of points falling in any region B has a Poisson distribution (see page 135);

(PP2) **homogeneous intensity:** the expected number of points falling in B is $\mathbb{E}[n(\mathbf{X} \cap B)] = \lambda \cdot |B|$;

(PP3) **independence:** if $B_1, B_2, \ldots$ are disjoint regions of space then $n(\mathbf{X} \cap B_1)$, $n(\mathbf{X} \cap B_2)$, ... are independent random variables;

(PP4) **conditional property:** given that $n(\mathbf{X} \cap B) = n$, the n points are independent and uniformly distributed in B.

We are justified in calling λ the 'intensity' because (PP2) implies that this point process has homogeneous intensity λ in the sense of Section 6.2. The intensity parameter λ has dimensions

SAMPLE

length^{-2} and determines the average number of points per unit area. Scaling properties of λ (for example, the effect of changing the unit of length) are discussed in Section 6.2. Any value can be specified for λ, as long as it is non-negative and finite. If $\lambda = 0$ then all realisations of the point process are empty.

The properties (PP1)–(PP4) provide the basis for many statistical techniques. For example, in quadrat counting, (PP3) implies that the counts in different quadrats are independent random variables; (PP1) implies that these counts have Poisson distributions; and (PP2) implies that the mean values of the counts are proportional to the areas of the quadrats. If the quadrats have equal area, then the quadrat counts are Poisson random variables with equal mean values, so they are identically distributed. This justifies the use of the χ^2 test.

When we need to check whether a point process is CSR, the most important properties to check are independence (PP3) and homogeneity (PP2). As explained in Section 5.3, the other properties follow logically from these, under reasonable conditions.[1] However, if any one of these properties does not hold, then the process is not CSR. Statistical methods for checking CSR can focus, for example, on detecting a departure from the Poisson distribution (PP1) or from conditional independence and uniformity (PP4).

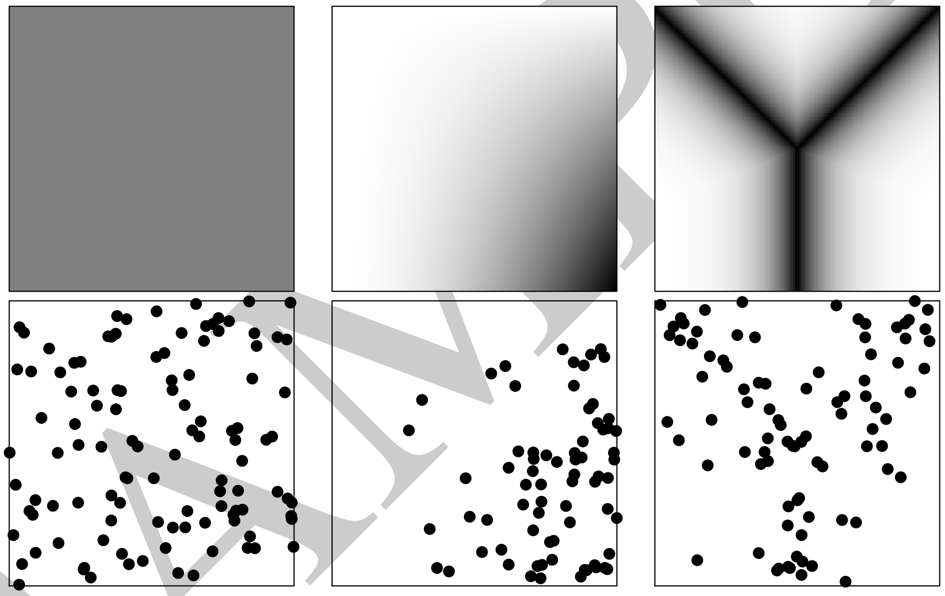


Figure 9.1. *Examples of intensity functions (top) and simulated realisations (bottom) for Poisson processes.*

SAMPLE

Inhomogeneous Poisson process

The *inhomogeneous* Poisson process with intensity function $\lambda(u)$, $u \in \mathbb{R}^2$, is a modification of the homogeneous Poisson process, in which properties (PP2) and (PP4) above are replaced by

(PP2′): the number $n(\mathbf{X} \cap B)$ of points falling in a region B has expected value

$$\mathbb{E}[n(\mathbf{X} \cap B)] = \int_B \lambda(u)\,\mathrm{d}u. \tag{9.1}$$

[1] The reasonable condition is that the probability of *more than one* random point falling in a region B, divided by the area of B, falls to zero as the area of B goes to zero. This excludes coincident points and fractal-like behaviour.

(PP4′): given that $n(\mathbf{X} \cap B) = n$, the n points are independent and identically distributed, with common probability density

$$f(u) = \frac{\lambda(u)}{I} \tag{9.2}$$

where $I = \int_B \lambda(u)\,\mathrm{d}u$.

Again we are justified in calling $\lambda(u)$ the 'intensity function' because, by virtue of (PP2′), it is the intensity function in the sense of Section 6.3. Its values have dimension *length*$^{-2}$ and determine the spatially varying intensity (average number of points per unit area).

The intensity function encapsulates both the abundance of points (by equation (9.1)) and the spatial distribution of individual point locations (by equation (9.2)).

Any function $\lambda(u)$ can be used, as long as its values are non-negative and *integrable* (the right-hand side of (9.1) must be finite).

9.2.2 Modelling scenarios giving rise to a Poisson process

There are many realistic models of physical processes which give rise to a Poisson point process, because the Poisson limit law is so widely applicable. We sketch some common examples below.

Random trials on a very fine grid

Imagine a radioactive material, consisting of N atoms arranged in a regular grid, where N is very large, and the spacing between atoms is tiny. In a specified time interval, each atom has a very small probability p of undergoing random decay. Decay events of different atoms are independent. The total number of decay events in the specified period is the number of successes in N trials with success probability p. Since N is large and p is small, the number of decay events has a Poisson distribution (to a very good approximation) and the locations of the decay events approximately constitute a Poisson point process.

Independent random thinning

Suppose that there is some initial population represented by a point process $\mathbf{X}$ (not assumed to be a Poisson process) and that each point of $\mathbf{X}$ is either deleted or retained, independently of other points. Suppose $\mathbf{X}$ has intensity function $\beta(u)$ and the probability of retaining a point at the location u is $p(u)$. The resulting process $\mathbf{Y}$ of retained points has intensity $\lambda(u) = p(u)\beta(u)$. If the original population density $\beta(u)$ is large and the retention probabilities $p(u)$ are small, then the resulting process $\mathbf{Y}$ is approximately an inhomogeneous Poisson process.

A model of plant propagation could assume that seeds are randomly dispersed according to some point process, and seeds randomly germinate or do not germinate, independently of each other, with a germination probability that depends on the local soil conditions. The resulting pattern of plants is well described by an inhomogeneous Poisson process.

In accident research, some models describe a population of 'near-accidents' or 'potential accidents', each of which has a small probability of progressing to a real accident. The thinning principle says that the real accidents are well described by an inhomogeneous Poisson process. Similarly in a galaxy survey of the distant universe, we may fail to detect some individual galaxies. The detected locations of very faint galaxies could be described by an inhomogeneous Poisson process.

In spatial epidemiology, $\mathbf{X}$ could represent the human population at risk of a rare disease, and $\mathbf{Y}$ would be the disease cases. The key assumption is that a person's disease status is statistically independent of other people.

SAMPLE

Random strewing

Suppose a large number N of points is scattered randomly in a large region D according to a binomial point process. That is, each point is uniformly distributed over D, and different points are independent. If this random pattern is observed within a subregion W, where D is much larger than W, then the observed pattern is approximately a Poisson point process inside W.

Random displacement

Again we imagine an initial population represented by any point process $\mathbf{X}$. Suppose that each point in $\mathbf{X}$ is subjected to a random displacement, independently of other points. A point at the original location x_i is moved to the new location $y_i = x_i + V_i$ where V_i is a random vector. Assume that $V_1, V_2, \ldots$ are independent random vectors. Then under reasonable conditions, the resulting point process $\mathbf{Y}$ is approximately an inhomogeneous Poisson process.

Spatial birth-and-death process

Consider a forest which evolves over time. In each small interval of time Δt, each existing tree has probability $m\Delta t$ of dying, independently of other trees, where m is the 'mortality rate' per tree per unit time. In the same time interval, in any small region of area Δa, a new tree germinates with probability $g\Delta a\Delta t$, independently of any existing trees, where g is the 'germination rate' per unit area per unit time. No matter what the initial state of the forest, over long time scales this 'spatial birth-and-death process' reaches an equilibrium in which any snapshot of the forest (the pattern of trees in existence at a fixed time) is a realisation of a Poisson process with intensity g/m.

9.2.3 Common parametric models of intensity

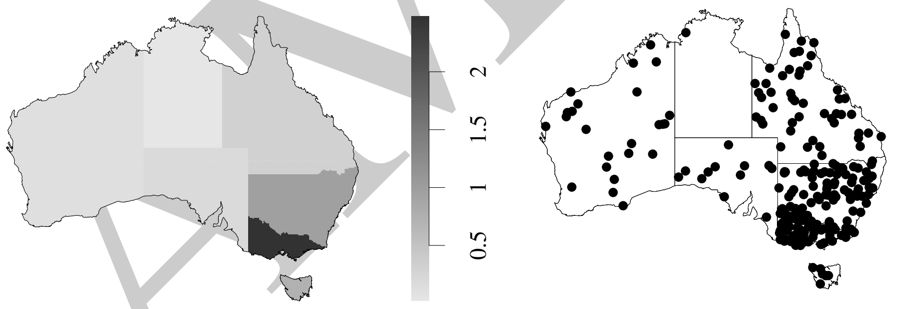


Figure 9.2. *Intensity (*Left*) and simulated realisation (*Right*) of the Poisson process with a different, homogeneous intensity in each Australian state and territory. Intensity is proportional to average population density within the state or territory.*

The intensity function $\lambda(u)$ of the Poisson process can be assumed to take any form that is appropriate to the context, so long as the intensity values are non-negative and the integral of the intensity over the observation window is finite. Some common models of the intensity function, and their interpretation, are discussed below.

homogeneous intensity: the Poisson process with constant intensity $\lambda(u) \equiv \lambda$ is the simplest model, called the homogeneous Poisson process or 'complete spatial randomness' (CSR).

homogeneous in different regions: Space is partitioned into non-overlapping regions $B_1, \ldots, B_m$. Within region B_j, the process is Poisson with intensity β_j, where $\beta_1, \ldots, \beta_m$ are parameters of

SAMPLE

the model. Then the overall process is an inhomogeneous Poisson process where the intensity function $\lambda(u)$ takes the value β_j when u is in region B_j. See Figure 9.2.

intensity proportional to baseline: A simple model is

$$\lambda(u) = \theta b(u) \tag{9.3}$$

where $b(u)$ is a known function ('baseline') and θ is an unknown parameter that must be estimated. If the baseline $b(u)$ represents the spatially varying density of a population, and we assume that each member of the population has equal chance θ of contracting a rare disease, then the cases of the disease will form a Poisson process with intensity (9.3), the 'constant risk' model. Modelling scenarios which involve random thinning often lead to a model of this form.

exponential function of covariate: A very important model is

$$\lambda(u) = \kappa e^{\beta Z(u)} = \exp(\alpha + \beta Z(u)) \tag{9.4}$$

where $Z(u)$ is a spatial covariate, and α, β, κ are parameters to be estimated. This is the model that is fitted in basic applications of logistic regression and maximum entropy. An important example is in prospectivity analysis, where $Z(u)$ is the distance from location u to the nearest geological fault. The parameter $\kappa = e^{\alpha}$ gives the intensity at locations where $Z(u) \approx 0$ (i.e. locations close to a geological fault), while the intensity changes by a factor e^{β} for every 1 unit of distance away from the faults. The Berman-Lawson-Waller test (Section 10.3.5) for a covariate effect is equivalent to testing whether $\beta = 0$ in this model.

raised incidence model: A combination of the last two models is

$$\lambda(u) = b(u)e^{\alpha+\beta Z(u)} = b(u)\exp(\alpha + \beta Z(u)) \tag{9.5}$$

where $b(u)$ is a known baseline function and $Z(u)$ is a spatial covariate. This model expresses the covariate effect relative to the baseline $b(u)$. For example in spatial epidemiology, $b(u)$ could be the spatially varying population density, and the term $\exp(\alpha + \beta Z(u))$ would then be the disease risk per person, depending on the value of Z. The exponential term could be replaced by something more complicated [224]. Modelling scenarios that involve independent random thinning often lead to models of this kind.

loglinear model: Embracing all of the models listed above is the general loglinear model

$$\lambda_{\boldsymbol{\theta}}(u) = \exp(B(u) + \boldsymbol{\theta}^{\top}\mathbf{Z}(u)) = \exp(B(u) + \theta_1 Z_1(u) + \theta_2 Z_2(u) + \ldots + \theta_p Z_p(u)) \tag{9.6}$$

where $B(u)$ and $Z_1(u), \ldots, Z_p(u)$ are known functions, $\theta_1, \ldots, \theta_p$ are parameters to be estimated, and we write $\mathbf{Z}(u) = (Z_1(u), \ldots, Z_p(u))$ for the vector-valued function. This was termed a *modulated* Poisson process by Cox [177].

As this sequence of models shows, we often build models for the intensity by multiplying together several terms which represent different effects. The logarithm of intensity is a sum of terms representing different effects. We can even include a term in the model that accounts for the probability of observing a point (e.g., related to the survey effort).

There are plausible modelling assumptions that give rise to Poisson point process models of either *additive* or *multiplicative* form. A model of the additive form has an intensity function

$$\lambda(u) = \gamma_1(u) + \gamma_2(u) + \ldots$$

where the terms $\gamma_k(u)$ are interpretable as the intensities of Poisson processes of points associated

SAMPLE

with different ‘origins’ or ‘causes’, that were superimposed to yield the observed point process. A model of multiplicative form has

$$\lambda(u) = \exp(A(u) + B_1(u) + B_2(u) + \ldots)$$

where the first term $A(u)$ is a baseline intensity (on a log scale) and the subsequent terms $B_1(u)$, $B_2(u)$, ... can be interpreted as ‘thinning’ or ‘modulation’ effects.

The multiplicative form is more convenient for inference, and corresponds directly to the canonical choice of the log link in a Poisson Generalised Linear Model. Inference for models of the additive form is more difficult as it effectively leads to a missing value problem, the missing information being the attribution of each observed point to one of the component Poisson processes.

9.3 Fitting Poisson models in `spatstat`

Fitting a statistical model to data means choosing values for the parameters governing the model, so that the model fits the data as well as possible. For example, to fit a line $y = ax + b$ through a scatter of data points, we find the values of the parameters a (slope) and b (intercept) that give the best-fitting line.

To fit a Poisson point process model to a point pattern dataset, we would specify the form of the intensity function — perhaps using one of the common models described above — and find the values of the parameters which best fit the point pattern dataset. The definition of ‘best fit’ and the technical details of model-fitting are described in Sections 9.7–9.10. This section explains how to use `spatstat` to fit Poisson models to point pattern data in practice.

9.3.1 The `ppm` function

Syntax

Fitting a Poisson point process model to a point pattern dataset is performed in `spatstat` by the function `ppm` (for **p**oint **p**rocess **m**odel). This is closely analogous to the standard model fitting functions in R such as `lm` (for fitting linear models) and `glm` (for fitting generalized linear models). The essential syntax for specifying a Poisson point process model to be fitted by `ppm` is

```
ppm(X ~ trend, ...)
```

where the first argument ‘`X ~ trend`’ is a `formula` in the R language (Section 2.1.10).

The left-hand side of the formula gives the name of the point pattern dataset `X`. Alternatively the left-hand side may be an expression which can be evaluated to yield a point pattern.

The right-hand side of the model formula specifies the form of the *logarithm* of the intensity function for the model. Thus, any *loglinear* intensity model (9.6) can be fitted.[2]

Simplest example

The simplest possible model for a Poisson planar point process is the constant intensity or homogeneous model, equivalent to complete spatial randomness. A call to `ppm` to fit this model employs the simplest possible formula, `X ~ 1`. We illustrate this using the *Beilschmiedia* forest data (Figure 1.10) available in `spatstat` as the `"ppp"` object `bei`.

[2]Fitting Poisson models which are *not* loglinear is discussed in Section 9.12.

SAMPLE

```
> fit <- ppm(bei ~ 1)
> fit

Stationary Poisson process
Intensity: 0.007208
             Estimate    S.E. CI95.lo CI95.hi Ztest   Zval
log(lambda)    -4.933 0.01666  -4.965    -4.9   *** -296.1
```

The result of ppm is a fitted point process model, an object belonging to the class "ppm". There are many facilities for handling such objects. For instance there is a print method print.ppm, which produced the output above. This output tells us that the fitted intensity was $\overline{\lambda} = 0.007208$ trees per square metre. For the *logarithm* of the intensity, the output gives an estimate, a standard error, and a 95% confidence interval. The fitted intensity can be recovered from this line of output by exp(-4.933) = 0.007208.

Variables in a formula

A 'variable name' in a formula is a name that plays the role of a variable rather than a function. For example, in the formula Y ~ f(X), the symbols X and Y are recognised by R as playing the role of 'variables', while the symbol f is recognised as playing the role of a function.

In a call to ppm of the form

```
ppm(X ~ trend)
```

all the variable names appearing in the formula should be the names of existing objects in the R session. An alternative is to use the form

```
ppm(X ~ trend, ..., data)
```

where the argument data is a list containing data needed to fit the model. In this case, for each variable name in the formula, ppm will first try to find a matching entry in the list data; if it is not found, then ppm will look for existing objects in the R session.

The left-hand side of the model formula should be either the name of a point pattern, or an expression which can be evaluated to yield a point pattern. On the left side of the formula, mathematical operators such as +, -, *, /, and ^ have their usual mathematical meaning.

The right-hand side of the model formula is an expression involving the names of spatial covariates which affect the intensity of the point process. On the right-hand side of a model formula, the operators +, -, *, /, and ^ have a special interpretation: they are model operators, which are used to combine terms in a model, as explained below.

The variable names x and y are reserved names which represent the Cartesian coordinates. Any other variable name in the formula should match the name of an existing object in the R session, or the name of an entry in the argument data if it is given. Each object should be either *(a)* a pixel image (object of class "im") giving the values of a spatial covariate at a fine grid of locations; *(b)* a function which can be evaluated at any location (x,y) to obtain the value of the spatial covariate. It should be a function(x, y, ...) in the R language; *(c)* a window (object of class "owin") which will be interpreted as a logical variable which is TRUE inside the window and FALSE outside it; *(d)* a tessellation (object of class "tess") which will be interpreted as a factor covariate. For each spatial location, the factor value indicates which tile of the tessellation contains the location in question; or *(e)* a single number, indicating a covariate that is constant in this dataset.

In what follows we explain how to use the ppm function and elaborate on the use of formulae introduced in Section 2.1.10. We illustrate the ideas by examples using point pattern datasets supplied with the spatstat package.

SAMPLE

9.3.2 Models with a single numerical covariate

Recall that the interpretation of a formula depends on the types of variables involved: for example, numerical variables and categorical variables are interpreted differently. We start by dealing with models involving a single numerical covariate and proceed from there to more complicated models. Given a numerical covariate (predictor) `Z`, say, we could use a formula such as `X ~ Z` to specify the model. This formula indicates that the intensity is a loglinear function of `Z` as given by (9.4).

9.3.2.1 Analysis of *Beilschmedia* data

The data set `bei` installed in `spatstat` is a point pattern giving the locations of *Beilschmiedia* trees. Covariate data are supplied in a separate object `bei.extra`, which is a list with two entries, `elev` and `grad`, which are pixel images of values of the terrain elevation and terrain slope, respectively. See Section 6.5.3, page 176 and Section 1.1.4.

```
> bei.extra
List of pixel images
elev:
real-valued pixel image
101 x 201 pixel array (ny, nx)
enclosing rectangle: [-2.5, 1002] x [-2.5, 502.5] metres
grad:
real-valued pixel image
101 x 201 pixel array (ny, nx)
enclosing rectangle: [-2.5, 1002] x [-2.5, 502.5] metres
```

The following command fits a Poisson point process model in which the intensity of *Beilschmiedia* trees is a *loglinear* function of terrain slope:

```
> fit <- ppm(bei ~ grad, data=bei.extra)
> fit
Nonstationary Poisson process
Log intensity:  ~grad
Fitted trend coefficients:
(Intercept)        grad
     -5.391       5.022
```

The model fitted by the commands above is a Poisson point process with intensity

$$\lambda(u) = \exp(\beta_0 + \beta_1 S(u)) \tag{9.7}$$

where $S(u)$ is the terrain slope at location u. The argument to the exp function (in this case $\beta_0 + \beta_1 S(u)$) is referred to as the *'linear predictor'*. The component $\beta_1 S(u)$ is called the *'effect'* of the covariate S.

The printout above[3] includes the fitted coefficients marked by the labels `"(Intercept)"` and `"grad"`. These are the maximum likelihood estimates of β_0 and β_1, respectively, the coefficients of the linear predictor; so the fitted model is

$$\lambda(u) = \exp(-5.391 + 5.022\,S(u)). \tag{9.8}$$

These results tell us that the estimated intensity of *Beilschmiedia* trees on a level surface (slope $S = 0$) is about $\exp(-5.391) = 0.004559$ trees per square metre, or 45.59 trees per hectare, and

[3]To save space on the printed page, we have set `options(digits=4)` so that numbers are printed to four significant digits for the rest of this chapter. We have also set `spatstat.options(terse=2)` to shorten the output.

SAMPLE

would increase by a *factor* of $\exp(5.022) = 151.7$ if the slope increased to 1.0. The largest slope value in the data is about 0.3, at which stage the predicted intensity has risen by a factor of $\exp(0.3 \times 5.022) = 4.511$, that is, more than quadrupled from its value on a level surface.

Although it is possible to read off the fitted model from the printout and calculate intensity levels for various values of the predictors, this can be tedious, and becomes intricate when the model is more complicated. The generic R command `predict` (explained in Section 9.4.3) has a method for point process models (class `"ppm"`), which can be used to calculate fitted intensities and other properties of the model. In this simple model, the fitted intensity can also be plotted as a function of a covariate (in the current example, the covariate `grad`) using the `spatstat` utility `effectfun`, as shown in the left panel of Figure 9.3.

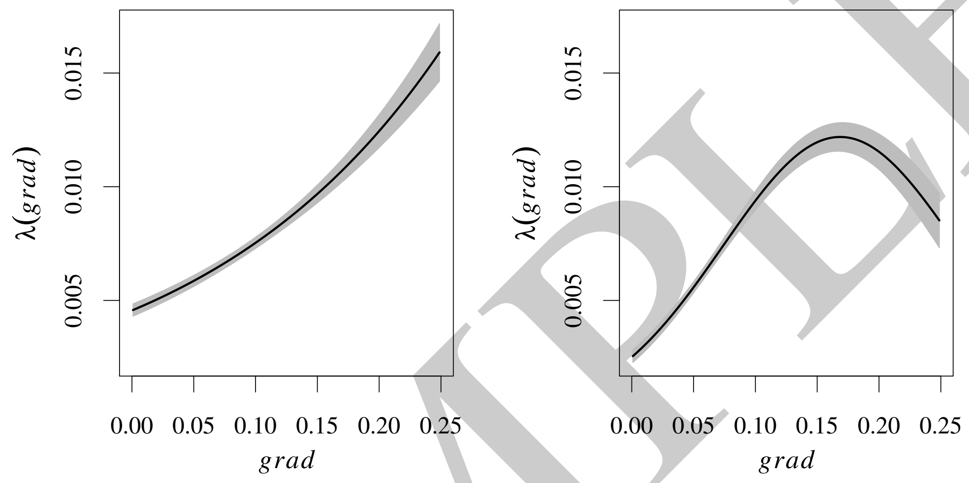


Figure 9.3. *Fitted intensity of* Beilschmiedia *trees as a function of slope. Solid line: maximum likelihood estimate. Shading: pointwise 95% confidence interval. Generated by* `plot(effectfun(fit, "grad", se.fit=TRUE))`. Left*:* `fit` *is a* `"ppm"` *object containing the loglinear model (9.7).* Right*:* `code` *is a* `"ppm"` *object containing the log-quadratic model (9.9).*

It should be clearly understood that the formula `bei ~ grad` does not represent a flexible model in which the abundance of trees depends, in some unspecified way, on the terrain slope. On the contrary, the dependence is very tightly specified: the formula `bei ~ grad` corresponds to the *loglinear* relationship (9.7), that is, it corresponds to assuming intensity is exponentially increasing or decreasing as a function of terrain slope.

If a loglinear relationship is *not* an appropriate assumption, but the intensity is believed to depend on terrain slope somehow, several strategies are available.

One strategy is to transform the slope values and fit a loglinear relationship to the transformed values. Transformations of the original covariates can be specified directly in the model formula. For example, the covariate `grad` gives the gradient as the *tangent* of the angle of inclination (vertical elevation divided by horizontal displacement) so that a gradient of 1.0 corresponds to a 45-degree incline. If we think it is more appropriate to use the *angle* of inclination as the covariate in the loglinear model, we can convert the gradient values to angles using the arctangent, simply by typing

```
> ppm(bei ~ atan(grad), data=bei.extra)
```

Any mathematical function can appear in a model formula. However, care is required with expressions that involve the symbols `+`, `-`, `*`, `/`, and `^`, which have special syntactic meanings in a formula, except inside a function call.

SAMPLE

For example, `atan` calculates angles in radians; to convert them to degrees, we would need to multiply by $180/\pi$, so the covariate should be `atan(grad) * 180/pi`. To prevent the mathematical symbols `*` and `/` from being interpreted as model formula operators, this expression should be enclosed inside the function `I` (for 'identity'):

```
> ppm(bei ~ I(atan(grad) * 180/pi), data=bei.extra)
```

The function `I` returns its argument unchanged, and the rules for interpreting a formula ensure that the expression inside the parentheses is evaluated using the usual mathematical operations. The command above fits a loglinear model depending on the angle of inclination, expressed in degrees. It may be neater to write a separate R function which can then appear in the model formula:

```
> degrees <- function(x) { x * 180/pi }
> ppm(bei ~ degrees(atan(grad)), data=bei.extra)
```

Another strategy is to replace the first-order function in (9.7) by a quadratic function,

$$\lambda(u) = \exp(\beta_0 + \beta_1 S(u) + \beta_2 S(u)^2), \tag{9.9}$$

which results in a model which is *log-quadratic* as a function of slope. Here there are three parameters, β_0, β_1, and β_2, to be estimated. This model can be fitted by specifying the `slope` and `slope^2` terms in the model formula:

```
> ppm(bei ~ grad + I(grad^2), data=bei.extra)
Nonstationary Poisson process
Log intensity:  ~grad + I(grad^2)
Fitted trend coefficients:
(Intercept)        grad   I(grad^2)
     -5.987      18.745     -55.602
```

An advantage of this approach is that there is a formal mechanism for deciding (in statistical terms) whether the `grad^2` term is needed, namely a test of the null hypothesis $H_0 : \beta_2 = 0$. We discuss this in Section 9.4. Higher-order polynomials can be fitted in the same way.

The command `effectfun` can be used to visualise the fitted effect of a covariate: in this context `effectfun(fit, "grad")` would show the combined effect (linear and quadratic terms together) of the slope variable as shown in the right panel of Figure 9.3.

From a theoretical viewpoint, all the models fitted above are 'loglinear' models, of the general form (9.6). It is useful to distinguish between *'original covariates'* (e.g., `grad` in the current example) and *'canonical covariates'* (the variables Z_j which appear in the loglinear model (9.6)). Canonical covariates may be transformations and combinations of the original covariates which were provided in the dataset.

Table 9.1 lists some of the common terms appearing in model formulae.

9.3.2.2 Murchison gold data

We continue to illustrate models involving a single numerical covariate.

The Murchison geological survey data shown in Figure 9.4 record the spatial locations of gold deposits and associated geological features in the Murchison area of Western Australia. They are extracted from a regional survey (scale 1:500,000) of the Murchison area made by the Geological Survey of Western Australia [693]. The features recorded are the known locations of gold deposits (point pattern `gold`); the known or inferred locations of geological faults (line segment pattern `faults`); and the region of greenstone outcrop (window `greenstone`). The study region is contained in a 330×400 kilometre rectangle. At this scale, gold deposits are point-like, i.e. their spatial extent is negligible. Gold deposits are strongly associated with greenstone bedrock and with faults,

SAMPLE

TERM	MEANING
`(age > 40)`	logical covariate: `TRUE` if age exceeds 40
`cut(age, 3)`	factor covariate: age classified into 3 bands
`I(age^2)`	numerical covariate: age squared
`polynom(age, 3)`	cubic polynomial function of age
`bs(age, 5)`	smooth function of age
`1`	constant term (intercept)
`-1`	remove intercept term
`+0`	remove intercept term

Table 9.1. *Common terms in model formulae, assuming* `age` *is a numerical covariate. (Note:* `bs` *requires the* `splines` *package.)*

but the geology is of course three-dimensional, and the survey data are a two-dimensional projection. The survey may not have detected all existing faults, because they are usually not observed directly; they are observed in magnetic field surveys or geologically inferred from discontinuities in the rock sequence.

These data were analysed in [269, 134]; see also [298, 386]. The main aim is to predict the intensity of the point pattern of gold deposits from the more easily observable fault pattern. To be precise, the aim is to use the observed pattern of faults to specify zones of high 'prospectivity' to be explored for gold. See [116, 139] for general background.

First we rescale the data to kilometres:

```
> mur <- solapply(murchison, rescale, s=1000, unitname="km")
```

The full dataset is shown in Figure 9.4, generated using `plot(as.layered(mur))`, and a detail in the upper right corner is shown in Figure 9.5.

A common model for such data is a Poisson process with intensity which is a loglinear function of distance to the nearest fault,

$$\lambda(u) = \exp(\beta_0 + \beta_1 d(u)), \tag{9.10}$$

where β_0, β_1 are parameters and $d(u)$ is the distance from location u to the nearest geological fault. This is equivalent to the model customarily fitted in GIS software by logistic regression of presence-absence indicators on distance to nearest fault.

We compute the distance covariate $d(u)$ using `distfun` (Section 6.6.4) which has the advantage that distances will be computed exactly by analytic geometry. A less accurate alternative would be to compute the distance values at a grid of pixel locations using `distmap`.

```
> dfault <- with(mur,distfun(faults))
```

Next we fit the loglinear model (9.10):

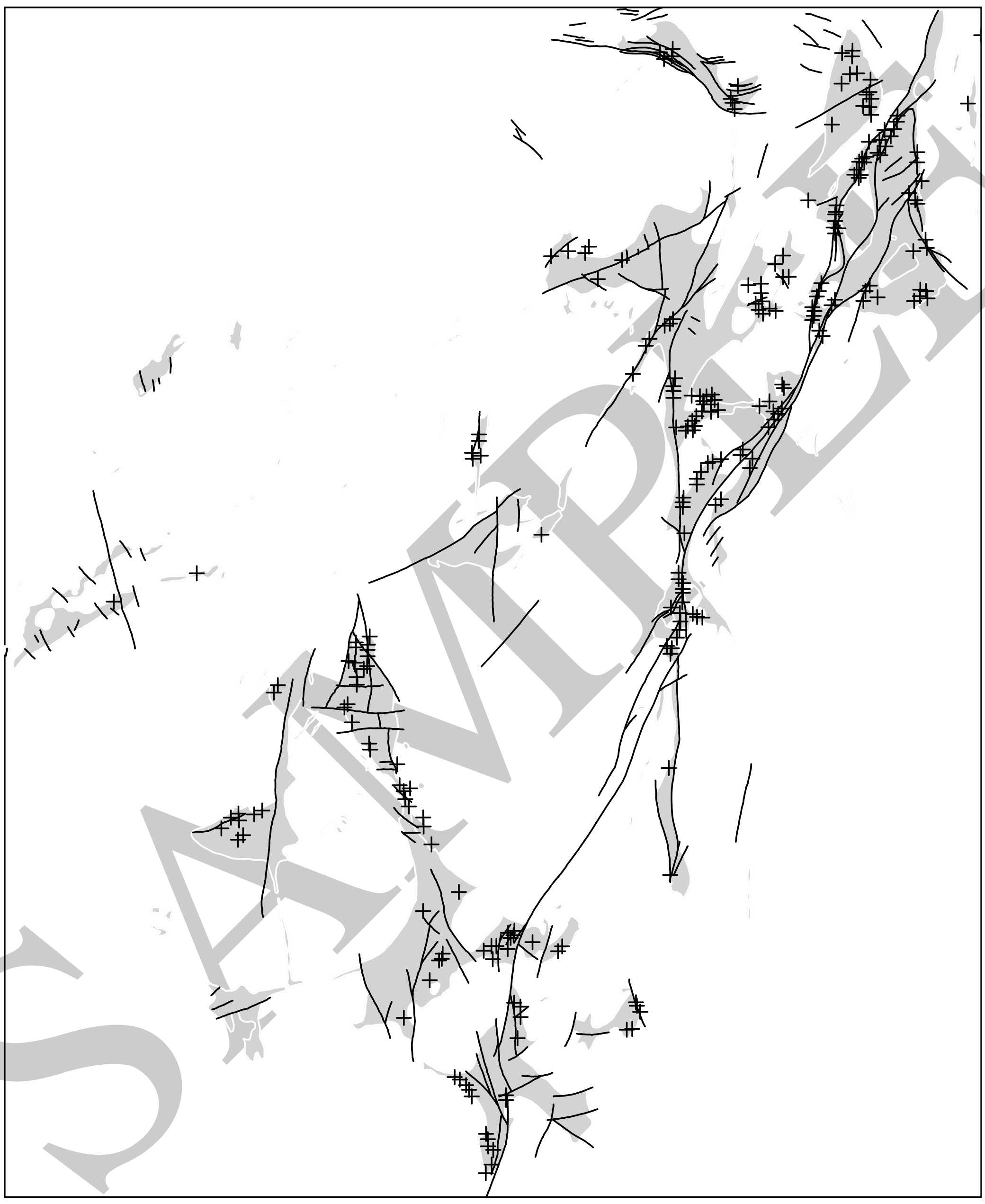
```
> fit <- ppm(gold ~ dfault, data=mur)
> fit
Nonstationary Poisson process
Log intensity:  ~dfault
Fitted trend coefficients:
(Intercept)      dfault
    -4.3413     -0.2608
```

The fitted model is

$$\lambda(u) = \exp(-4.341 - 0.2608\, d(u)). \tag{9.11}$$

That is, the estimated intensity of gold deposits in the immediate vicinity of a geological fault

SAMPLE



SAMPLE

Figure 9.4. *Murchison geological survey data. Gold deposits (+), geological faults (—), and greenstone outcrop (grey shading) in a survey region about 200 by 300 km across.*

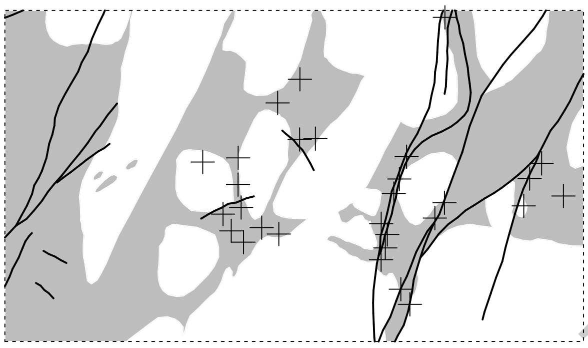


Figure 9.5. *Detail of Murchison geological survey data in a 70 by 40 km region near the abandoned town of Reedy. Gold deposits (+), faults (solid lines), and greenstone outcrop (grey shading).*

is about $\exp(-4.341) = 0.01302$ deposits per square kilometre or 1.302 deposits per 100 square kilometres, and decreases by a *factor* of $\exp(-0.2608) = 0.7704$ for every additional kilometre away from a fault. At a distance of 10 kilometres, the intensity has fallen by a factor of $\exp(10 \times (-0.2608)) = 0.07368$ to $\exp(-4.341 + 10 \times (-0.2608)) = 0.0009596$ deposits per square kilometre or 0.09596 deposits per 100 square kilometres. Figure 9.6 shows the effect of the distance covariate on the intensity function.

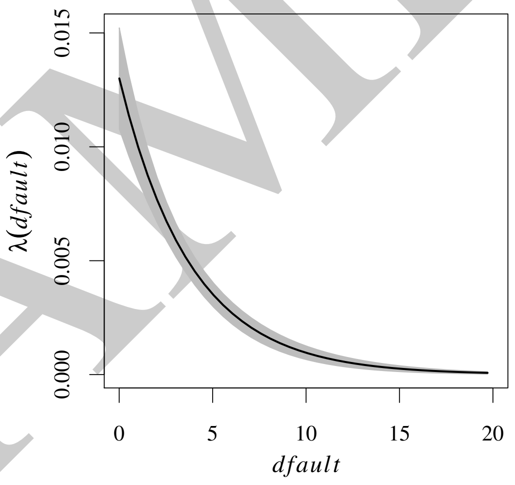


Figure 9.6. *Fitted intensity of Murchison gold deposits as a function of distance to the nearest fault, assuming it is a loglinear function of distance (equivalent to logistic regression). Solid line: maximum likelihood estimate. Shading: pointwise 95% confidence interval. Generated by* `plot(effectfun(fit, "dfault", se.fit=TRUE))`.

We caution that, by fitting the model with formula `gold ~ dfault` using `ppm` or equivalently using logistic regression, we have not fitted a highly flexible model in which the abundance of deposits depends, in some unspecified way, on the distance to the nearest fault. Rather, the very specific relationship (9.10) has been fitted. To fit more flexible models, we need to modify the model formula, as discussed below.

In prospectivity analysis it may or may not be desirable to fit any explicit relationship between

SAMPLE

deposit abundance and covariates such as distance to the nearest fault. Often the objective is simply to select a distance threshold, so as to delimit the area which is considered highly prospective (high predicted intensity) for the mineral. The ROC curve (Section 6.7.3) is more relevant to this exercise. However, if credible models can be fitted, they contain much more valuable predictive information.

9.3.3 Models with a logical covariate

A logical covariate takes the values `TRUE` and `FALSE`. Logical covariates can arise in many ways, as we shall see in this chapter.

The Murchison data include the polygonal boundary of greenstone outcrop, given as a window called `greenstone`. Setting aside the information about geological faults for a moment, we could fit a simple Poisson model in which the intensity of gold deposits is constant inside the greenstone, and constant outside the greenstone, but with different intensity values inside and outside.

```
> ppm(gold ~ greenstone, data=mur)
Nonstationary Poisson process

Log intensity:  ~greenstone

Fitted trend coefficients:
   (Intercept) greenstoneTRUE
        -8.103          3.980

               Estimate   S.E. CI95.lo CI95.hi Ztest   Zval
(Intercept)      -8.103 0.1667  -8.430  -7.777   *** -48.62
greenstoneTRUE    3.980 0.1798   3.628   4.333   ***  22.13
```

The function `ppm` recognises that `greenstone` is a spatial window, and interprets it as a covariate with *logical* values, equal to `TRUE` inside the greenstone outcrop and `FALSE` outside.

Since the model formula does not explicitly exclude a constant term ('intercept'), this term is assumed to be present. The model fitted is

$$\lambda(u) = \exp(\alpha + \beta G(u)) \tag{9.12}$$

where G is the indicator covariate for the greenstone, taking the value $G(u) = 1$ if the location u is inside the greenstone outcrop, and $G(u) = 0$ if it is outside. The printed output shows the estimates for the parameters α (labelled `Intercept`) and β (labelled `greenstoneTRUE`) as -8.103 and 3.98 respectively. The estimated intensities are $\lambda_{\text{out}} = \exp(-8.103) = 0.0003026$ outside the greenstone outcrop and $\lambda_{\text{in}} = \exp(-8.103 + 3.98) = 0.0162$ inside. The last few lines of output give the estimates of α and β together with estimates of standard error, 95% confidence intervals, and a report that the coefficient β is significantly different from 0.

Slightly different output is obtained with the command

```
> ppm(gold ~ greenstone - 1, data=mur)
Nonstationary Poisson process

Log intensity:  ~greenstone - 1

Fitted trend coefficients:
greenstoneFALSE  greenstoneTRUE
         -8.103          -4.123

                Estimate    S.E. CI95.lo CI95.hi Ztest   Zval
greenstoneFALSE   -8.103 0.16667  -8.430  -7.777   *** -48.62
greenstoneTRUE    -4.123 0.06757  -4.255  -3.990   *** -61.01
```

SAMPLE

The model formula now includes `-1` forbidding the use of a constant term ('intercept'). The model fitted here is

$$\lambda(u) = \exp(\gamma G_{\text{out}}(u) + \delta G_{\text{in}}(u)) \tag{9.13}$$

where $G_{\text{in}}(u) = G(u)$ is the indicator of the greenstone as before, and $G_{\text{out}}(u) = 1 - G(u)$ is the indicator of the complement of the greenstone. The estimates of γ and δ are -8.103 and -4.123 respectively. The estimates of the intensities are $\lambda_{\text{out}} = \exp(-8.103) = 0.0003026$ outside the greenstone outcrop and $\lambda_{\text{in}} = \exp(-4.123) = 0.0162$ inside. The two fitted models agree, but are parametrised differently. The last few lines of output give standard errors and confidence intervals for the parameters γ and δ, and indicate that these parameters are significantly different *from zero*. The first parameterisation (i.e. with an intercept included) is more useful for deciding whether the greenstone has an effect on the intensity of gold deposits, while the second parameterisation is more useful for directly reading off the estimated intensities.

The foregoing discussion depends upon the default 'treatment contrasts' being used, as explained below.

9.3.4 Models with a factor covariate

A spatial covariate with *categorical* values could arise from a map which divides the spatial domain into regions of different type, according to criteria such as rock type, vegetation cover, land use, administrative region, socioeconomic level, government building zone type, or anatomical subdivision of tissue. The map itself is a tessellation of the spatial domain. The associated spatial covariate $J(u)$ tells us which type or category applies to each given location u, so that J is a spatial covariate with categorical values.

In some cases the original data will be in 'vector graphics' form, giving the spatial coordinates of the boundaries of the regions of each type. To preserve accuracy, vector data should not be discretised. The boundary polygons should be converted to a tessellation (object of class `"tess"`) which can then be passed directly as a covariate to `ppm`.

If the categorical data are provided as a pixel image, it is important to ensure that the pixel values are recognised as categorical values. Printing or summarising the image in question should indicate that it is `factor`-valued.

A numerical covariate Z can also be converted into a factor by dividing the range of values into a few bands, and treating each band of values as a category. This is often a useful exploratory tool, as we saw in Section 6.6.2.

9.3.4.1 Gorilla nest data

The `gorillas` dataset comes from a study [272] in the Kagwene Gorilla Sanctuary, Cameroon, by the Wildlife Conservation Society Takamanda-Mone Landscape Project (WCS-TMLP). A detailed description and analysis of the data is reported in [273]. The data were kindly contributed to `spatstat` by Dr Funwi-Gabga Neba. The collaboration of Prof Jorge Mateu is gratefully acknowledged.

The data include the marked point pattern of gorilla nest sites `gorillas`, and auxiliary data in a list `gorillas.extra` containing seven pixel images of spatial covariates. We rescale the data to kilometres:

```
> gor <- rescale(gorillas, 1000, unitname="km")
> gor <- unmark(gor)
> gex <- lapply(gorillas.extra, rescale,
                s=1000, unitname="km")
```

Figure 9.7 shows the spatial locations of the (unmarked) gorilla nests, and the covariate `vegetation` which reports the vegetation or cover type.

SAMPLE

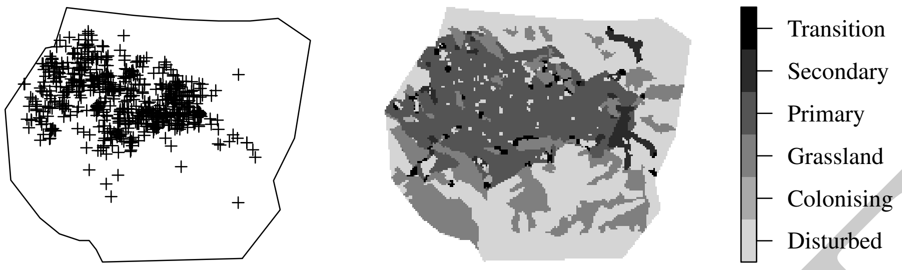


Figure 9.7. *Gorillas data.* Left: *gorilla nest locations.* Right: *vegetation type.*

To save space on the printed page, we will abbreviate the names of the covariates, and the names of the levels of factors. We could use `substr` to extract the first few characters of each name, or `abbreviate` to convert the names to shorthand.

```
> names(gex)
[1] "aspect"     "elevation"  "heat"       "slopeangle" "slopetype"
[6] "vegetation" "waterdist"
> shorten <- function(x) substr(x, 1, 4)
> names(gex) <- shorten(names(gex))
> names(gex)
[1] "aspe" "elev" "heat" "slop" "slop" "vege" "wate"
> names(gex)[4:5] <- c("sang", "styp")
> names(gex)
[1] "aspe" "elev" "heat" "sang" "styp" "vege" "wate"
> isfactor <- !sapply(lapply(gex, levels), is.null)
> for(i in which(isfactor))
    levels(gex[[i]]) <- shorten(levels(gex[[i]]))
> levels(gex$vege)
[1] "Dist" "Colo" "Gras" "Prim" "Seco" "Tran"
```

The following command fits a simple Poisson model in which the intensity is constant in each vegetation type, but may depend on vegetation type:

```
> ppm(gor ~ vege, data=gex)
Nonstationary Poisson process

Log intensity:  ~vege

Fitted trend coefficients:
(Intercept)    vegeColo    vegeGras    vegePrim    vegeSeco    vegeTran
     2.3238      2.0817     -0.7721      2.1148      1.1341      1.6151

            Estimate   S.E. CI95.lo CI95.hi Ztest   Zval
(Intercept)   2.3238 0.1060  2.1161  2.5316   *** 21.923
vegeColo      2.0817 0.5870  0.9312  3.2322   ***  3.546
vegeGras     -0.7721 0.2475 -1.2571 -0.2871    ** -3.120
vegePrim      2.1148 0.1151  1.8892  2.3403   *** 18.373
```

SAMPLE

```
vegeSeco      1.1341 0.2426  0.6586  1.6096   ***  4.675
vegeTran      1.6151 0.2647  1.0964  2.1339   ***  6.102
```

For models which involve a factor covariate, the interpretation of the fitted coefficients depends on which *contrasts* are in force, as specified by `getOption("contrasts")`. We explain more about this below. By default the 'treatment contrasts' are assumed. This means that — if an intercept term is present — the coefficient associated with the first level of the factor is taken to be zero, and the coefficients associated with the other levels are effectively differences relative to the first level. In the model above, the fitted log intensities for each vegetation type are calculated as follows:

TYPE	RELEVANT COEFFICIENTS	LOG INTENSITY
Disturbed	`(Intercept)`	2.324
Colonising	`(Intercept) + vegeColo`	$2.324+2.082$
Grassland	`(Intercept) + vegeGras`	$2.324-0.7721$
Primary	`(Intercept) + vegePrim`	$2.324+2.115$
Secondary	`(Intercept) + vegeSeco`	$2.324+1.134$
Transition	`(Intercept) + vegeTran`	$2.324+1.615$

Rather than relying on such interpretations, it would be prudent to use the command `predict` to compute predicted values of the model, as explained in Section 9.4.1 below.

Another way to fit the same model is to remove the intercept, so that a single coefficient will be associated with each level of the factor.

```
> fitveg <- ppm(gor ~ vege - 1, data=gex)
> fitveg
Nonstationary Poisson process

Log intensity:  ~vege - 1

Fitted trend coefficients:
vegeDist vegeColo vegeGras vegePrim vegeSeco vegeTran
   2.324    4.406    1.552    4.439    3.458    3.939

         Estimate    S.E. CI95.lo CI95.hi Ztest   Zval
vegeDist    2.324 0.10600   2.116   2.532   *** 21.923
vegeColo    4.406 0.57735   3.274   5.537   ***  7.631
vegeGras    1.552 0.22361   1.114   1.990   ***  6.940
vegePrim    4.439 0.04486   4.351   4.527   *** 98.952
vegeSeco    3.458 0.21822   3.030   3.886   *** 15.846
vegeTran    3.939 0.24254   3.464   4.414   *** 16.241
> exp(coef(fitveg))
vegeDist vegeColo vegeGras vegePrim vegeSeco vegeTran
   10.21    81.90     4.72    84.66    31.75    51.37
```

For this simple model, where intensity is constant inside each region defined by vegetation type, an equivalent way to fit the same model is to estimate the intensities using quadrat counts. To check this, we convert the pixel image to a tessellation and apply quadrat counting:

```
> vt <- tess(image=gex$vege)
> intensity(quadratcount(gor, tess=vt))
tile
  Dist   Colo   Gras   Prim   Seco   Tran
10.201 69.155  4.781 84.012 32.651 50.922
```

The discrepancies are due to discretisation effects: refitting the model by calling `ppm` with `nd=512` gives results consistent with the quadrat-counting estimates.

SAMPLE

9.3.4.2 Factor effects and contrasts

To understand fully the results obtained for the `gorillas` data above, we need to know more about the handling of factors by the model-fitting code in R. If `f` is a factor then `X ~ f` specifies a point process model in which the intensity depends on the level of `f`. The model is

$$\lambda(u) = \exp(\mu + \alpha_{J(u)}) \tag{9.14}$$

where $J(u)$ is the level of the factor `f` at the location u. The parameters of this model are the *intercept* μ and the *effects* $\alpha_1, \ldots, \alpha_L$ of the different factor levels. Here we are assuming the factor has L different levels numbered 1 to L. For the ith level of the factor, the corresponding value of the intensity is $e^{\mu+\alpha_i}$.

While the model (9.14) is conceptually useful, it has the practical drawback that it is *overparameterised*. If there are L different levels of the factor, then there are $L+1$ parameters to be estimated from data, but only L different values of the intensity on the right-hand side of (9.14). In order to fit the model we need to reduce the number of parameters by 1.

One option is to remove the intercept parameter μ. The model is then

$$\lambda(u) = \exp(\alpha_{J(u)}) \tag{9.15}$$

so that the intensity value for the ith level of the factor is e^{α_i}. There are now only L parameters, the effects $\alpha_1, \ldots, \alpha_L$ of the factor levels. The model `ppm(gor ~ vege - 1)` has this form, so the fitted coefficients are the fitted estimates of the log intensity for each type of vegetation.

The default behaviour in R is to retain the intercept and instead to constrain the effect of the first level to be zero, $\alpha_1 = 0$. This convention is called the *'treatment contrast'*; it makes sense when the first level of the factor is a baseline or control, while the other levels are different possible 'treatments' that could be applied to a subject in an experiment. The model (9.14) then states that the intensity value for the first level is e^{μ}, while the intensity value for another level i is $e^{\mu+\alpha_i}$, so the coefficient α_i is the *difference* in effect between level i and the first level, on a log scale. The table on page 316 shows this calculation. An advantage of the treatment contrast is that we are often interested in whether there is any difference between the intensities associated with different factor levels: this can be assessed by testing the hypothesis $\alpha_i = 0$ for each $i > 1$.

In its linear modelling procedures, the commercial statistics package SAS® constrains the last coefficient α_L to be 0. Another commonly used constraint is $\alpha_1 + \ldots + \alpha_L = 0$. In R one can choose the constraint that is used by specifying the *contrast* or linear expression that will be set to zero. This is done using the `options` function, with a call of the form:

```
    options(contrasts=c(arg1,arg2))
```

where `arg1`, `arg2` are strings which specify the contrasts to be used with 'unordered' and 'ordered' factors, respectively. All the factors discussed in this book are 'unordered'. Note that if you do set the contrasts, you have to specify both `arg1` and `arg2` even though you only care about `arg1`.

The possible values for either argument are the character strings `"contr.sum"`, `"contr.poly"`, `"contr.helmert"`, `"contr.treatment"` and `"contr.SAS"`. The default for `arg1` is the 'treatment contrasts' `"contr.treatment"` which impose the constraint $\alpha_1 = 0$ in the context of a single categorical predictor. We use this default throughout the book. If you are unsure what contrasts are currently in force, type `getOption("contrasts")`.

It is important to remember that models obtained by employing different constraints are *precisely equivalent*, although the interpretation of the parameter estimates is different.

The overparameterised form (9.14) of the model allows one to conveniently specify models that are of genuine interest and that would be very hard to specify otherwise. We elaborate on this when we come to models with two (or more) categorical predictors.

The model-fitting software in R converts logical variables to categorical variables, so that the

SAMPLE

variable `greenstone` in the models above is converted to a `factor` with levels `TRUE` and `FALSE`. The intercept term in the model is associated with the first level of the factor, which is `FALSE`, since this comes alphabetically before `TRUE`. Happily this convention gives a sensible result here.

9.3.5 Additive models

We now start looking at models which involve more than one 'original' covariate. The *Beilschmiedia* data (see Sections 9.3.1 and 9.3.2.1) include two (numerical) covariates, the terrain elevation and terrain slope. The simplest loglinear model depending on both covariates is the additive model

$$\lambda(u) = \exp(\beta_0 + \beta_1 E(u) + \beta_2 S(u)) \tag{9.16}$$

where $\beta_0, \beta_1, \beta_2$ are parameters to be estimated, $E(u)$ is the terrain elevation in metres at location u, and $S(u)$ is the terrain slope (a dimensionless gradient). Additive models are specified by joining the relevant model terms together using `+`:

```
> fitadd <- ppm(bei ~ elev + grad, data=bei.extra)
> fitadd
Nonstationary Poisson process
Log intensity:  ~elev + grad
Fitted trend coefficients:
(Intercept)        elev        grad
   -8.55862     0.02141     5.84104
```

The interpretation of the fitted model is straightforward, since both covariates are continuous numerical quantities. On a level parcel of terrain (slope zero) at sea level (elevation zero) the intensity of *Beilschmiedia* trees would be $e^{-8.559} = 0.0001918$ trees per square metre. For each additional metre of terrain elevation, the intensity of increases by a factor $e^{1\times 0.02141} = 1.022$ or about 2.2 percent. For each additional increase of (say) 0.1 units in slope, the intensity increases by a factor $e^{0.1\times 5.841} = 1.793$. These two effects are *additive* on the scale of the linear predictor: elevation increases by 1 metre and slope increases by 0.1 units, then the *log* intensity increases by the elevation effect 1×0.02141 **plus** the slope effect 0.1×5.841.

To assess the relative 'importance' of the slope and elevation variables in the model, we need to account for the range of values of each variable. The terrain slope varies from zero to `max(grad)=` 0.3285 so the effect of slope varies by a factor of $\exp(5.841 \times 0.3285) = 6.813$ between flattest and steepest slopes. Terrain elevation ranges between 119.8 and 159.5 metres, so the effect of elevation varies by a factor of $\exp(0.02141(159.5 - 119.8)) = 2.34$ between lowest and highest elevations. Hence the slope effect is more 'important' in magnitude.

For the Murchison data we have two covariates, namely `dfault` (a continuous numerical variable) and `greenstone` (a logical covariate). The additive model with these two covariates is

$$\lambda(u) = \exp(\beta_0 + \beta_1 d(u) + \beta_2 G(u)) \tag{9.17}$$

where $\beta_0, \beta_1, \beta_2$ are parameters to be estimated, $d(u)$ is the distance to nearest fault, and $G(u)$ is the indicator which equals 1 inside the greenstone outcrop.

```
> ppm(gold ~ dfault + greenstone,data=mur)
Nonstationary Poisson process
Log intensity:  ~dfault + greenstone
Fitted trend coefficients:
   (Intercept)         dfault greenstoneTRUE
       -6.6171        -0.1038         2.7540
```

SAMPLE

The fitted model states that the intensity of gold deposits declines by a factor $e^{-0.1038} = 0.9014$ for every additional kilometre of distance from the nearest fault, and for a given distance, the intensity is $e^{2.754} = 15.71$ times higher inside the greenstone outcrop than it is outside the greenstone.

Similarly we could build an additive model with two factor covariates, such as

```
> ppm(gor ~ vege + styp, data=gex)
```

which states that the log intensity of gorilla nesting sites is the sum of an effect due to vegetation/cover type and an effect due to slope type. Equivalently the *intensity* is a vegetation effect *multiplied by* a slope effect.

Note that the model `bei ~ grad + I(grad^2)` discussed in Section 9.3.2 is also an additive model. It adds the effects of the two covariates `grad` and `I(grad^2)`. Although the two covariates are closely related, nevertheless this is formally an additive model.

Finally, note that adding the same covariate twice has no effect in a model formula: `grad + grad` is equivalent to `grad`.

9.3.6 Modelling spatial trend using Cartesian coordinates

The Cartesian coordinates x and y can also serve as spatial covariates. They are particularly useful for investigating spatial trend when there are no relevant covariate data, or perhaps when it is suspected that the intensity is not only dependent on the available covariates. A wide class of models can be built up by combining the Cartesian coordinates in convenient ways.

We illustrate this idea using the full Japanese Pines data of Ogata and Numata,

```
> jpines <- residualspaper[["Fig1"]]
```

A plot of these data (Figure 1.3 on page 4) suggests that they exhibit spatial inhomogeneity. No auxiliary covariate data are available, so we resort to the Cartesian coordinates. For brevity we shall write the intensity function for such models in the form $\lambda((x,y))$ rather than the more long-winded form '$\lambda(u)$ where u is the point with coordinates (x,y)'. For instance an inhomogeneous Poisson model with an intensity that is first order loglinear in the Cartesian coordinates will be written as $\lambda_{\theta}((x,y)) = \exp(\theta_0 + \theta_1 x + \theta_2 y)$. To fit such a model to the Japanese Pines data simply type

```
> ppm(jpines ~ x + y)
Nonstationary Poisson process
Log intensity:  ~x + y
Fitted trend coefficients:
(Intercept)           x           y
   0.591840    0.014329    0.009644
```

Here the symbols `x` and `y` are reserved variable names that always indicate the Cartesian coordinates. The fitted intensity function is

$$\lambda_{\theta}((x,y)) = \exp(0.5918 + 0.01433x + 0.009644y).$$

To fit an inhomogeneous Poisson model with an intensity that is log-quadratic in the Cartesian coordinates, i.e. such that $\log \lambda_{\theta}((x,y))$ is a quadratic in x and y:

```
> ppm(jpines ~ polynom(x,y,2))
Nonstationary Poisson process
Log intensity:  ~x + y + I(x^2) + I(x * y) + I(y^2)
Fitted trend coefficients:
(Intercept)           x           y      I(x^2)    I(x * y)      I(y^2)
   0.130330    0.461497   -0.200945   -0.044738    0.001058    0.020399
```

SAMPLE

Notice that the expression `polynom(x,y,2)` has been syntactically expanded into

```
x + y + I(x^2) + I(x*y) + I(y^2)
```

This is a special `spatstat` trick and applies only to the model formulae passed to `ppm` or `kppm`.

In the same vein we could fit a log-cubic polynomial `polynom(x,y,3)` and so on. An alternative to the full polynomial of order 3, which has 10 coefficients, is the *harmonic* polynomial `harmonic(x,y,3)` which has only 7 coefficients.

Any transformation of the Cartesian coordinates can also be used as a covariate. For example, to fit a model with constant but unequal intensities on each side of the vertical line $x = 0.5$, we simply use the expression `x < 0.5` to make a logical covariate:

```
> ppm(jpines ~ (x < 0.5))
Nonstationary Poisson process
Log intensity:  ~(x < 0.5)
Fitted trend coefficients:
(Intercept) x < 0.5TRUE
     0.7668     -1.8931
```

9.3.7 Models with offsets

An *offset* is a term in the linear predictor which does not involve any parameters of the model. Offsets are useful when the effect of one variable is already known, or when we want to fit the model relative to a known baseline.

For instance, in some cases it is appropriate to fit an inhomogeneous Poisson model with intensity that is *proportional* to the covariate,

$$\lambda(u) = \kappa Z(u) \tag{9.18}$$

where Z is the covariate and κ is the parameter to be estimated. We called this a 'baseline' model in Section 9.2.3. Taking logarithms, this model is equivalent to

$$\log\lambda(u) = \log\kappa + \log Z(u) = \theta + \log Z(u) \tag{9.19}$$

where $\theta = \log\kappa$ is the only parameter of the model. Note that there is no coefficient in front of the term $\log Z(u)$ in (9.19), so $\log Z(u)$ is an *offset*.

An important example of a baseline covariate $Z(u)$ is the spatially varying density of human population. The spatial point pattern of cases of a rare disease could reasonably be expected to follow a Poisson point process with intensity (9.18), where κ is the (constant) disease risk per head of population.

The Chorley-Ribble data (Figure 1.12 on page 9) give the locations of cases of the rare cancer of the larynx, and a sample of cases of the much more common lung cancer. The smoothed intensity of lung cancer cases can serve as a surrogate for the spatially varying density of the susceptible population.

```
>   lung <- split(chorley)$lung
>   larynx <- split(chorley)$larynx
>   smo <- density(lung, sigma=0.15, eps=0.1, positive=TRUE)
```

Here we have adopted the smoothing bandwidth $\sigma = 0.15km$ chosen by Diggle [224]; the different rules for automatic bandwidth selection give a wide range of results between 0.07 and 2 km. The pixels are `eps=0.1`*km* wide. The argument `positive=TRUE` ensures that negative or zero pixel values (due to numerical error) are replaced by a small positive number. To avoid numerical instability we shall raise the threshold slightly:

SAMPLE

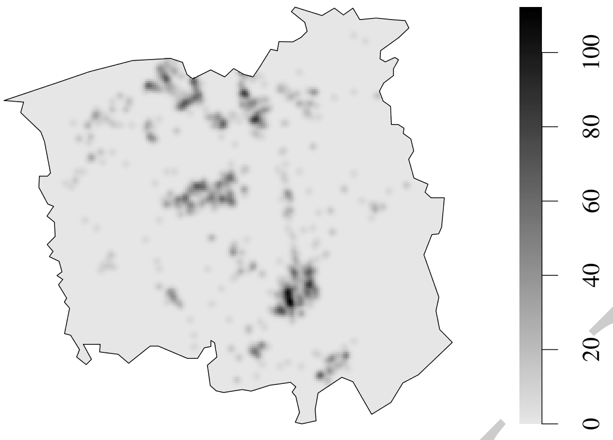


Figure 9.8. *Smooth intensity estimate for lung cancer cases in the Chorley-Ribble data.*

```
> smo <- eval.im(pmax(smo, 1e-10))
```

The resulting pixel image `smo`, serving as the baseline for our model, is shown in Figure 9.8.

In a model formula, we indicate an offset term by enclosing it in the function `offset`. Accordingly we fit the constant risk model (9.18) by

```
> ppm(larynx ~ offset(log(smo)))
Nonstationary Poisson process
Log intensity:  ~offset(log(smo))
Fitted trend coefficient:  (Intercept) = -2.939
```

The fitted coefficient `(Intercept)` is the constant $\log \kappa$ appearing in (9.19), so converting back to the form (9.18), the fitted model is

$$\lambda(u) = e^{-2.939}\, Z(u) = 0.05292\, Z(u)$$

where in this case $Z(u)$ is the smoothed intensity of lung cancer cases.

In this example, note that the fitted parameter κ is not the estimated risk of laryngeal cancer per head of population, because the `lung` data are a subsample from the cancer registry, and lung cancer cases are a subset of the susceptible population. The best way to estimate the risk of laryngeal cancer assuming constant risk is to divide the total number of laryngeal cancer cases by an estimate of the total susceptible population, since the constant risk model does not involve spatial information. The model fitted above is useful mainly for comparison against alternative models where the risk of laryngeal cancer is spatially varying.

Spatial transformations, such as geographic projections, also introduce offset terms. Suppose that a point process $\mathbf{X}$ has intensity function $\lambda_{\mathbf{X}}(u) = \exp(\beta^{\mathsf{T}} \mathbf{Z}(u))$. We change the coordinate system so that the points x_i are mapped to new coordinate positions $y_i = f(x_i)$. Changes of coordinates were discussed in Section 6.5.3: the transformed point process $\mathbf{Y} = f(\mathbf{X})$ has intensity function given by equation (6.17). The new model is

$$\lambda_{\mathbf{Y}}(u) = \exp(\log J(u) + \beta^{\mathsf{T}} \mathbf{Z}(f^{-1}(u))) \tag{9.20}$$

which includes the offset term $\log J(u)$, the log of the Jacobian of the change of coordinates.

In a study of the spatial pattern of tree deaths in a water catchment [684, 145] the spatial pattern of *live* trees, observed at a certain date, was smoothed to give a covariate image $A(u)$ representing the

SAMPLE

spatially varying density of the forest. Other covariates $Z_1(u),\dots,Z_p(u)$ were available, including terrain elevation, depth to water table, and seasonal recharge flow of groundwater. Models of the form

$$\lambda(u) = A(u)\exp(\theta_0 + \theta_1 Z_1(u) + \ldots + \theta_p Z_p(u)) \tag{9.21}$$

were fitted, using a model formula of the type `~ offset(log(A)) + Z1 + ... + Zp`. The term in the exponential represents the spatially varying tree death risk.

When fitting a baseline model it is absolutely crucial to express the baseline term in the form `offset(log(baseline))`. If the `offset` is omitted, we get a completely different model, as discussed below.

9.3.8 Power law models

It is sometimes appropriate to fit a *power law* relationship

$$\lambda(u) = \alpha Z(u)^k \tag{9.22}$$

where the exponent k is not known, and must be estimated from data, along with the coefficient α. Taking logarithms, the power law (9.22) is equivalent to

$$\log \lambda(u) = \beta_0 + \beta_1 \log Z(u) \tag{9.23}$$

where $\beta_0 = \log \alpha$ and $\beta_1 = k$. This is a loglinear model with covariate $\log Z$.

In the *Beilschmiedia* data, suppose we believe that the forest density obeys a power law as a function of terrain slope. This can be fitted easily:

```
> ppm(bei ~ log(grad), data=bei.extra)
Nonstationary Poisson process
Log intensity:  ~log(grad)
Fitted trend coefficients:
(Intercept)   log(grad)
    -3.4797      0.5549
```

Note that there is no `offset` here. The printed output describes a model in which the intensity is proportional to the 0.55th power of the terrain slope, or roughly the *square root* of the slope.

Care needs to be taken with power models if the covariate data may include zero values. If the exponent $k = \beta_1$ is positive, then the power law (9.22) implies that, in places where the covariate value $Z(u)$ is zero, the intensity $\lambda(u)$ is also zero, so there is zero probability of observing a random point at such places. If a data point *is* observed at a place where the covariate is zero, this model is invalidated (it has likelihood zero). In the examples above, if there had been a *Beilschmiedia* tree growing on a perfectly level patch of land (i.e. where the terrain slope is zero) then the data would have been inconsistent with the model (9.18) and (9.22). In some cases this may be detected by the `ppm` function, but in many cases it will not be detected until the low-level code throws a rather undignified error. We can reproduce this scenario by artificially assigning a zero value to a pixel where there is a data point:

```
> G <- bei.extra[["grad"]]
> G[bei[42]] <- 0
> ppm(bei ~ log(G))
Error in glm.fit(x = structure(c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  :
NA/NaN/Inf in 'x'
```

If the fitted exponent $k = \beta_1$ is negative, then the power law implies that, in places where $Z(u) = 0$, the intensity is *infinite*. This means it is possible that the fitted model may be **improper**. In

SAMPLE

Section 9.2.1 we noted that, in order for a Poisson process to be well defined, the integral of the intensity function over the window must be finite. Intensity functions of the form (9.22) may not be integrable if the exponent β_1 is negative.

For example with the Murchison data, the covariate $d(u)$, distance to nearest fault, takes zero values along the faults themselves. If we fit a power model,

```
> ppm(gold ~ log(dfault), data=mur)
Nonstationary Poisson process
Log intensity:  ~log(dfault)
Fitted trend coefficients:
(Intercept) log(dfault)
    -5.0576     -0.7188
```

the fitted model says that the intensity is the negative 0.72th power of $d(u)$. This function is not integrable so the model is improper, that is, it is not well defined.

Improper models can occur in other contexts where some values of the covariate are infinite (in this case $Z = 0$ gives $\log Z = -\infty$). An improper model will lead to difficulties with simulation code and various other algorithms.

9.3.9 Models with interaction between covariates

A model which is not additive is said to have 'interaction'. A statistical model involving two covariates `A` and `B` is additive if the linear predictor is the sum of a term depending only on `A` plus another term depending only on `B`. Otherwise, the model is said to exhibit *interaction* between the covariates `A` and `B`. Interaction means that the effect of a change in `A` depends on the current value of `B`, and *vice versa*.

Be warned that there is another use of the word 'interaction' in spatial statistics. A point process which is not Poisson is said to exhibit 'interaction' between the *points of the process*. This is a distinct and completely unrelated concept for which the same word is used. We will call this 'interpoint interaction' when the distinction needs to be emphasised.

In the present context we are talking about *interaction between covariates* in a model for the intensity of the point process. In a model formula, the expression `A:B` represents a particular model term called 'the' interaction between the covariates `A` and `B`. This is effectively a new covariate depending on both `A` and `B`. Its definition depends, as usual, on the type of variables involved: we explain further below.

Normally, a model that includes an interaction `A:B` should also include the 'main effects' `A` and `B`. Usually it is *possible* to write a model which includes only an interaction term but this makes the results difficult to interpret and reduces their usefulness. How to interpret them, and whether they actually make any sense at all, depends on circumstances.

To include interaction between `A` and `B` in a model we would thus generally write the formula in the form `X ~ A + B + A:B`. The expression `A + B + A:B` can be abbreviated to `A * B` which is read as '`A` cross `B`' (and the operator `*` is referred to as the crossing operator). Examples are discussed below.

Table 9.2 lists operators that can be applied to terms in a model formula.

9.3.9.1 Interaction between two numerical covariates

If `A` and `B` are both numerical covariates, the expression `A:B` is interpreted as simply their numerical product, `A` times `B`. This makes sense: there is 'interaction' between `A` and `B` if the rate of change of the response with respect to `A` depends upon the value of `B` (and vice versa). This will be the case if the response depends on the product of `A` and `B`.

Thus, if `A` and `B` are numerical covariates, `A:B` is equivalent to `I(A * B)` and `A * B` is equivalent to `X ~ A + B + I(A*B)`. Some analysts prefer to use the expanded forms to prevent confusion.

SAMPLE

OPERATOR	EXAMPLE	MEANING
`+`	`+A`	include this term
`-`	`-A`	remove this term
`:`	`A:B`	include interaction between `A` and `B`
`*`	`A*B`	include these terms and their interaction
`^`	`(A+B)^3`	include these terms and interactions up to order 3
`/`	`A/B`	nesting: `B` nested within `A`
`%in%`	`A %in% B`	interaction between `A` and `B`
`|`	`A|B`	conditioning: `A` given `B`

Table 9.2. *Operators in model formulae. (The operator '|' is only recognised by some functions.)*

For the *Beilschmiedia* data, we have already looked at an additive model involving the terrain elevation `elev` and terrain slope `grad`. Fitting a model with interaction between these covariates is just as easy:

```
> fit <- ppm(bei ~ elev * grad, data=bei.extra)
> fit
Nonstationary Poisson process
Log intensity:  ~elev * grad
Fitted trend coefficients:
(Intercept)        elev        grad   elev:grad
  -4.389734   -0.007115  -36.608966    0.293532
```

9.3.9.2 Interaction between two factors

If `A` and `B` are both factors, the formula `X ~ A * B` represents the (overparameterised) mathematical model

$$\lambda(u) = \exp(\mu + \alpha_{A(u)} + \beta_{B(u)} + \gamma_{A(u),B(u)}) \tag{9.24}$$

where $A(u)$ is the level of factor `A` at the location u, and $B(u)$ is the level of `B` at u. This model has parameters μ (the intercept), $\alpha_1,\ldots,\alpha_L$ (the main effects of the levels of `A`), $\beta_1,\ldots,\beta_M$ (the main effects of the levels of `B`), and $\gamma_{1,1},\ldots,\gamma_{L,M}$ (the interaction effects for each combination of levels of `A` and `B`). At a location u where $A(u) = i$ and $B(u) = j$, the predicted intensity is $\exp(\mu + \alpha_i + \beta_j + \gamma_{i,j})$. Compared with the additive model

$$\lambda(u) = \exp(\mu + \alpha_{A(u)} + \beta_{B(u)}), \tag{9.25}$$

the interaction model (9.24) has an additional 'synergy'or 'catalysis' term between the individual covariates (which could be negative; a 'counter-synergy') of the form $\gamma_{A(u),B(u)}$ which corresponds to the interaction term `A:B`.

The gorillas dataset (page 314) is supplied with factor covariates `vegetation` and `heat`. A model involving interaction between these two factors is:

```
> ppm(gor ~ vege * heat, data=gex)
Nonstationary Poisson process
Log intensity:  ~vege * heat
Fitted trend coefficients:
      (Intercept)          vegeColo          vegeGras          vegePrim
           2.6687          -13.9713           -1.1268            1.7189
         vegeSeco          vegeTran          heatMode          heatCool
           0.6746            0.8753           -0.7432           -0.8605
vegeColo:heatMode vegeGras:heatMode vegePrim:heatMode vegeSeco:heatMode
```

SAMPLE

```
           16.5848            0.7797            0.8326            0.9814
vegeTran:heatMode vegeColo:heatCool vegeGras:heatCool vegePrim:heatCool
            1.3636           17.9549          -11.9840            0.9214
vegeSeco:heatCool vegeTran:heatCool
         -13.7854                NA
*** Model is not valid ***
*** Some coefficients are NA or Inf ***
```

The result gives a fitted log intensity for every possible combination of vegetation type and heat load index. Note that one of the fitted coefficients is NA, because the corresponding combination of `vegetation` and `heat` does not occur in the data — at least, not at any of the quadrature points (sample locations used to fit the model).

9.3.9.3 Interaction between factor and numerical covariate

If A is a factor and Z is numerical, the additive model `X ~ A + Z` is

$$\lambda(u) = \exp(\mu + \alpha_{A(u)} + \beta Z(u)) \tag{9.26}$$

while the full interaction model `X ~ A * Z` is

$$\lambda(u) = \exp(\mu + \alpha_{A(u)} + \beta Z(u) + \gamma_{A(u)} Z(u)). \tag{9.27}$$

In the additive model (9.26) the parameters $\alpha_1, \ldots, \alpha_L$ are the effects of the different levels of the factor A, while the 'slope' parameter β is the effect of a unit increase in the numerical covariate Z. If we were to plot the linear predictor (the log intensity) as a function of Z for different levels of the covariate, the graph would consist of parallel lines with the same slope β but with different intercepts $\mu + \alpha_i$.

In the interaction model (9.27) there is an extra term $\gamma_{A(u)} Z(u)$ which causes the effect of a unit increase in Z to depend on the level of the factor A. The plot of the linear predictor described above would show lines with *different* slopes $\beta + \gamma_{A_i}$.

The Murchison data (Section 9.3.2.2) include two spatial objects serving as covariate data: a window `greenstone` and a line segment pattern `faults`. The `greenstone` window is automatically interpreted as a logical-valued covariate function, equal to TRUE inside the window. From the `faults` pattern we have constructed a numerical spatial covariate function `dfault`. The model with interaction is:

```
> ppm(gold ~ dfault * greenstone, data=mur)
Nonstationary Poisson process
Log intensity:  ~dfault * greenstone
Fitted trend coefficients:
          (Intercept)                dfault        greenstoneTRUE
              -6.0184               -0.2047                2.0013
dfault:greenstoneTRUE
               0.1674
```

The estimates of the coefficients in (9.27) are $\mu = \texttt{315.4}$, $\beta = \texttt{316.3}$, $\alpha_{\texttt{FALSE}} = 0$, $\alpha_{\texttt{TRUE}} = \texttt{316.5}$, $\gamma_{\texttt{FALSE}} = 0$, and $\gamma_{\texttt{TRUE}} = \texttt{317.6}$.

9.3.9.4 Nested interaction

It is sometimes appropriate to fit a model with the formula `X ~ A + A:B`. That formula can also be rendered as `y ~ A/B`. The operator / stands for a *nested interaction* and we read A/B as 'B nested within A'.

For example, in a taxonomic classification of organisms, suppose G and S are the Genus and

SAMPLE

Species names, treated as factors. The Species name only makes sense when the Genus is specified. We may wish to compare models which depend only on Genus, of the form `y ~ G`, with models which depend on Species, of the form `y ~ G/S`.

Nesting a numerical covariate inside a factor is effectively the same as crossing the two covariates. For the Murchison data, we may try nesting the numerical covariate `dfault` inside the factor `greenstone`:

```
> ppm(gold ~ greenstone/dfault, data=mur)
Nonstationary Poisson process
Log intensity:  ~greenstone/dfault
Fitted trend coefficients:
          (Intercept)         greenstoneTRUE greenstoneFALSE:dfault
             -6.01845                2.00131               -0.20466
 greenstoneTRUE:dfault
             -0.03728
> ppm(gold ~ greenstone/dfault - 1, data=mur)
Nonstationary Poisson process
Log intensity:  ~greenstone/dfault - 1
Fitted trend coefficients:
       greenstoneFALSE         greenstoneTRUE greenstoneFALSE:dfault
             -6.01845               -4.01714               -0.20466
 greenstoneTRUE:dfault
             -0.03728
```

These two models are exactly equivalent, except for the way in which they are parametrised. They are also equivalent to the interaction model with formula `gold ~ greenstone * dfault`. Different choices of parameterisation make it easier or harder to extract particular kinds of information.

Nesting a factor `A` inside a numerical covariate `Z` gives a model where the effect of a unit change in `Z` depends on the level of `A`, but the factor `A` has no effect on the intercept. In the plot of the linear predictor against `Z` described above, the lines are not parallel, but they all have the same intercept. For the Murchison data, the model `gold ~ dfault/greenstone` stipulates that the intensity of gold deposits very close to a geological fault (`dfault` ≈ 0) must be the same inside and outside the greenstone, but as we increase distance from the faults, the intensity may decrease at different rates inside and outside the greenstone:

```
> ppm(gold ~ dfault/greenstone, data=mur)
Nonstationary Poisson process
Log intensity:  ~dfault/greenstone
Fitted trend coefficients:
         (Intercept)                dfault dfault:greenstoneTRUE
             -4.3472               -0.4972                0.5020
```

Nested interactions also arise in connection with random effects, as discussed in Chapter 16.

In nested designs the operator `%in%` is sometimes useful. It is equivalent to an interaction: `A %in% B` is equivalent to `A:B`.

9.3.10 Formulae involving many variables

A model can involve any number of covariates. In an additive model, the variable names are simply joined by the '+' operator:

```
ppm(Y ~ X1 + X2 + X3 + ... + Xn)
```

This expression could be long and tedious to type out. A useful shortcut is the symbol '`.`' representing *all* the available covariates. This only works when the covariates are supplied in the argument `data` (see page 306). The additive model involving all covariates can be fitted using the formula '`Y ~ .`' as in

```
> ppm(gor ~ . , data=gex)
```

To fit the additive model involving all covariates *except* the `heat` covariate,

```
> ppm(gor ~ . - heat, data=gex)
```

Interactions between more than two predictors ('higher order interactions') are easy to define. For three factors `A`, `B`, and `C` the second-order interactions are `A:B`, `A:C`, and `B:C`. The third-order interaction is `A:B:C`. The mathematical expressions defining higher-order interactions get increasingly cumbersome to write down, but the ideas are basically the same as for second-order interactions. Higher-order interactions are often difficult to interpret in practical terms.

In the model formula context the `+` and `:` operators are what the pure mathematicians (poor dears!) call 'idempotent' operations. That is, `A+A` is just equal to `A`, and likewise `A:A` is equal to `A`.

The crossing operator `*` obeys the distributive and associative laws, so that `(A + B + C) * (A + B + C)` expands to `A + B + C + A:B + A:C + B:C`, and `A * B * C` expands to `A + B + C + A:B + A:C + B:C + A:B:C`.

We have now dealt with the model operators `+`, `-`, `:`, `*`, and `/`. The last operator to be mentioned is the power operator `^`. If `A` is any model term, `A^2` is equivalent to `A * A`, while `A^3` is equivalent to `A * A * A` and so on. This is particularly useful when we want to specify all main effects and interactions up to a certain order. For example `(A + B + C)^2` is equivalent to `(A + B + C) * (A + B + C)` which expands to `A + B + A:B + A:C + B:C`, containing all main effects and second order interactions between the covariates `A`, `B`, and `C`. One could also use the '`.`' symbol, for example

```
> ppm(gor ~ .^2, data=gex)
```

would fit all main effects and all second-order interaction terms involving all the covariates in the list `gex`.

Tip: To check the expansion of a complicated formula `f`, try `update(f, . ~ .)` This uses the symbol '`.`' in another sense, explained in Section 9.4.4.

9.4 Statistical inference for Poisson models

9.4.1 Fitted models

The value returned by the model-fitting function `ppm` is an object of class `""ppm""` that represents the fitted model. This is analogous to the fitting of linear models (`lm`), generalised linear models (`glm`), and so on. There are many standard operations on fitted models in R which can be applied to point process models: these are listed in Table 9.3. That is, these generic operations have methods for the class `"ppm"`. For information on these methods, consult the help for `print.ppm`, `summary.ppm`, `plot.ppm`, etc. The methods are described in the present chapter, except for `residuals`, `influence`, and `dfbetas` which are described in Chapter 11.

Additionally Table 9.4 lists some non-generic functions in the base R system which work on

SAMPLE

`print`	print basic information
`summary`	print detailed summary information
`plot`	plot the fitted intensity
`predict`	compute the fitted intensity
`simulate`	generate simulated realisations of model
`update`	re-fit the model
`coef`	extract the fitted coefficient vector $\widehat{\boldsymbol{\theta}}$
`vcov`	variance-covariance matrix of $\widehat{\boldsymbol{\theta}}$
`anova`	analysis of deviance
`logLik`	loglikelihood value
`formula`	extract the model formula
`terms`	extract the terms in the model
`fitted`	compute the fitted intensity at data points
`residuals`	compute residuals
`influence`	compute likelihood influence diagnostics
`dfbetas`	compute parameter influence diagnostics
`model.matrix`	compute the design matrix

Table 9.3. *Standard* R *generic operations which have methods for point process models (class* `"ppm"`*).*

`"ppm"` objects. For information on these functions, consult the help for the function itself. Finally Table 9.5 on page 338 lists some generic functions defined in `spatstat` which apply to `"ppm"` objects and are useful for Poisson models.

`confint`	confidence intervals for parameters
`step`	stepwise model selection
`stepAIC`	(package `"MASS"`) stepwise model selection
`drop1`	one step model deletion
`add1`	one step model augmentation
`AIC`	Akaike Information Criterion

Table 9.4. *Functions in the base* R *system which work on* `"ppm"` *objects.*

The `print` method gives important information about the model structure and the fitted model parameters. For Poisson models it also gives a table of parameter estimates with standard errors, confidence intervals, and the result of a test that the parameter is significantly different from zero. For example with the *Beilschmiedia* data we may do the following.

```
> beikm <- rescale(bei, 1000, unitname="km")
> bei.extrakm <- lapply(bei.extra, rescale, s=1000, unitname="km")
> fitkm <- ppm(beikm ~ x + y)
> fitkm

Nonstationary Poisson process

Log intensity:  ~x + y

Fitted trend coefficients:
(Intercept)           x           y 
     9.0910     -0.8031      0.6496 

            Estimate    S.E. CI95.lo CI95.hi Ztest    Zval
(Intercept)   9.0910 0.04306  9.0066  9.1754   *** 211.128
```

SAMPLE

```
x             -0.8031 0.05863 -0.9180 -0.6882   *** -13.698
y              0.6496 0.11571  0.4228  0.8764   ***   5.614
```

This is the fitted model with intensity function

$$\lambda_{\boldsymbol{\theta}}((x,y)) = \exp(\theta_0 + \theta_1 x + \theta_2 y) \tag{9.28}$$

where spatial coordinates x and y are measured in kilometres and, for example, the estimate of θ_1 is $\hat{\theta}_1 = -0.8031$ with standard error $\mathsf{se}(\hat{\theta}_1) = 0.05863$ and 95% confidence interval $[-0.918, -0.6882]$. The amount of information displayed, and the layout, depend on `spatstat.options('terse')` and `spatstat.options('print.ppm.SE')`.

The fitted coefficients of the model can be extracted by `coef.ppm`, a method for the generic function `coef`:

```
> coef(fitkm)

(Intercept)           x           y
     9.0910     -0.8031      0.6496
```

The `plot` method is useful for initial inspection of a fitted model, because it offers a wide range of displays. Figure 9.9 shows the result of the command

```
> plot(fitkm, how="image", se=FALSE)
```

with modifications to the graphics parameters.

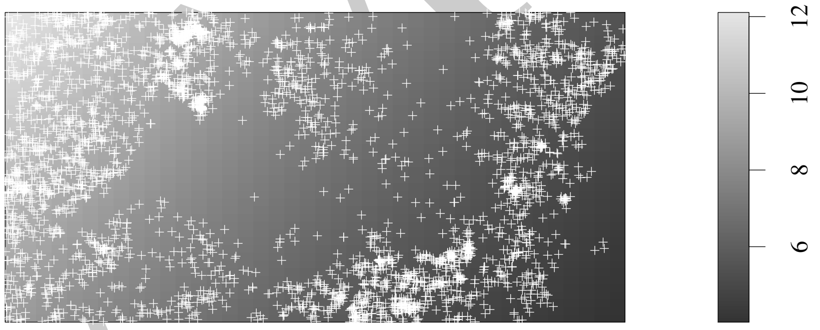


Figure 9.9. *Result of plotting a fitted Poisson model.*

Very detailed information about the model can be printed by typing `summary(fitkm)`. The result of `summary.ppm` is an object of class `summary.ppm` which can be printed and manipulated in other ways. For example the table of parameter estimates, standard errors, and confidence intervals is obtained by

```
> coef(summary(fitkm))

             Estimate    S.E. CI95.lo CI95.hi Ztest    Zval
(Intercept)    9.0910 0.04306  9.0066  9.1754   *** 211.128
x             -0.8031 0.05863 -0.9180 -0.6882   *** -13.698
y              0.6496 0.11571  0.4228  0.8764   ***   5.614
```

SAMPLE

9.4.2 Standard errors and confidence intervals for parameters

The accuracy of the fitted model coefficients can be analysed by standard asymptotic theory as explained in Section 9.7 and in [396, 565]. For sufficiently large samples from a loglinear Poisson model (9.6), the estimated parameters $\hat{\boldsymbol{\theta}}$ have approximately a multivariate normal distribution with mean equal to the true parameters $\boldsymbol{\theta}$ and variance-covariance matrix $\mathrm{var}\,\hat{\boldsymbol{\theta}} = \boldsymbol{I_\theta}^{-1}$, where $\boldsymbol{I_\theta}$ is the Fisher information matrix

$$\boldsymbol{I_\theta} = \int_W \mathbf{Z}(u)\,\mathbf{Z}(u)^\mathsf{T} \lambda_{\boldsymbol{\theta}}(u)\,\mathrm{d}u. \tag{9.29}$$

The estimated variance-covariance matrix $\widehat{\mathrm{var}}\,\hat{\boldsymbol{\theta}} = \boldsymbol{I_{\hat{\theta}}}^{-1}$ is computed by `vcov.ppm`, a method for the generic function `vcov`.

```
> vcov(fitkm)
            (Intercept)          x          y
(Intercept)    0.001854 -1.491e-03 -3.528e-03
x             -0.001491  3.438e-03  1.208e-08
y             -0.003528  1.208e-08  1.339e-02
```

The diagonal of this matrix contains the estimated variances of the individual parameter estimates $\theta_0, \theta_1, \theta_2$, so the standard errors are:

```
> sqrt(diag(vcov(fitkm)))
(Intercept)           x           y
    0.04306     0.05863     0.11571
```

Confidence intervals for the parameters $\boldsymbol{\theta}$ can be obtained from the standard function `confint`:

```
> confint(fitkm, level=0.95)
              2.5 %  97.5 %
(Intercept)  9.0066  9.1754
x           -0.9180 -0.6882
y            0.4228  0.8764
```

and these could also be constructed manually from the parameter estimate `coef(fitkm)` and the standard error `sqrt(diag(vcov(fitkm)))`.

The estimated correlation between individual parameter estimates can be useful in detecting collinearity and confounding. Correlations can be computed using `vcov.ppm`:

```
> co <- vcov(fitkm, what="corr")
> round(co, 2)
            (Intercept)     x     y
(Intercept)        1.00 -0.59 -0.71
x                 -0.59  1.00  0.00
y                 -0.71  0.00  1.00
```

This suggests fairly strong negative correlation between the intercept parameter estimate $\hat{\theta}_0$ and the estimates of the coefficients of x and y. However, the correlation between an intercept estimate and a slope estimate depends on the origin for the covariate. We get a different answer if we place the coordinate origin in the centre of the study region:

```
> fitch <- update(fitkm, . ~ I(x-0.5) + I(y-0.25))
> co <- vcov(fitch, what="corr")
> round(co, 2)
```

SAMPLE

```
             (Intercept) I(x - 0.5) I(y - 0.25)
(Intercept)         1.00       0.23       -0.09
I(x - 0.5)          0.23       1.00        0.00
I(y - 0.25)        -0.09       0.00        1.00
```

Standard errors and confidence intervals for these parameters also depend on the origin for the covariate.

9.4.3 Prediction

For any Poisson model with intensity $\lambda(u) = \lambda_{\boldsymbol{\theta}}(u)$ where $\boldsymbol{\theta}$ is a parameter vector, the *fitted intensity* or *predicted intensity* is the function $\lambda_{\widehat{\boldsymbol{\theta}}}(u)$ obtained by substituting the fitted parameter estimates into the intensity formula.

The fitted intensity is computed by `predict.ppm`, a method for the generic function `predict`. By default it computes the fitted intensity at a regular grid of locations yielding a pixel image:

```
> fit <- ppm(bei ~ polynom(grad, elev, 2), data=bei.extra)
> lamhat <- predict(fit)
```

The result is shown in the left panel of Figure 9.10. Alternatively `predict.ppm` can be used to obtain predicted values at any locations, including the original data points:

```
> lamB <- predict(fit, locations=bei)
```

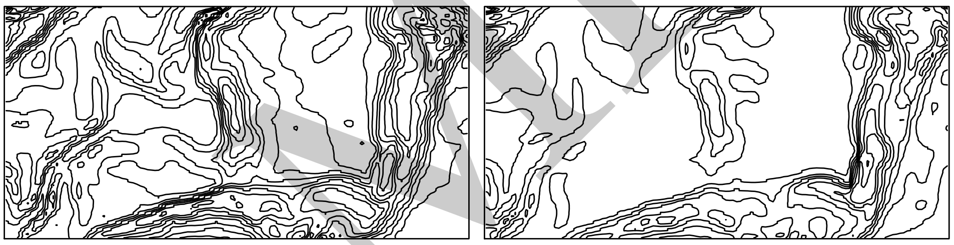


Figure 9.10. *Contour plots of the fitted intensity* `predict(fit)` (Left) *and standard error* `predict(fit, se=TRUE)$se` (Right).

There are many useful ways to plot the fitted intensity function, using the tools described in Section 4.1.4 of Chapter 4. The front cover illustration of this book (shown in grey scale in Figure 9.11) is a perspective view of the rainforest terrain, coloured according to the value of the fitted intensity, with the *Beilschmiedia* tree locations added. It was produced by the following code:

```
> M <- persp(bei.extra$elev, colin=lamhat, colmap=topo.colors,
             shade=0.4, theta=-55, phi=25, expand=6,
             box=FALSE, apron=TRUE, visible=TRUE)
> perspPoints(bei, Z=bei.extra$elev, M=M, pch=20, cex=0.1)
```

More detail about the use of `persp.im` and `perspPoints` can be found on page 88.

For a loglinear model (9.6), the asymptotic variance of the predicted intensity $\lambda_{\hat{\boldsymbol{\theta}}}(u)$ at a given location u is

$$\operatorname{var}\hat{\lambda}(u) \sim \mathbf{Z}(u)\boldsymbol{I}_{\boldsymbol{\theta}}{}^{-1}\mathbf{Z}(u)^{\top}\lambda_{\boldsymbol{\theta}}(u). \tag{9.30}$$

The right panel of Figure 9.10 shows the standard error, i.e. the square root of this variance, which is computed by `predict(fit, se=TRUE)$se`.

A confidence interval for the true value of $\lambda(u)$ at each location u can be computed by

SAMPLE

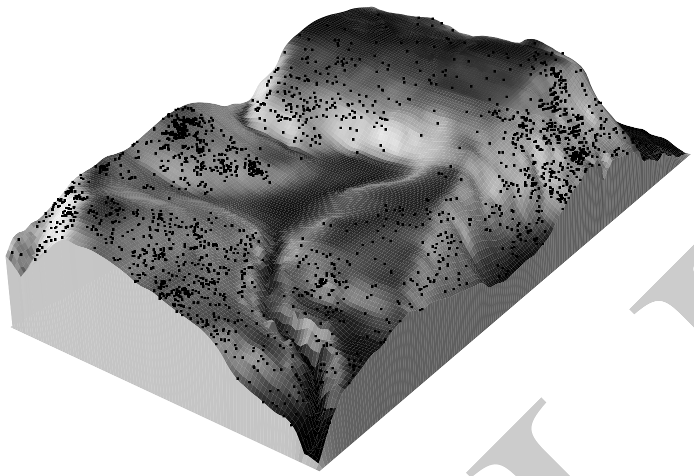


Figure 9.11. *Perspective view of rainforest terrain shaded according to intensity of fitted model, with original point pattern superimposed. Colour version on front cover of book.*

`predict(fit, interval="confidence")`. This yields a list of two images giving the lower and upper limits of the confidence interval.

It is also possible to plot the 'effect' of a single covariate in the model. The command `effectfun` computes the intensity of the fitted model as a function of one of its covariates. An example was shown in Figure 9.3.

We may also be interested in predicting the number of points in a spatial region B. For a Poisson process, the expected number of points in B is the integral of the intensity over B, equation (9.1) on page 301. The fitted mean (expected) number of points in B is

$$\hat{\mu}(B) = \int_B \hat{\lambda}(u)\,\mathrm{d}u.$$

This can be computed using `predict.ppm` by setting `type="count"` and specifying the region B as `window`. To find the expected number of trees at elevations below 130 metres:

```
> B <- levelset(bei.extra$elev, 130)
> predict(fit, type="count", window=B)
[1] 37.4
```

The asymptotic variance of $\hat{\mu}(B)$ is

$$\operatorname{var}\hat{\mu}(B) \sim M(B)\boldsymbol{I_\theta}^{-1}M(B)^\top \tag{9.31}$$

where

$$M(B) = \int_B \mathbf{Z}(u)\lambda_{\boldsymbol{\theta}}(u)\,\mathrm{d}u \tag{9.32}$$

is the expected sum of the covariate values at the data points, by Campbell's formula (6.11). This allows us to compute the standard error for the estimate of the mean number of trees:

```
> predict(fit, B, type="count", se=TRUE)
$estimate
[1] 37.4

$se
[1] 3.631
```

SAMPLE

or a confidence interval for the true expected number (9.1):

```
> predict(fit, B, type="count", interval="confidence")
 2.5% 97.5%
30.28 44.51
```

In this case it is also meaningful to compute a *prediction interval* for the random number of trees in the specified region:

```
> predict(fit, B, type="count", interval="prediction")
 2.5% 97.5%
   23    51
```

It is not clear exactly what this means for the *Beilschmiedia* data. A 95% prediction interval for the number of points in region B is designed so that, if the experiment were repeated, there is a 95% probability that the random value of $n(\mathbf{X} \cap B)$ would lie in the interval. The concept of 'repeating the experiment' makes more sense if there is a time dimension — for example if the point pattern is a record of accidents reported in a specific year. Assuming independence between successive years of observations, there is a 95% probability that next year's count of accidents in the same region will lie between the calculated limits. The calculation assumes that the model is true, but takes into account the uncertainty in the model parameters due to estimation from the data.

9.4.4 Updating a model

In data analysis we typically fit several different candidate models to the same data [181, 180]. The generic function `update` makes it easy to do this. It modifies a fitted model, for example by changing the model formula and re-fitting the model.

The method `update.ppm` is provided in `spatstat` for updating a fitted point process model. The syntax is

```
    update(object, ...)
```

where `object` is a fitted point process model (class `"ppm"`) and the subsequent arguments '`...`', if any, determine how the model should be changed. The result is another fitted model of the same kind.

The syntax `update(object)` makes sense, and causes the `object` to be re-fitted. The result is different from the original `object` if any of the data used to fit the original model have changed.

The second argument of `update` may be a formula, specifying the new model formula to be used in re-fitting the model.

```
> fitcsr <- ppm(bei ~ 1, data=bei.extra)
> update(fitcsr, bei ~ grad)
Nonstationary Poisson process
Log intensity:  ~grad
            Estimate    S.E. CI95.lo CI95.hi Ztest    Zval
(Intercept)   -5.391 0.03002  -5.449  -5.332   *** -179.58
grad           5.022 0.24540   4.541   5.503   ***   20.46
```

Notice that we needed to provide the `data` argument to the first model `fitcsr`, even though that model does not depend on any covariates, in order for the covariate `grad` to be available for the updated model. This would have been unnecessary if `grad` had been an existing object in the R session.

The new formula may include the symbol '`.`' representing *'what was there before'*. To keep the same left-hand side of the formula, use '`.`' on the left-hand side:

SAMPLE

```
> fitgrad <- update(fitcsr, . ~ grad)
```

To modify the right-hand side of the formula, use '`.`' on the right-hand side, modifying it with the model operators:

```
> fitall <- update(fitgrad, . ~ . + elev)
> fitall
Nonstationary Poisson process
Log intensity:  ~grad + elev
             Estimate     S.E.   CI95.lo  CI95.hi Ztest    Zval
(Intercept) -8.55862 0.341101 -9.22717 -7.89008   *** -25.091
grad         5.84104 0.255861  5.33956  6.34252   ***  22.829
elev         0.02141 0.002288  0.01693  0.02589   ***   9.358
```

To remove the intercept, use `-1` or `+0`:

```
> fitp <- update(fitall, . ~ . - 1)
```

The symbolic manipulation of formulae involving the symbol '`.`' is handled by the update method for formulae, `update.formula`. This is quite useful in its own right as a quick way to see the effect of a change in a formula:

```
> update(gor ~ (heat + vege)^2,    . ~ . - heat:vege)
gor ~ heat + vege
```

and simply to expand a complicated formula:

```
> update(gor ~ (heat + vege + sang)^3, . ~ .)
gor ~ heat + vege + sang + heat:vege + heat:sang + vege:sang +
    heat:vege:sang
```

Warning: the operator '`-`' *does not remove offset terms* in model formulae. A command like `update(fit, . ~ . - offset(A))` does not delete the term `offset(A)`.

9.4.5 Testing significance of a term in the model

Often we want to decide whether a particular covariate Z has an effect on the point pattern (see Sections 6.6–6.7). In a modelling context this can be studied by comparing two different models, which are identical except that one of the models includes a term depending on the covariate Z. For example, to decide whether the *Beilschmiedia* forest density depends on terrain slope, the two models would be:

```
> fit0 <- ppm(bei ~ 1)
> fit1 <- ppm(bei ~ grad, data=bei.extra)
```

We can assess the strength of evidence for the covariate effect by performing a *significance test* (Chapter 10) comparing the two models. The null hypothesis is the smaller model `fit0` which does not include the covariate effect; the alternative hypothesis is the larger model `fit1` including the covariate effect.

The theoretically optimal technique is the Likelihood Ratio Test (Section 10.3.2) which can be carried out using **analysis of deviance**, the generalisation of analysis of variance. It is performed in `spatstat` using `anova.ppm`, a method for the generic function `anova`:

```
> anova(fit0, fit1, test="LR")
```

SAMPLE

```
Analysis of Deviance Table
Model 1: ~1          Poisson
Model 2: ~grad          Poisson
  Npar Df Deviance Pr(>Chi)
1    1
2    2  1      382   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This example shows the likelihood ratio test of the null hypothesis of CSR against the alternative of an inhomogeneous Poisson process with intensity that is a loglinear function of the slope covariate (9.7). The *p*-value, shown under the heading `Pr(>Chi)`, is extremely small, indicating rejection of CSR in favour of the alternative. Note that `2e-16` or 2×10^{-16} is the smallest detectable difference between 'real numbers'[4] on the 32-bit computer which produced this output, so the output says that the *p*-value is effectively zero.

An advantage of this approach is that we can easily allow for the effects of other covariates, by including these 'other covariates' in both models. For example, we could test the significance of a terrain slope effect after accounting for the effect of terrain elevation:

```
> fit2e <- ppm(bei ~ polynom(elev, 2), data=bei.extra)
> fit2e1g <- update(fit2e, . ~ . + grad)
> anova(fit2e, fit2e1g, test="LR")
Analysis of Deviance Table
Model 1: ~elev + I(elev^2)          Poisson
Model 2: ~elev + I(elev^2) + grad          Poisson
  Npar Df Deviance Pr(>Chi)
1    3
2    4  1      418   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Similarly, for the Chorley-Ribble data (page 9) the key question is whether distance to the incinerator has an effect on the intensity of laryngeal cancer, after allowing for the fact that the intensity of laryngeal cancer cases will depend on the spatially varying density of the population at risk. A likelihood ratio test can be used to compare two models, both involving a term related to the population density, but only one model including a term related to distance from the incinerator.

9.4.6 Model selection using AIC

Model selection [154] is the task of choosing the 'best fitting' statistical model from amongst several competing models for the same dataset.

Hypothesis testing, as described above, can be regarded as a special case of model selection. Alternatives to hypothesis testing are sometimes required. For example, the likelihood ratio test requires the null hypothesis to be a sub-model of the alternative. Unfortunately the model (9.18), in which forest intensity is proportional to terrain slope, does not include homogeneous intensity as a special case, so we cannot use analysis of deviance to test the null hypothesis of CSR against the alternative of an inhomogeneous Poisson process with intensity (9.18).

Models which are not nested can nevertheless be compared using the *Akaike Information Criterion* [9]

$$\mathrm{AIC} = -2 \log L_{max} + 2p \qquad (9.33)$$

[4] `.Machine$double.eps` gives the smallest double-precision floating-point number `x` that satisfies `1 + x != 1`.

SAMPLE

where $L_{max} = L(\widehat{\boldsymbol{\theta}})$ is the maximised likelihood for the model in question, and p the number of parameters for this model. The model with the *lowest* value of AIC is preferred. This strikes a balance between the degree of fit and the complexity of the model.

The R function `AIC` computes the AIC of any model, including a `"ppm"` object:

```
> fitprop <- ppm(bei ~ offset(log(grad)),data=bei.extra)
> fitnull <- ppm(bei ~1)
> AIC(fitprop)
[1] 42497
> AIC(fitnull)
[1] 42764
```

In this case the preferred model is `fitprop`, the model (9.18) with intensity proportional to slope.

Model selection techniques are especially valuable when there are many covariates, or many possible terms which could be included in the model. One very simple approach is 'forward stepwise selection', in which we start with a minimal acceptable model (containing few terms), and add new terms to the model, one by one. In 'backward stepwise selection' we start with a maximal model (containing many terms) which is believed to be adequate, and delete terms from the model, one by one.

We could use the Likelihood Ratio Test as the criterion for deciding whether to add a term (or to delete a term) at each step. However, this often results in models which are smaller than they should be: the customary significance level $\alpha = 0.05$ is an excessively stringent standard of evidence for including a proposed term in the model. A better approach is to compare models using the AIC.

In stepwise model selection, choosing the model with the smaller AIC is equivalent to applying the likelihood ratio test, but taking the critical value to be $2d$ where d is the number of degrees of freedom, i.e. the number of added or deleted parameters. The significance level of this test is $\alpha = 0.157, 0.135, 0.112$ when $d = 1, 2, 3$, respectively, dropping below 0.05 when $d = 8$.

The `stats` package (included in a standard installation of R) provides functions `add1`, `drop1`, and `step` to perform 'automatic' stepwise model selection. The `MASS` package provides `stepAIC`, an improved version of `step`. These functions can be applied to many kinds of models, including fitted point process models. The function `drop1` compares a model with all the sub-models obtained by deleting a single term, and evaluates the AIC for each sub-model. For example, to explore the dependence of the intensity of the Swedish Pines data on the Cartesian coodinates, we could do:

```
> fitxy <- ppm(swedishpines ~ x + y)
> drop1(fitxy)
Single term deletions

Model:
~x + y
       Df AIC
<none>    844
x       1 843
y       1 842
```

The output indicates that the lowest AIC would be achieved by deleting the `y` term. Similarly `add1` compares a model with all the 'super-models' obtained by adding a single term:

```
> fitcsr <- ppm(swedishpines ~ 1)
> add1(fitcsr, ~x+y)
Single term additions

Model:
```

SAMPLE

```
~1
       Df AIC
<none>    841
x       1 842
y       1 843
```

The output indicates that CSR has lower AIC than the models in which we add the term `x` or add the term `y`.

The function `step` performs stepwise model selection using AIC. By default it performs backward stepwise selection. Starting from a 'maximal' model, the procedure considers each term in the model, and decides whether the term should be deleted. The deletion yielding the biggest reduction in AIC is carried out. This is applied recursively until no more terms can be deleted (i.e. until all further deletions would lead to an *increase* in AIC).

Continuing the example:

```
> fitxy <- ppm(swedishpines ~ x + y)
> fitopt <- step(fitxy)
Start:  AIC=843.6
~x + y

       Df AIC
- y     1 842
- x     1 843
<none>    844

Step:  AIC=841.6
~x

       Df AIC
- x     1 841
<none>    842

Step:  AIC=840.8
~1
> fitopt
Stationary Poisson process
Intensity: 0.007396
            Estimate   S.E. CI95.lo CI95.hi Ztest   Zval
log(lambda)   -4.907 0.1187  -5.139  -4.674   *** -41.35
```

The output here is different from the output of `drop1` or `add1`. At each step in the procedure, the possible deletions are ranked in ascending order of AIC: the deletion giving the greatest reduction in AIC is at the top of the table. The output shows that the term `y` was first deleted from the model, and then the term `x` was deleted, leaving only the formula `~1` corresponding to CSR.

In what follows, we set `trace=0` to suppress the progress reports, so the output shows only the final result. Larger values of `trace` give more information about the sequence of models considered. The default is `trace=1`. For brevity one can use `formula.ppm` to extract the model formula:

```
> bigfit <- ppm(swedishpines ~ polynom(x,y,3))
> formula(bigfit)
~x + y + I(x^2) + I(x * y) + I(y^2) + I(x^3) + I(x^2 * y) + I(x * y^2) + I(y^3)
> goodfit <- step(bigfit, trace=0)
> formula(goodfit)
```

SAMPLE

```
~x + y + I(x * y) + I(y^2)
> AIC(goodfit)
[1] 841.5
```

The selected model goodfit has slightly larger AIC than the constant intensity model in fitopt. This illustrates the fact that dropping terms one at a time is not guaranteed to produce the submodel with minimal AIC.

Many other methods for model selection exist [154]. We also warn that there are some perils in automatic model selection techniques: see [470]. See Section 13.5.11 for a discussion of model selection using AIC for *Gibbs* point process models.

9.4.7 Simulating the fitted model

A fitted Poisson model can be simulated automatically using the function simulate.ppm. The result of the commands `X <- simulate(fitprop); plot(X[[1]])` is shown in Figure 9.12.

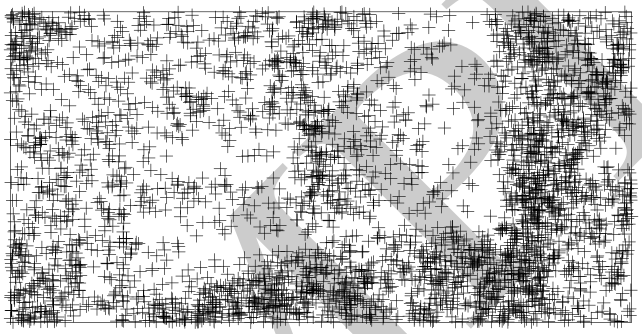


Figure 9.12. *Simulated realisation of a Poisson model fitted to the* Beilschmiedia *data.*

It is also possible to perform conditional simulation (conditional on the number of points, on the configuration of points in a particular subregion, or on the presence of certain points). See Section 13.9.1.4 or the help for rmhcontrol.

9.4.8 Additional capabilities

Table 9.5 lists some additional generic functions defined in spatstat which have methods for fitted models. For the full list, see Table 13.1 on page 508 and Table 13.5 on page 536, or type `methods(class="ppm")`.

envelope	Simulation envelopes of summary function
berman.test	Berman's tests
cdf.test	Spatial CDF tests
quadrat.test	χ^2 test based on quadrat counts
leverage	Leverage diagnostic
model.images	Pixel images of the canonical covariates
relrisk	Relative risk predicted by model
rhohat	Relative intensity as a function of covariate

Table 9.5. *Some of the generics defined in* spatstat *with methods for class* "ppm" *which are useful for Poisson models.*

The R generic function model.matrix computes the 'design matrix' of a model. For a Poisson point process model, model.matrix.ppm returns a matrix with one row for each quadrature point,

SAMPLE

and one column for each of the *canonical* covariates appearing in (9.6). Note that, if the model formula involves transformations of the original covariates, then `model.matrix(fit)` gives values of these transformed covariates.

The `spatstat` package also defines a new generic function `model.images` with a method for `"ppm"` objects. This produces a list of *pixel images* of the canonical covariates.

```
> fit2 <- ppm(bei ~ sqrt(grad) + x, data=bei.extra)
> mo <- model.images(fit2)
> names(mo)
[1] "(Intercept)" "sqrt(grad)"  "x"
```

Additionally `spatstat` has some functions that can be applied to any fitted model (of class `"lm"`, `"glm"`, `"ppm"`, etc.) to understand the structure of the model: `model.covariates` determines which of the original covariates in `data` are used in the model; `model.depends` identifies which of the covariates is involved in each term of the model; `model.is.additive` determines whether the model is additive; and `has.offset` determines whether the model has an offset.

9.5 Alternative fitting methods

Here we review alternative options for fitting the loglinear Poisson model in `spatstat` and related packages.

Faster fitting

If `ppm` is called with the option `method="logi"`, it fits the point process model using conditional logistic regression, as described in Section 9.10. This method is typically faster than the default technique, and often more reliable, but involves randomisation.

More robust fitting

The *lasso* [655] is an improvement to classical regression which increases its robustness against anomalies or extremes in the data. The lasso version of linear regression, for example, finds the parameter value θ that maximises $RSS(\theta) - \alpha\|\theta\|$, where $RSS(\theta)$ is the usual sum of squared errors, $\|\theta\| = \sum_j |\theta_j|$ is the sum of absolute values of all parameters, and α is a tradeoff coefficient. A lasso version of `ppm` has been implemented in the R package `ppmlasso` [568, 567]. In a nutshell, this approach will fit *the same loglinear model* as is fitted by `ppm`, but with less sensitivity to anomalous observations. Variable selection for point process models using the lasso has recently been studied [654, 709]. In future there may also be a Stein estimator [156].

9.6 More flexible models

Until this point in the chapter, we have been concerned exclusively with fitting *loglinear* Poisson point process models. These models have a specially convenient structure in which the log intensity is a linear function of the *parameters*. Note that the *covariates* can be quite general functions; so, this is a very wide class of models.

However, there are many applications where we need to fit a model which does not have this

SAMPLE

loglinear form. An important example is in spatial epidemiology where the effect of a pollution source may not be easily expressible in loglinear form.

Technique for building and fitting such models are reviewed below.

Generalised additive models

Modern regression tools, including machine learning algorithms [328], smooth function estimators with roughness penalties, and semi-parametric models, allow more flexible modelling than classical regression. Some of these tools are available for point process models in `spatstat` and related packages.

For more flexible models of covariate effects, one standard tool is a generalised additive model [327] in which the covariate effect is modelled as a polynomial or spline function and the model is fitted by maximising $L(\theta) - \alpha Q(\theta)$, where $Q(\theta)$ is a penalty for the roughness of the function, and again α is a tradeoff coefficient. These models can be fitted immediately in `ppm` by including appropriate terms in the model formula, and setting `use.gam=TRUE` to ensure that the fitting is performed by `gam` rather than `glm`. For example,

```
> ppm(gold ~ bs(dfault, 5), data=mur, use.gam=TRUE)
```

fits a model in which the log intensity of Murchison gold deposits is a smooth function of distance to the nearest fault. Here `bs` is part of the standard `splines` package and provides a B-spline basis for smooth functions.

General non-loglinear model

Often a model has a simple mathematical form, with some parameters φ that appear in loglinear form, but other parameters ψ that do not. In the model

$$\lambda(u) = \exp\left(\varphi Z(u) + \frac{1}{1+\psi Y(u)}\right)$$

where $Y(u), Z(u)$ are known covariate functions, the parameter φ appears in loglinear form, but the parameter ψ does not. The principle of maximum likelihood can still be applied to estimate both parameters (φ, ψ), but the algorithm in `ppm` cannot be used to compute the maximum likelihood estimate.

Instead, such models are usually fitted by *maximum profile likelihood*, as described in Section 9.12. Two available algorithms for maximum profile likelihood are brute force maximisation, implemented in `spatstat` by the function `profilepl` described in Section 9.12.2, and Newton's method, implemented in `spatstat` by `ippm` as described in Section 9.12.3. Specialised algorithms also exist for particular models [236].

Local ('geographically weighted') models

In some studies, we may be willing to assume a simple loglinear model, but not willing to assume that the model parameters are constant across the entire study region. For example in the Queensland copper data (Figure 1.11) we may be willing to assume a loglinear relationship between the intensity of copper deposits and distance to the nearest lineament, $\lambda(u) = \exp(\alpha + \beta d(u))$, but we might believe that the coefficients α and β are spatially varying because of the geological history.

Models of this kind can be fitted using the methods of *local likelihood* [335, 428], also known in this context as *geographically weighted regression* [267]. Code for this purpose will be added to `spatstat` shortly. For details, see Section 9.13.

SAMPLE

9.7 Theory∗

This section covers the theory which underpins the methods for fitting a Poisson point process model to a point pattern dataset. Since there has been some confusion in the literature, we shall go to some lengths to explain maximum likelihood estimation for Poisson point processes. See also [484, chapter 3], [396].

9.7.1 Introduction to maximum likelihood

In mainstream statistical methodology, a standard way to fit models to data is by maximising the *likelihood* of the model for the data. For any choice of parameter values, the likelihood value is defined as the probability (or probability density) that the given observations would have been obtained, from the model with these parameter values. For different possible values of the model parameters, the likelihood gives a measure of relative plausibility of these values. The parameter values which maximise the likelihood are the 'most plausible' values, and are called the *maximum likelihood estimates (MLE)*.

Maximum likelihood is theoretically the *optimal* method for parameter estimation, provided (a) the model is true, (b) the model satisfies regularity conditions, and (c) there are many observations and few parameters. Under these conditions, the MLE is more precise than other estimators. The MLE is always logically consistent with the data. However, maximum likelihood can perform poorly if the model is wrong (even if it is only slightly wrong) or if the data contain anomalies.

As an example, suppose that we have counted the number n of insects caught in a trap, and we believe this number follows a Poisson distribution with some mean μ. Our goal is to infer ('estimate') the value of μ from the data. The likelihood is

$$L(\mu) = e^{-\mu}\frac{\mu^n}{n!}$$

from (5.4), where n is the observed number of insects. When we want to stress that this is the likelihood for μ given the data n, we may write $L(\mu) = L(\mu;n)$. We need to find the value of μ which gives the maximum value of $L(\mu)$. It is easier to work with the natural logarithm of the likelihood,

$$\log L(\mu) = -\mu + n\log\mu - \log(n!)$$

and maximising $\log L$ is equivalent to maximising L. Take the derivative of $\log L(\mu)$ with respect to μ (known as the score function or simply the *score*):

$$U(\mu) = U(\mu;n) = \frac{\mathrm{d}}{\mathrm{d}\mu}\log L(\mu) = -1 + \frac{n}{\mu}.$$

The likelihood is maximised when $U(\mu) = 0$, which gives $\mu = n$. That is, the *maximum likelihood estimate* of μ is $\hat{\mu} = n$, the observed number of insects.

Note that the term $\log(n!)$ disappeared in the score. This always happens to additive terms in the loglikelihood (or equivalently multiplicative factors in the likelihood) that only depend on data and not the parameters. Since such terms do not affect the location of the maximum of the (log)likelihood it is common practice to leave them out and still call the function the (log)likelihood.

∗ Starred sections contain advanced material, and can be skipped by most readers.

SAMPLE

9.7.2 Maximum likelihood for CSR

For simplicity we start with the homogeneous Poisson point process (CSR). This model has a single parameter λ, the intensity of the process. Assume we have observed a point pattern **x** inside a spatial window W. The likelihood function is

$$L(\lambda) = L(\lambda; \mathbf{x}) = \lambda^{n(\mathbf{x})} e^{(1-\lambda)|W|} \tag{9.34}$$

where $n(\mathbf{x})$ is the number of points of **x**. The right-hand side is the probability density of observing the pattern **x** if the true intensity is λ. Notice that this probability density depends only on the number of points, and not on their location, because all spatial locations are equally likely under CSR. The point locations are 'ancillary' for λ.

More details about point process likelihoods are given in Section 13.12.1. For the moment, it is useful to appreciate why the likelihood (9.34) consists of two terms, one associated with the data **x** and the other with the window W. Imagine that space is divided into pixels of area a, as illustrated in Figure 5.20 on page 144. If a is small, there is negligible chance that any pixel will contain more than one point, so that each pixel contains either 1 point (with probability λa) or 0 points (with probability $1 - \lambda a$). Pixels are independent of each other, so the probability of a particular configuration of zeroes and ones is obtained by multiplying together the probabilities of the outcomes for each separate pixel. If there are n pixels which contain random points, the presence probability λa will appear n times, giving a factor of $\lambda^n a^n$. For the remaining pixels, which do not contain random points, the contribution to the probability is $(1 - \lambda a)^m$ where m is the number of these pixels. Since m is large and a is small, the second term is close to $e^{-\lambda|W|}$. Thus, ignoring some rescaling,[5] the first term in the likelihood (9.34) is the probability of observing the data points in **x**, and the second term is the probability of **not** observing any other points in the window W.

The likelihood (9.34) is conventionally used in theoretical work, but as mentioned at the end of Section 9.7.1 the constant factor $e^{|W|}$ can be omitted. Thus the likelihood we use in practice is

$$L(\lambda) = \lambda^{n(\mathbf{x})} e^{-\lambda|W|}. \tag{9.35}$$

The maximum likelihood estimate (MLE) of the intensity λ is the value $\hat{\lambda}$ which maximises $L(\lambda)$ defined in (9.35). As before it is easier to work with the logarithm of the likelihood, which in this case is

$$\log L(\lambda) = n(\mathbf{x}) \log \lambda - \lambda |W|. \tag{9.36}$$

The score (derivative of the loglikelihood) is

$$U(\lambda) = U(\lambda; \mathbf{x}) = \frac{\mathrm{d}}{\mathrm{d}\lambda} \log L(\lambda) = \frac{n(\mathbf{x})}{\lambda} - |W|. \tag{9.37}$$

The maximum of the likelihood is attained when this derivative is zero, so the MLE is

$$\hat{\lambda} = \frac{n(\mathbf{x})}{|W|}. \tag{9.38}$$

That is, the maximum likelihood estimate $\hat{\lambda}$ is the average intensity of **x**, a good 'commonsense' estimate of the intensity λ.

Another way to estimate the intensity λ would have been to use the *'method of moments'*. We would equate the observed number of points, $n(\mathbf{x})$, to the theoretically expected number of points, $\mathbb{E}n(\mathbf{X} \cap W) = \lambda|W|$, and solve the equation $\lambda|W| = n(\mathbf{x})$ for λ, yielding $\widehat{\lambda} = n(\mathbf{x})/|W|$. In this case, the method-of-moments estimate agrees with the MLE.

[5] The point process likelihood is conventionally measured relative to the probability for a Poisson process of rate 1. To do this we divide the value obtained above, $\lambda^n a^n e^{-\lambda|W|}$, by the corresponding value for a Poisson process of intensity $\lambda = 1$, namely $a^n e^{-|W|}$ (which is a constant only depending on data, not the parameter), finally yielding $\lambda^n e^{(1-\lambda)|W|}$.

SAMPLE

9.7.3 Maximum likelihood for general Poisson process

Maximum likelihood estimation for Poisson point processes is treated extensively in [565, 396].

9.7.3.1 General form of likelihood

Now consider a general, inhomogeneous Poisson process model, governed by a parameter $\boldsymbol{\theta}$. This model states that the intensity is $\lambda_{\boldsymbol{\theta}}(u)$ where the value of $\boldsymbol{\theta}$ is to be estimated. The likelihood for $\boldsymbol{\theta}$ is

$$L(\boldsymbol{\theta}) = L(\boldsymbol{\theta};\mathbf{x}) = \lambda_{\boldsymbol{\theta}}(x_1)\lambda_{\boldsymbol{\theta}}(x_2)\dots\lambda_{\boldsymbol{\theta}}(x_n)\exp\left(\int_W (1-\lambda_{\boldsymbol{\theta}}(u))\,\mathrm{d}u\right) \tag{9.39}$$

where $x_1,\dots,x_n$ are the points of $\mathbf{x}$. This can be derived by pixel approximation as described above. This probability density *does* depend on the locations of the data points x_i, because the intensity function $\lambda_{\boldsymbol{\theta}}(u)$ makes some locations more likely than others. Therefore the data on spatial locations x_i, and not just the total number of points, are needed for model-fitting.

The loglikelihood for $\boldsymbol{\theta}$ is

$$\log L(\boldsymbol{\theta}) = \sum_{i=1}^{n} \log \lambda_{\boldsymbol{\theta}}(x_i) - \int_W \lambda_{\boldsymbol{\theta}}(u)\,\mathrm{d}u \tag{9.40}$$

where we have omitted the constant term $\int_W 1\,\mathrm{d}u = |W|$.

The MLE $\widehat{\boldsymbol{\theta}}$ is usually not a simple function of the data, and must be computed by maximising the likelihood numerically. In general, with no assumptions on the way $\lambda_{\boldsymbol{\theta}}(u)$ depends on $\boldsymbol{\theta}$, the likelihood function might behave poorly, and might not have a unique maximum. Further analysis depends on assuming more about the intensity model.

9.7.3.2 Maximum likelihood for baseline model

One Poisson model that is easy to analyse is that in which the intensity is an unknown multiple of a known baseline (9.3). The loglikelihood (9.40) is

$$\log L(\theta) = \sum_{i=1}^{n} \log(\theta\, b(x_i)) - \int_W \theta\, b(u)\,\mathrm{d}u = n\log\theta + \sum_{i=1}^{n}\log b(x_i) - \theta\int_W b(u)\,\mathrm{d}u. \tag{9.41}$$

The loglikelihood is differentiable with respect to θ for fixed $\mathbf{x}$, even if $b(u)$ is not a continuous function. The score is

$$U(\theta) = \frac{\mathrm{d}}{\mathrm{d}\theta}\log L = \frac{n(\mathbf{x})}{\theta} - \int_W b(u)\,\mathrm{d}u. \tag{9.42}$$

If there are no constraints on θ, the maximum likelihood estimate (MLE) of θ is the solution of the score equation $U(\theta) = 0$, which is

$$\widehat{\theta} = \frac{n(\mathbf{x})}{\int_W b(u)\,\mathrm{d}u}. \tag{9.43}$$

This is also the method-of-moments estimate, because under the baseline model, the expected total number of points is

$$\mathbb{E}_\theta[n(\mathbf{X})] = \int_W \lambda_\theta(u)\,\mathrm{d}u = \theta\int_W b(u)\,\mathrm{d}u$$

so that $\hat{\theta}$ is the solution of $n(\mathbf{x}) = \mathbb{E}_\theta[n(\mathbf{X})]$.

SAMPLE

9.7.4 Maximum likelihood for loglinear Poisson models

9.7.4.1 Loglinear models

For the vast majority of Poisson models treated in this book, the intensity is a **loglinear** function of the parameters:

$$\lambda_{\boldsymbol{\theta}}(u) = \exp(B(u) + \boldsymbol{\theta}^\top \mathbf{Z}(u)) \tag{9.44}$$

where $B(u)$ is a known function (the 'offset' or 'log baseline'), $\boldsymbol{\theta} = (\theta_1, \dots, \theta_p)$ is the vector of parameters, $\mathbf{Z}(u) = (Z_1(u), \dots, Z_p(u))$ is a vector of covariate functions, and

$$\boldsymbol{\theta}^\top \mathbf{Z}(u) = \theta_1 Z_1(u) + \cdots + \theta_p Z_p(u).$$

Note that the model implies that the *logarithm* of the intensity is a linear function *of the parameters*:

$$\log \lambda_{\boldsymbol{\theta}}(u) = B(u) + \boldsymbol{\theta}^\top \mathbf{Z}(u). \tag{9.45}$$

The functions B and $Z_1, \dots, Z_p$ could be spatially varying in any fashion, so this is a very wide and flexible class of models.

The loglinear intensity model has several advantages. The intensity of a point process must be greater than or equal to zero, and this is always satisfied by the loglinear model, regardless of the value of $\boldsymbol{\theta}$ and the values of the functions B and $Z_1, \dots, Z_p$, because of the exponent in (9.44). In statistical theory the logarithm is the 'canonical' transformation of the mean for a Poisson model, and this confers many advantages in theory and practice.

9.7.4.2 Likelihood for loglinear model

For the loglinear intensity the loglikelihood takes the form

$$\log L(\boldsymbol{\theta}) = \sum_{i=1}^{n} B(x_i) + \boldsymbol{\theta}^{\mathsf{T}} \sum_{i=1}^{n} \mathbf{Z}(x_i) - \int_W \exp(B(u) + \boldsymbol{\theta}^{\mathsf{T}} \mathbf{Z}(u))\, \mathrm{d}u. \tag{9.46}$$

This model is a canonically parametrised exponential family [80, p. 113], [416, pp. 23–24]. The loglikelihood (9.46) is a concave function of the parameter $\boldsymbol{\theta}$, and is differentiable with respect to $\boldsymbol{\theta}$, even if the functions B and Z_j are not continuous. If the matrix

$$M = \int_W \mathbf{Z}(u)\, \mathbf{Z}(u)^{\mathsf{T}}\, \mathrm{d}u$$

is positive definite, then the model is identifiable. If the data are such that $\sum_i Z_j(x_i) \neq 0$ for all j, the MLE exists and is unique. Unless there are further constraints on $\boldsymbol{\theta}$, the MLE is the solution of the *score equations* $\mathbf{U}(\boldsymbol{\theta}) = 0$, where the score function is

$$\mathbf{U}(\boldsymbol{\theta}) = \mathbf{U}(\boldsymbol{\theta}; \mathbf{x}) = \sum_{i=1}^{n(\mathbf{x})} \mathbf{Z}(x_i) - \int_W \mathbf{Z}(u) \lambda_{\boldsymbol{\theta}}(u)\, \mathrm{d}u. \tag{9.47}$$

The score is a vector $\mathbf{U}(\boldsymbol{\theta}; \mathbf{x}) = (U_1(\boldsymbol{\theta}; \mathbf{x}), ..., U_p(\boldsymbol{\theta}; \mathbf{x}))$ with components

$$U_j(\boldsymbol{\theta}; \mathbf{x}) = \sum_{i=1}^{n(\mathbf{x})} Z_j(x_i) - \int_W Z_j(u) \lambda_{\boldsymbol{\theta}}(u)\, \mathrm{d}u$$

for $j = 1, \dots, p$. The integral in (9.47) is the Laplace transform of $\mathbf{Z}$, which is generally intractable, so that in general the score equations cannot be solved analytically.

SAMPLE

9.7.4.3 Accuracy of maximum likelihood estimate

In many statistical models, as the sample size increases, the maximum likelihood estimator $\widehat{\boldsymbol{\theta}}$ follows a standard pattern of behaviour. It is a consistent estimator (it converges in probability to the true answer $\boldsymbol{\theta}$), and it asymptotically follows a normal distribution, with mean equal to the true parameter vector $\boldsymbol{\theta}$, and variance ${\boldsymbol{I}_{\boldsymbol{\theta}}}^{-1}$, where $\boldsymbol{I}_{\boldsymbol{\theta}}$ is the *Fisher information* matrix. The Fisher information is defined as the variance-covariance matrix of the score:

$$\boldsymbol{I}_{\boldsymbol{\theta}} = \mathrm{var}_{\boldsymbol{\theta}}\left[\mathbf{U}(\boldsymbol{\theta};\mathbf{X})\right] = \mathbb{E}_{\boldsymbol{\theta}}\left[\mathbf{U}(\boldsymbol{\theta};\mathbf{X})\mathbf{U}(\boldsymbol{\theta};\mathbf{X})^{\top}\right] = \mathbb{E}_{\boldsymbol{\theta}}\left[-\frac{\partial}{\partial\boldsymbol{\theta}}\mathbf{U}(\boldsymbol{\theta};\mathbf{X})\right] \tag{9.48}$$

where $\mathbb{E}_{\boldsymbol{\theta}}$ and $\mathrm{var}_{\boldsymbol{\theta}}$ denote the mean and variance when the true parameter value is $\boldsymbol{\theta}$. The negative Hessian matrix

$$H(\boldsymbol{\theta};\mathbf{x}) = -\frac{\partial}{\partial\boldsymbol{\theta}}\mathbf{U}(\boldsymbol{\theta};\mathbf{x}) \tag{9.49}$$

when evaluated at $\boldsymbol{\theta} = \widehat{\boldsymbol{\theta}}$ is called the *observed information*, while the Fisher information is sometimes called the *expected information*.

For the loglinear Poisson point process model, asymptotic distribution theory is available [565, 396, 34] and the results conform to the pattern described above. Under suitable conditions,[6] the MLE $\hat{\boldsymbol{\theta}}$ is consistent, asymptotically normal, and asymptotically efficient in a 'large domain' limiting regime [565, Theorem 11, pp. 135–136], [396, Thm. 2.4, p. 51], [393]. The Hessian is

$$H(\boldsymbol{\theta};\mathbf{x}) = \int_W \mathbf{Z}(u)\mathbf{Z}(u)^{\mathsf{T}}\lambda_{\boldsymbol{\theta}}(u)\,\mathrm{d}u. \tag{9.50}$$

Since the Hessian does not depend on the data $\mathbf{x}$, it is equal to the Fisher information $\boldsymbol{I}_{\boldsymbol{\theta}}$. The Fisher information is a matrix with entries (on row i, column j)

$$(\boldsymbol{I}_{\boldsymbol{\theta}})_{ij} = \int_W Z_i(u)Z_j(u)\lambda_{\boldsymbol{\theta}}(u)\,\mathrm{d}u.$$

This is the basis of calculations of standard error and confidence intervals for Poisson models in `spatstat`. Asymptotic theory is also available [565, 396] to support the likelihood ratio test.

The integral in the loglinear Poisson process likelihood (9.46) is the Laplace transform of the covariate function $\mathbf{Z}$, which is not usually available in closed form. Consequently, it is not usually possible to find an exact analytic solution for the maximum likelihood estimate. Some form of numerical approximation is required. Strategies are presented in the next sections.

9.8 Coarse quadrature approximation∗

The likelihood function (9.40) of a Poisson point process involves an integral over the spatial window. Except in special cases, this means that the likelihood cannot be computed exactly, but must be approximated numerically.

A good strategy is to set up the approximation so that that the approximate likelihood function is equivalent to the likelihood of another, simpler, statistical model which we know how to handle. Then using statistical techniques for the simpler model, we can compute approximate parameter

[6]The 'suitable conditions' are essentially that every entry in the Fisher information matrix should be large, and that regularity conditions hold.

∗ Starred sections contain advanced material, and can be skipped by most readers.

SAMPLE

estimates for the original, complex model. This strategy has been used in statistical science since earliest times.

For a point process model, approximation of the likelihood converts the point process into a *regression* model. Lewis [419] and Brillinger [124, 126, 125] showed that the likelihood of a general point process in one-dimensional time, or a Poisson point process in higher dimensions, can be usefully approximated by the likelihood of logistic regression for the discretised process. Asymptotic equivalence was established in [101]. This makes it practicable to fit spatial Poisson point process models of general form to point pattern data [90, 158, 50, 51] by enlisting efficient and reliable software already developed for generalized linear models. The approximation has effectively reduced the problem to a standard statistical model-fitting task. Approximation of a stochastic process by a generalized linear model is now commonplace in applied statistics [424, 425, 427, 426].

9.8.1 Berman-Turner device

Numerical quadrature is a simple and efficient computational strategy for numerical integration, in which the integral $\int_W f(u)\,\mathrm{d}u$ of some function f is approximated by a weighted sum $\sum_j w_j f(u_j)$ of values of the function at a finite list of 'quadrature points' u_j which have 'quadrature weights' w_j.

9.8.1.1 Berman-Turner quadrature

Berman and Turner [90] developed a numerical quadrature method for approximate maximum likelihood estimation for an inhomogeneous Poisson point process. Suppose we approximate the integral in (9.40) by a finite sum using any quadrature rule,

$$\int_W \lambda_{\boldsymbol\theta}(u)\,\mathrm{d}u \approx \sum_{j=1}^{m} \lambda_{\boldsymbol\theta}(u_j)\, w_j \tag{9.51}$$

where $u_j,\ j = 1,\dots,m$ are points in W and $w_j > 0$ are quadrature weights summing to $|W|$. This yields an approximation to the loglikelihood,

$$\log L(\boldsymbol\theta) \approx \sum_{i=1}^{n(\mathbf{x})} \log \lambda_{\boldsymbol\theta}(x_i) - \sum_{j=1}^{m} \lambda_{\boldsymbol\theta}(u_j)\, w_j. \tag{9.52}$$

Berman and Turner observed that if the list of points $\{u_j, j = 1,\dots,m\}$ *includes all the data points* $\{x_i, i = 1,\dots,n\}$, then we can rewrite (9.52) as a sum over quadrature points:

$$\log L(\boldsymbol\theta) \approx \sum_{j=1}^{m} (I_j \log \lambda_j - w_j \lambda_j) \tag{9.53}$$

where $\lambda_j = \lambda_{\boldsymbol\theta}(u_j)$ and

$$I_j = \begin{cases} 1 & \text{if } u_j \text{ is a data point} \\ 0 & \text{if } u_j \text{ is a dummy point.} \end{cases} \tag{9.54}$$

Berman and Turner [90] re-expressed this as

$$\log L(\boldsymbol\theta) \approx \sum_{j=1}^{m} (y_j \log \lambda_j - \lambda_j)\, w_j \tag{9.55}$$

where $y_j = I_j / w_j$, and noted that the right side of (9.55), for fixed $\mathbf{x}$, is formally equivalent to the weighted loglikelihood of independent Poisson variables $Y_j \sim \text{Poisson}(\lambda_j)$ taken with weights w_j. The expression (9.55) can therefore be maximised using standard software for fitting generalised linear models [458].

SAMPLE

Later it was pointed out [459] that (9.53) is the *unweighted* loglikelihood of independent Poisson variables $I_j \sim \text{Poisson}(w_j \lambda_j)$.

The main attraction of the Berman-Turner device is that the use of standard statistical packages rather than *ad hoc* software confers great advantages in applications. Modern statistical packages have a convenient notation for statistical models [7, 142, 669] which makes it very easy to specify and fit a wide variety of models. Algorithms in the package may allow one to fit very flexible model terms such as the smooth functions in a generalised additive model [327]. Interactive software allows great freedom to reanalyse the data. The fitting algorithms are typically more reliable and stable than in homegrown software. This approach is the basis of `spatstat`.

In summary, the procedure is as follows: (1) generate a set of dummy points, and combine it with the data points x_i to form the set of quadrature points u_j; (2) compute the quadrature weights w_j; (3) form the indicators I_j as in (9.54) and calculate $y_j = I_j / w_j$; (4) compute the (possibly vector) values $\mathbf{z}_j = \mathbf{Z}(u_j)$ of the covariates at each quadrature point; (5) invoke standard model-fitting software, specifying that the model is a loglinear Poisson regression $\log \lambda_j = \boldsymbol{\theta}^\mathsf{T} \mathbf{z}_j$, to be fitted to the responses y_j and covariate values $\mathbf{z}_j$, with weights w_j; (6) the coefficient estimates returned by the software give the (approximate) MLE $\widehat{\boldsymbol{\theta}}$ of $\boldsymbol{\theta}$.

In the `spatstat` implementation, step (5) is performed by the standard R model-fitting functions `glm` or `gam`. The estimates of standard errors returned by `glm` are also valid for the Poisson point process, because both the weighted loglinear Poisson regression and the loglinear Poisson point process model are canonically parametrised: they have the property that the Fisher information is equal to the negative Hessian of the loglikelihood, The negative Hessians of the two models are approximately equal by (9.55). Additionally `glm` returns the deviance D of the fitted model; this is related to the loglikelihood of the fitted model by

$$-\log L(\widehat{\boldsymbol{\theta}}; \mathbf{x}) = \frac{D}{2} + \sum_{j=1}^{m} I_j \log w_j + n(\mathbf{x}) \tag{9.56}$$

where the sum is effectively over data points only.

Conveniently, the null model $\lambda_j \equiv \lambda$ in the weighted loglinear Poisson regression corresponds to the uniform Poisson point process with intensity λ. The MLE is $\widehat{\lambda} = n(\mathbf{x}) / \sum_j w_j = n(\mathbf{x})/|W|$ with corresponding loglikelihood $\log L(\widehat{\lambda}) = n(\mathbf{x}) \left[\log n(\mathbf{x}) - \log |W| - 1\right]$.

Note that this formulation assumes $\lambda(u)$ is positive everywhere. Zero values are also permissible, provided the set of zeroes does not depend on $\boldsymbol{\theta}$. Thus we formally allow negative infinite values for $\mathbf{Z}(u)$. In the approximation (9.52) all points u_j with $\lambda(u_j) = 0$ will be dummy points. Their contribution is zero and so they should be omitted in all contexts.

9.8.1.2 Design of quadrature schemes

Berman and Turner [90] used the Dirichlet tessellation or Voronoï diagram [521] to generate quadrature weights. The data points are augmented by a list of dummy points; then the Dirichlet tessellation of the combined set of points is computed as sketched in the left panel of Figure 9.13. The quadrature weight w_j associated with a (data or dummy) point u_j is the area of the corresponding Dirichlet tile.

It is computationally cheaper to use the *counting weights* proposed in [50]. In the simplest form we assign equal weight to each quadrature point, namely $w_j = |W|/m$. In general we divide the window into 'tiles', and all quadrature points within a given tile T receive the same weight $w_j = |T|/k$ where k is the number of quadrature points in T. The right panel of Figure 9.13 shows an example where W is partitioned into equal rectangular tiles.

Other choices of quadrature scheme have been explored in [50, 412].

SAMPLE

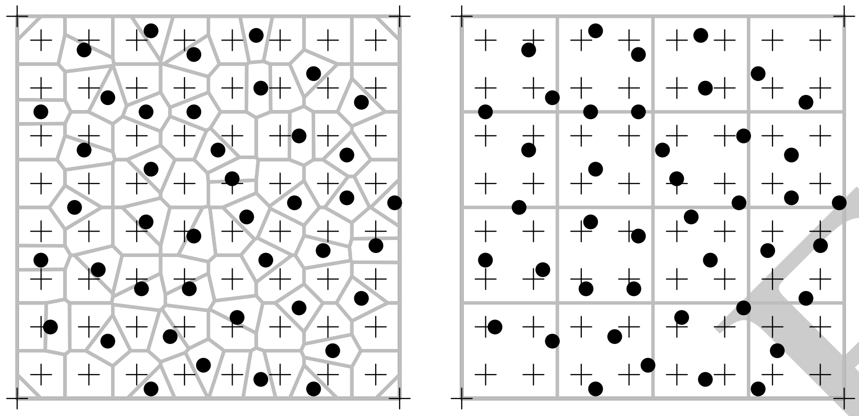


Figure 9.13. *Quadrature schemes for a point pattern dataset. Data points (•), dummy points (+), and tile boundaries (grey lines).* Left: *Dirichlet tiles and weights.* Right: *rectangular tiles, 'counting weights'. Coarsely spaced example for illustration only.*

9.8.1.3 Quadrature schemes in `spatstat`

The `spatstat` function `ppm` fits a point process model to an observed point pattern. By default, it uses Berman-Turner quadrature approximation.

In normal use, the quadrature scheme is generated automatically from the data point pattern, so the user does not need to think about it. However, the default quadrature scheme may give unsatisfactory results in some cases. The default rules for building a quadrature scheme are designed so that `ppm` will execute quickly, rather than guaranteeing a highly accurate result.

Users who wish to override the defaults may modify the rules for building a quadrature scheme, or may even provide their own quadrature scheme.

Quadrature schemes are created by the function `quadscheme`:

```
quadscheme(data, dummy, ..., method="grid")
```

The arguments `data` and `dummy` specify the data and dummy points, respectively.

`quadscheme(X)`	default for data pattern `X`
`quadscheme(X, nd=128)`	dummy points in 128×128 grid
`quadscheme(X, eps=0.1)`	dummy point spacing 0.1 units
`quadscheme(X, random=TRUE)`	stratified random dummy points
`quadscheme(X, quasi=TRUE)`	quasirandom dummy points
`quadscheme(X, D)`	data `X`, dummy `D`

Table 9.6. *Typical options for controlling the dummy points in a quadrature scheme (options are passed to* `default.dummy`*).*

For the dummy points, there is a sensible default, provided by `default.dummy(X)`. Table 9.6 shows some of the important options for modifying the default dummy pattern. By default, the dummy points are arranged in a rectangular grid; recognised arguments include `nd` (the number of grid points in the horizontal and vertical directions) and `eps` (the spacing between dummy points). If `random=TRUE`, a systematic random (also called stratified random) pattern of dummy points is generated instead. If `quasi=TRUE`, a quasirandom pattern of dummy points is generated.

Alternatively the dummy point pattern may be specified arbitrarily and given in any format recognised by `as.ppp`. There are functions for creating special dummy patterns including `corners`,

SAMPLE

`gridcentres`, `stratrand`, and `spokes`. It is also possible to generate dummy points according to a quasirandom scheme, using the functions `Halton` and `Hammersley`. Quasirandom patterns fill the space uniformly but without any obvious regularities: they may be preferable to regular grids.

`quadscheme(X, ..., method="d")`	specify Dirichlet weights
`quadscheme(X, ..., ntile=8)`	8×8 array of counting tiles

Table 9.7. *Typical options for controlling the quadrature weights.*

The **quadrature weights** are determined by further arguments to `quadscheme`. If `method = "grid"` (the default) the window is divided into an `ntile[1]` by `ntile[2]` grid of rectangular tiles, and the 'counting weights' are applied: the weight for each quadrature point is the area of a tile divided by the number of quadrature points in that tile. By default the values `ntile` and `nd` are the same so all tiles (except the corners) contain exactly one dummy point as illustrated in the right panel of Figure 9.13. If `method="dirichlet"`, the quadrature points (both data and dummy) are used to construct the Dirichlet tessellation. The quadrature weight of each point is the area of its Dirichlet tile inside the quadrature region.

A quadrature scheme (consisting of the original data point pattern, an additional pattern of dummy points, and a vector of quadrature weights for all these points) is represented by an object of class `"quad"`. In principle, the user could create one of these objects from scratch, using the creator function `quad`. More details of the software implementation of `ppm` are given in [50, 51, 52] and particularly [54].

9.8.1.4 Gorilla nests example

In Section 9.3.4.1 we fitted a simple model to the `gorillas` data in which the intensity is constant inside each region defined by vegetation type:

```
> ppm(gor ~ vege, data=gex)
Nonstationary Poisson process
Log intensity:  ~vege
             Estimate   S.E. CI95.lo CI95.hi Ztest   Zval
(Intercept)    2.3238 0.1060  2.1161  2.5316   *** 21.923
vegeColo       2.0817 0.5870  0.9312  3.2322   ***  3.546
vegeGras      -0.7721 0.2475 -1.2571 -0.2871    ** -3.120
vegePrim       2.1148 0.1151  1.8892  2.3403   *** 18.373
vegeSeco       1.1341 0.2426  0.6586  1.6096   ***  4.675
vegeTran       1.6151 0.2647  1.0964  2.1339   ***  6.102
```

Fitting this model by maximum likelihood is equivalent to estimating the intensities using quadrat counts. To check this, we convert the pixel image to a tessellation and apply quadrat counting:

```
> vt <- tess(image=gex$vege)
> intensity(quadratcount(gor, tess=vt))
tile
  Dist   Colo   Gras   Prim   Seco   Tran
10.201 69.155  4.781 84.012 32.651 50.922
```

Again there are discrepancies due to discretisation of the integral in `ppm` in the Berman-Turner technique. Better agreement can be obtained by increasing the density of dummy points, for example, using the parameter `nd`:

```
> fitveg2 <- ppm(gor ~ vege - 1, data=gex, nd=256)
> exp(coef(fitveg2))
```

SAMPLE

```
vegeDist vegeColo vegeGras vegePrim vegeSeco vegeTran
  10.188   69.040    4.786   83.807   32.329   51.199
```

Exact agreement (up to numerical rounding error) can be obtained by using a quadrature scheme with one point at each pixel of the covariate image. This is constructed by the command `pixelquad`:

```
> Q <- pixelquad(gor, gex$vege)
> fitveg3 <- ppm(Q ~ vege - 1, data=gex)
> exp(coef(fitveg3))
vegeDist vegeColo vegeGras vegePrim vegeSeco vegeTran
  10.201   69.155    4.781   84.012   32.651   50.922
```

9.8.1.5 Covariate values known only at some locations

In practice, covariate values may only be available at a small set of spatial sample locations. For example, measuring a covariate at a given location may require digging a bore hole at that location, or physically extracting material, or performing a complicated chemical assay. It may require the presence of a sensor at that location, such as a weather station or a ship. Some astronomical measurements require the fortuitous presence of a light source behind the region of interest.

One strategy for dealing with this situation is to estimate the covariate values at other locations by interpolating between the observed values. One could use the Nadaraya-Watson smoother (Section 6.9), or kriging interpolation, or many other methods.

An alternative strategy is to build a coarse quadrature scheme using only the available sample locations. This is more faithful to the observations, and avoids some potential artefacts of the smoothing. If `X` is the observed point pattern and `D` the pattern of locations where covariate values have been observed, then `quadscheme(X, D)` will build a quadrature scheme using these points. Additional arguments determine the weighting scheme: we recommend the Dirichlet weights. The corresponding covariate values should be organised in a data frame, with one column for each covariate, and one row for each quadrature point (the points of `X` are listed first, followed by the points of `D`). This data frame can then be passed to `ppm` as the argument `data`.

Just to demonstrate that this is possible, we take the gorilla nests data analysed above, and pretend that the land condition was only recorded at certain locations, namely the gorilla nest sites `gor` and some arbitrarily chosen sample locations `samp`:

```
> dfdata <- as.data.frame(lapply(gex, "[", i=gor))
> samp <- rSSI(0.5, 42, Window(gor))
> dfsamp <- as.data.frame(lapply(gex, "[", i=samp))
```

The previous model can be fitted to these data by building a quadrature scheme based on the available locations:

```
> G <- quadscheme(gor, samp, method="d")
> df <- rbind(dfdata, dfsamp)
> fitdf <- ppm(G ~ vege - 1, data=df)
> exp(coef(fitdf))
vegeDist vegeColo vegeGras vegePrim vegeSeco vegeTran
   9.765   24.175    5.468   93.612   20.337   26.731
```

We emphasise that covariate values must be available at *all data points* and at *some non-data points*. If covariate values have only been observed at the data points, it is effectively impossible to fit a point process model in which the presence or absence of points depends on the covariate value.

SAMPLE

9.9 Fine pixel approximation[∗]

9.9.1 Pixel counts

Another quadrature strategy for approximating the Poisson process likelihood is to divide the window W into small pixels of equal area a. The integral over the window W is then approximated by a sum over pixels:

$$\int_W \lambda_{\boldsymbol{\theta}}(u)\,\mathrm{d}u \approx \sum_j \lambda_{\boldsymbol{\theta}}(u_j)a \tag{9.57}$$

where u_j is the centre of the jth pixel. We also discard the exact locations of the data points, and effectively move each data point to the centre of the pixel which contains it. Thus we approximate the sum over data points by a sum over pixels,

$$\sum_i \log \lambda_{\boldsymbol{\theta}}(x_i) \approx \sum_j n_j \log \lambda_{\boldsymbol{\theta}}(u_j) \tag{9.58}$$

where n_j is the number of data points falling in the jth pixel. Collecting (9.57) and (9.58), we approximate the true loglikelihood (9.40) by

$$\log L(\boldsymbol{\theta}) \approx \sum_j [n_j \log \lambda_{\boldsymbol{\theta}}(u_j) - \lambda_{\boldsymbol{\theta}}(u_j)a] = \sum_j (n_j \log \lambda_j - \lambda_j a) \tag{9.59}$$

where $\lambda_j = \lambda_{\boldsymbol{\theta}}(u_j)$. The adequacy of this approximation is discussed in Section 9.9.4.

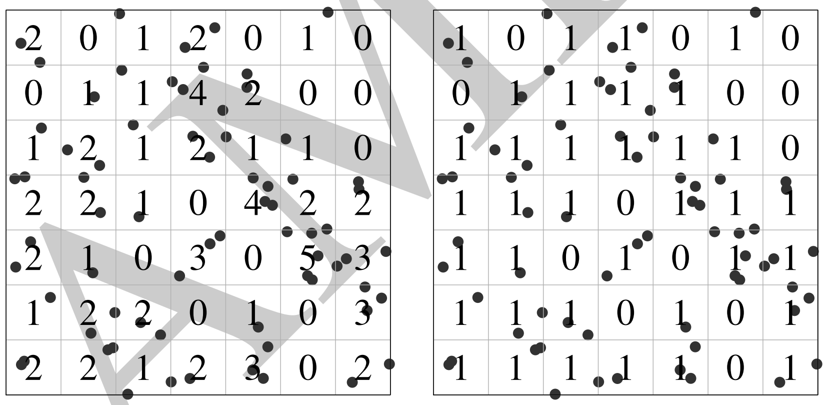


Figure 9.14. *Pixel counts (*Left*) and presence-absence indicators (*Right*) for the same point pattern.*

The right-hand side of (9.59) has the same form as the loglikelihood of independent Poisson random variables N_j with means $a\lambda_j$. This was to be expected, because the pixel counts N_j *are* independent Poisson random variables, and $a\lambda_j$ is an approximation to the true mean of N_j.

For a loglinear Poisson point process model (9.44), we have

$$\lambda_j = \lambda_{\boldsymbol{\theta}}(u_j) = \exp(B(u_j) + \boldsymbol{\theta}^\top \mathbf{Z}(u_j)) = \exp(b_j + \boldsymbol{\theta}^\top \mathbf{z}_j) \tag{9.60}$$

where $b_j = B(u_j)$ and $\mathbf{z}_j = \mathbf{Z}(u_j)$. So the right-hand side of (9.59) is the loglikelihood of independent Poisson random variables N_j with means

$$\mu_j = a\lambda_j = \exp(\log a + b_j + \boldsymbol{\theta}^\top \mathbf{z}_j) = \exp(\log o_j + \boldsymbol{\theta}^\top \mathbf{z}_j) \tag{9.61}$$

∗ Starred sections contain advanced material, and can be skipped by most readers.

SAMPLE

where $o_j = \log a + b_j$. This is *loglinear Poisson regression* with regression covariates $\mathbf{z}_j$ and offset $o_j = b_j + \log a$.

A practical strategy for fitting a loglinear Poisson point process model is therefore: (1) divide the window W into a fine grid of pixels of area a; (2) count the number n_j of data points falling in each pixel j; (3) evaluate the offset term $o_j = \log a + B(u_j)$ and the covariate vector $\mathbf{z}_j = \mathbf{Z}(u_j)$ at the centre u_j of each pixel j; (4) use standard statistical software (such as the `glm` function in R) to fit (by maximum likelihood) a loglinear Poisson regression model with responses n_j, regression covariates $\mathbf{z}_j$, and offsets o_j; (5) the fitted coefficients $\hat{\boldsymbol{\theta}}$ for the loglinear Poisson regression are the approximate maximum likelihood estimates $\hat{\boldsymbol{\theta}}$ for the loglinear Poisson point process model (9.44).

9.9.2 Pixel presence-absence indicators

A further simplification is to replace the counts n_j by *presence-absence indicators* $I_j = \mathbf{1}\{n_j > 0\}$ which tell us whether data points were present ($I_j = 1$) or absent ($I_j = 0$) in each pixel j. See the right panel of Figure 9.14. The presence-absence indicators are independent random variables, so their loglikelihood is the binomial

$$\log L = \sum_j (I_j \log p_j + (1 - I_j)\log(1 - p_j)) \tag{9.62}$$

where p_j denotes the probability that there are any data points in pixel j. Using the same approximation (9.57) we have $p_j = \mathbb{P}\{I_j = 1\} = \mathbb{P}\{N_j \geq 1\} = 1 - e^{-\mu_j} = e^{-a\lambda_j}$. For a loglinear Poisson point process model (9.44), substituting (9.61) we get $p_j = 1 - \exp(-\exp(o_j + \boldsymbol{\theta}^\top \mathbf{z}_j))$, and inverting this relationship gives

$$\log(-\log(1 - p_j)) = o_j + \boldsymbol{\theta}^\top \mathbf{z}_j \tag{9.63}$$

where the offset o_j is

$$o_j = b_j + \log a. \tag{9.64}$$

The transformation $\log(-\log(1 - p))$ is called the *complementary log-log* link. Thus, the presence-absence indicators follow a binary *complementary log-log regression* with offset o_j and regression covariates $\mathbf{z}_j$ as before.

Thus, the correct way to analyse the presence-absence indicator data from a loglinear Poisson point process is to fit a complementary log-log regression, using the offset (9.64) to account for pixel size. This ensures that estimates of $\boldsymbol{\theta}$ obtained using pixel grids of different size are exactly compatible. The parameter estimates $\widehat{\boldsymbol{\theta}}$ obtained from this procedure are estimates of the parameters of the loglinear Poisson point process model (9.44).

9.9.3 Logistic regression in a fine pixel grid

There is a further simplification when the pixels are so small that they have negligible chance of containing more than one data point. Given that $N_j \leq 1$ for all pixels j, the conditional probability that $N_j = 1$ is

$$p_j^* = \mathbb{P}\{N_j = 1 \mid N_j \leq 1\} = \frac{\mathbb{P}\{N_j = 1\}}{\mathbb{P}\{N_j \leq 1\}} = \frac{\mathbb{P}\{N_j = 1\}}{\mathbb{P}\{N_j = 0\} + \mathbb{P}\{N_j = 1\}}.$$

Using the formula for the Poisson probabilities (5.4) this is

$$p_j = \frac{\mu_j \exp(-\mu_j)}{\exp(-\mu_j) + \mu_j \exp(-\mu_j)} = \frac{\mu_j}{1 + \mu_j}$$

and similarly the conditional probability that $N_j = 0$ is $1 - p_j^* = 1/(1 + \mu_j)$. We find that $p_j^*/(1 - p_j^*) = \mu_j$ so that the *odds* of presence (that is, the ratio of the presence probability divided by the

SAMPLE

absence probability) is equal to μ_j. For a loglinear Poisson process, substituting (9.61) and taking the logarithm gives

$$\log\left(\frac{p_j^*}{1-p_j^*}\right) = o_j + \boldsymbol{\theta}^\top \mathbf{z}_j \tag{9.65}$$

saying that the *log odds* of presence is equal to a linear function of the parameters. The transformation $\log(p/(1-p))$ is called the *logistic* link, and (9.65) states that the presence-absence indicators follow a *logistic regression* with offset o_j as in (9.64) and covariates $\mathbf{z}_j$ as before. Thus, *if the probability of getting more than one point in a pixel is negligible, the presence-absence indicators I_j approximately satisfy a logistic regression with the same linear predictor*.

For computation and for statistical inference, logistic regression has some practical advantages over complementary log-log regression, because the logistic link is the 'canonical link' for binary regression [458]. In logistic regression the estimated variance-covariance matrix of $\widehat{\boldsymbol{\theta}}$ depends only on $\widehat{\boldsymbol{\theta}}$, whereas for complementary log-log regression it depends on $\mathbf{x}$ as well.

Pixel-based logistic regression for point events was pioneered in geology by F.P. Agterberg [4] on the suggestion of J.W. Tukey [658]. It was later independently rediscovered in archaeology [604, 326, 397, 398] and is now a standard technique in GIS applications [116].

Despite its popularity, pixel logistic regression does not seem to be universally well understood. Some writers describe it as a 'nonparametric' technique [399, p. 24]. The interpretation of the fitted parameters is widely held to be obscure [698, p. 175] and is typically based only on the *sign* of the slope parameters, that is, parameters other than the intercept [288, pp. 405–407].

Contrary to these statements, there is a clear physical meaning for the model parameters of (pixel-based) logistic regression when pixels are sufficiently small. The model is effectively a Poisson point process with loglinear intensity (9.6): see [34, 691]. This is a relatively simple parametric model. The fitted parameters of the logistic regression have a direct interpretation as the parameters of the Poisson point process (provided the model is fitted using an offset that includes the logarithm of pixel area). The known properties of the Poisson model (Section 9.2.1) enable us to make numerous predictions about quantities of interest, such as the expected number of points in a target region, the probability of exactly k points in a target region, and the distribution of distance from a fixed starting location to the nearest random point. See Section 9.4.3.

9.9.4 The effect of pixel size

The need for small pixels

By converting a spatial point pattern into an array of pixel counts or pixel presence-absence indicators, we have effectively *aggregated* the individual points into groups. Each group consists of the points that fall in a given pixel. Normally the pixels would be so small that at most one point would fall in each pixel (unless of course there are duplicated points). The theory above relies on this approximation (9.58) and the corresponding approximation for the *expected* number of points in the pixel (9.57).

But how small is 'small enough'? This was investigated in [34]. The adequacy of the approximation depends on the spatial regularity of the function $\mathbf{Z}$, spelt out in [34, Theorem 10]. In the most optimistic case, $\mathbf{Z}$ is *constant within each pixel*,

$$\mathbf{Z}(u) = \mathbf{z}_j \text{ for } u \in S_j, \tag{9.66}$$

so that (9.57) and (9.58) are exact equations rather than approximations, and the loglikelihoods (9.59) and (9.62)–(9.64) are the exact loglikelihoods. Covariates of this kind include factor-valued classifications relating to ownership and management, and classifications of terrain into large geological units. In the next best case, $\mathbf{Z}$ is a smooth function.[7] Then (9.57), (9.58), (9.59), and (9.62)–

[7]Here $\mathbf{Z}$ should be a Lipschitz function, i.e. $||\mathbf{Z}(u)-\mathbf{Z}(v)|| \le C||u-v||$ where $C<\infty$ is constant.

SAMPLE

(9.64) are good approximations when the pixels are small. Examples include distance transforms (e.g., $\mathbf{Z}(u)$ is the distance from u to the nearest geological fault), spatial coordinates, and kernel-smoothed geochemical assay values. In the least optimistic case, $\mathbf{Z}$ is a discontinuous function, such as the indicator of a very irregular spatial domain such as a rock outcrop. The approximations (9.59) and (9.62)–(9.64) can then give rise to considerable bias, even for quite small pixel sizes [34].

Replacing pixel counts by presence-absence indicators causes a loss of statistical efficiency (i.e. variance of the estimator increases). A rule-of-thumb [34] is that the relative efficiency (ratio of variances, assuming the same size of data) is $\bar{\mu}/(e^{\bar{\mu}} - 1) \approx 1 - \bar{\mu}/2$ where $\bar{\mu} = \frac{1}{N}\sum \mu_j$ is the average expected number of points per pixel, estimated by $n(\mathbf{X})/N$, the observed average number of data points per pixel. To ensure the loss of efficiency is less than 10%, we would require $\bar{\mu} < 0.2$, which would normally be satisfied in practice. Logistic regression involves a further approximation in which we say there is a negligible probability of getting more than one point in the same pixel, *in any pixel*. This again is not a very onerous condition, as shown in [34].

In summary, the accuracy of the pixel approximation depends mainly on the spatial irregularity of the covariates. Very small pixels may be necessary if the covariates are highly irregular.

Problems with small pixels

If very small pixels are used, practical problems can arise. There will be a very large number of pixels, and hence a large amount of data to handle. Software for logistic regression may not be designed for large amounts of data; for example, computing the likelihood simply by summing all the individual terms in (9.62) could lead to numerical overflow. In a fine pixellation, the overwhelming majority of pixels do not contain a data point, so the overwhelming majority of indicator values I_j are equal to zero. This causes numerical instability (leading to error messages about singular matrices, for example). It also causes some of the associated statistical tools to fail or behave unexpectedly, due to the *Hauck-Donner effect* [330], essentially the breakdown of the delta-method approximation.

One valid strategy for avoiding these problems is to take only a random sample of the zero-pixels (the pixels with $I_j = 0$), as discussed in Section 9.10.

Problems with large pixels

Some researchers choose to use quite large pixels, and apply logistic regression to the presence-absence indicators. This leads to difficult problems related to the aggregation of points into geographical areas. These problems have been studied extensively in epidemiology and in ecological and environmental statistics [253, 686, 682, 681]. Important questions include the equivalence of models fitted using different pixel grids (the 'modifiable area unit problem' [529] or 'change-of-support' [289, 76, 184]), the relation between discrete and continuous spatial models ('ecological fallacy' [583]), and bias due to aggregation over pixels ('ecological bias' [681, 682] or aggregation bias [207, 13]).

An unexpected finding in [34] is that it may be *impossible* to reconcile two spatial logistic regression models that were fitted to the same spatial point pattern data using different pixel grids. Two such models are logically incompatible except in simple cases. Perhaps the easiest way to understand this is to remember that the pixels are artificial. If we fit a logistic regression using pixels of a particular size, we are making an assumption about the underlying random process of points in continuous space. If we now choose pixels of a different size, this corresponds to a different assumption, and in general the two sets of assumptions are logically incompatible. There is no point process in continuous space which satisfies a logistic regression model when it is discretised on *any* pixel grid. The implication is that two research teams who apply spatial logistic regression to the same data, but using different pixel sizes, may obtain results that cannot be reconciled.

SAMPLE

9.9.5 Implementation in `spatstat`

The `spatstat` package provides facilities for performing pixel logistic regression, mainly for teaching and for methodological research. Models are fitted using the command `slrm` (for **s**patial **l**ogistic **r**egression **m**odel, pronounced *'SLURM'*) with a syntax very similar to `ppm`.

```
> fit <- slrm(bei ~ grad, data=bei.extra)
> fit
Fitted spatial logistic regression model
Formula:        bei ~ grad
<environment: 0xcbbae28>
Fitted coefficients:
(Intercept)        grad
     -5.727       6.516
```

The result is a 'fitted spatial logistic regression model' object of class `"slrm"`. Methods for this class include `print`, `plot`, `predict`, `coef`, `fitted`, `update`, `terms`, `formula`, `anova`, and `logLik`. Additionally `step` and `AIC` can be used for model selection.

We reiterate that `slrm` is designed for methodological research rather than day-to-day analysis. For example, it helps to demonstrate that spatial logistic regression is equivalent to fitting a loglinear Poisson point process [34].

9.10 Conditional logistic regression∗

This section describes an alternative strategy for fitting point process models to point pattern data. Instead of using a dense grid of pixels or dummy points, we generate a smaller number of dummy points at *random* locations. Given only the data at these (data plus dummy) locations, a loglinear Poisson point process model becomes a logistic regression model, which we can fit using standard software.

Although there is a loss of statistical efficiency (increase in variance of parameter estimates) due to the sub-sampling, this method avoids some of the biases inherent in quadrature approximations, so it may have better mean square error overall.

9.10.1 Connection between loglinear Poisson and logistic models

Statisticians will be familiar with the connection between loglinear Poisson regression models and logistic models, for example in contingency tables. The basic principle is as follows.

Suppose X apples and Y oranges have fallen from the trees in an orchard, where X and Y are independent, Poisson random variables with means μ and ν. Then the special properties of the Poisson distribution imply various relationships. The total number of fruit $X+Y$ is a Poisson variable with mean $\mu+\nu$. Given that there are $X+Y=n$ fruit in total, each fruit is either an apple with probability p or an orange with probability $1-p$, independently of other fruit, where $p=\mu/(\mu+\nu)$.

Suppose the expected number of apples μ depends on a covariate z through a *loglinear* relationship, $\log\mu = a+bz$, or equivalently, $\mu = e^{a+bz}$. Given there are n fruit in total, the probability that each fruit is an apple is $p=\mu/(\mu+\nu)=e^{a+bz}/(e^{a+bz}+\nu)$, and for an orange, $1-p=\nu/(\mu+\nu)=\nu/(e^{a+bz}+\nu)$. The odds of an apple against an orange are $p/(1-p)=e^{a+bz}/\nu$ so that the logarithm

∗ Starred sections contain advanced material, and can be skipped by most readers.

SAMPLE

of the odds is

$$\log \frac{p}{1-p} = a + bz - \log \nu.$$

That is, the probability p of an apple is related to the covariate z through a *logistic regression* relationship. Thus, *loglinear Poisson regression becomes logistic regression when we condition on the total count*.

9.10.2 Random dummy points and logistic regression

Assume, as before, that the data points come from a Poisson process $\mathbf{X}$ with intensity function $\lambda_{\boldsymbol{\theta}}(u)$ where $\boldsymbol{\theta}$ is to be estimated. Suppose that we generate dummy points at random, according to a Poisson process $\mathbf{D}$ with known intensity function $\delta(u) > 0$, independently of the data points. Combining these two types of points, the superposition $\mathbf{Y} = \mathbf{X} \cup \mathbf{D}$ is again a Poisson point process (by the superposition property, Section 5.3.3) with intensity function $\kappa(u) = \lambda_{\boldsymbol{\theta}}(u) + \delta(u)$. For each point $y_i \in \mathbf{Y}$, let $I_i = \mathbf{1}\{y_i \in \mathbf{X}\}$ be the indicator that equals 1 if y_i was a data point, and 0 if it was a dummy point.

Let us now *condition on* $\mathbf{Y}$, so that we regard the points y_i as fixed locations. The data now consist only of the data/dummy indicators $I_1, \dots, I_m$ where $m = n(\mathbf{X}) + n(\mathbf{D})$ is the total number of (data and dummy) points. Conditional on location, the indicators I_i are independent random variables, with probabilities

$$\mathbb{P}\{I_i = 1\} = p_i = \frac{\lambda_{\boldsymbol{\theta}}(y_i)}{\lambda_{\boldsymbol{\theta}}(y_i) + \delta(y_i)}, \qquad \mathbb{P}\{I_i = 0\} = 1 - p_i = \frac{\delta(y_i)}{\lambda_{\boldsymbol{\theta}}(y_i) + \delta(y_i)}$$

so that the odds are

$$\frac{\mathbb{P}\{I_i = 1\}}{\mathbb{P}\{I_i = 0\}} = \frac{p_i}{1-p_i} = \frac{\lambda_{\boldsymbol{\theta}}(y_i)}{\delta(y_i)}.$$

If the intensity of $\mathbf{X}$ is a loglinear function of the covariate, $\lambda_{\boldsymbol{\theta}}(u) = \exp(B(u) + \boldsymbol{\theta}^\top \mathbf{Z}(u))$, then the log odds are

$$\log \frac{\mathbb{P}\{I_i = 1\}}{\mathbb{P}\{I_i = 0\}} = B(y_i) + \boldsymbol{\theta}^\top \mathbf{Z}(y_i) - \log \delta(y_i). \tag{9.67}$$

The indicators I_i therefore satisfy a *logistic regression*. The loglikelihood given the data $\mathbf{x}$ and a realisation $\mathbf{d}$ of the dummy process is

$$\begin{aligned} \log L(\boldsymbol{\theta}; \mathbf{x}, \mathbf{d}) &= \sum_i I_i \log(p_i) + (1 - I_i) \log(1 - p_i) \\ &= \sum_{u \in \mathbf{x}} \log \frac{\lambda_{\boldsymbol{\theta}}(u)}{\lambda_{\boldsymbol{\theta}}(u) + \delta(u)} + \sum_{u \in \mathbf{d}} \log \frac{\delta(u)}{\lambda_{\boldsymbol{\theta}}(u) + \delta(u)}. \end{aligned} \tag{9.68}$$

That is, if we treat the pattern of data and dummy points $\mathbf{y} = \mathbf{x} \cup \mathbf{d}$ as fixed locations, then the maximum likelihood estimator of the parameters $\boldsymbol{\theta}$ is obtained by fitting a logistic regression (9.67) to the data/dummy indicators I_i.

This approach has many advantages. Estimation can be implemented straightforwardly using standard software for generalised linear models. The loglikelihood (9.68) is a concave function of $\boldsymbol{\theta}$, and conditions for existence and uniqueness of the maximum are well known [616], [38].

9.10.3 Pixel logistic regression with subsampled zero pixels

Another motivation for the same technique comes from pixel logistic regression (Section 9.9). As noted in Section 9.9.4, a fine grid of pixels is needed in order to minimise bias, but this results in a large volume of data, in which most of the presence-absence indicators are zero, causing computational and statistical problems. One strategy is to take a random sub-sample of the 'zero pixels'

SAMPLE

(pixels with presence-absence indicator equal to zero). Suppose each zero-pixel is sampled with probability s independently of other zero-pixels. Each pixel with the presence value 1 is retained. If the data on the full pixel grid satisfy a logistic regression

$$\log \frac{p_j}{1-p_j} = \boldsymbol{\theta}^\top \mathbf{z}_j$$

where p_j is the presence probability for pixel j, then the subsampled data satisfy

$$\log \frac{p_j^*}{1-p_j^*} = \boldsymbol{\theta}^\top \mathbf{z}_j - \log s \tag{9.69}$$

where p_j^* is the presence probability for pixel j given that it has been sampled. Since s is a probability, $\log s$ is negative, so the effect of subtracting $\log s$ is that the log odds of presence are increased in the subsample, as we would expect.

In summary, if we discretise the point pattern data on a fine grid, set up the pixel presence-absence indicators I_j, then *randomly subsample* the zero pixels with retention probability s, a logistic regression is appropriate, with adjustment for the sub-sampling probability as shown in (9.69).

9.10.4 Conditional logistic regression as a point process method

A pivotal question about conditional logistic regression is whether it is statistically reliable. Quadrature methods use a fine grid of pixels or dummy points in an effort to achieve accurate approximation of the likelihood of the Poisson process. However, random sampling can produce unbiased estimates of the score using much fewer function evaluations, at the expense of increased variability due to the randomisation.

Suppose **x** is a point pattern dataset, and **d** is a random pattern of dummy points which we have generated with intensity function $\delta(u)$. Assume as usual that **x** is a realisation of a Poisson process with loglinear intensity. The score is

$$\mathbf{U}(\boldsymbol{\theta}; \mathbf{x}, \mathbf{d}) = \sum_{u \in \mathbf{x}} \frac{\delta(u)\mathbf{Z}(u)}{\lambda_{\boldsymbol{\theta}}(u) + \delta(u)} - \sum_{u \in \mathbf{d}} \frac{\lambda_{\boldsymbol{\theta}}(u)\mathbf{Z}(u)}{\lambda_{\boldsymbol{\theta}}(u) + \delta(u)}. \tag{9.70}$$

By Campbell's formula (6.11) the expectation of the first sum in (9.70) over all outcomes of **X** is

$$\mathbb{E} \sum_{u \in \mathbf{X}} \frac{\delta(u)\mathbf{Z}(u)}{\lambda_{\boldsymbol{\theta}}(u) + \delta(u)} = \mathbb{E} \int_W \frac{\lambda_{\boldsymbol{\theta}}(u)\delta(u)\mathbf{Z}(u)}{\lambda_{\boldsymbol{\theta}}(u) + \delta(u)} \, \mathrm{d}u \tag{9.71}$$

and for the second sum in (9.70) the expectation over all outcomes of **D** is

$$\mathbb{E} \sum_{u \in \mathbf{D}} \frac{\lambda_{\boldsymbol{\theta}}(u, X)\mathbf{Z}(u)}{\lambda_{\boldsymbol{\theta}}(u, X) + \delta(u)} = \mathbb{E} \int_W \frac{\lambda_{\boldsymbol{\theta}}(u)\delta(u)\mathbf{Z}(u)}{\lambda_{\boldsymbol{\theta}}(u) + \delta(u)} \, \mathrm{d}u. \tag{9.72}$$

It follows that $\mathbb{E}_{\boldsymbol{\theta}} \mathbf{U}(\boldsymbol{\theta}; \mathbf{X}, \mathbf{D}) = 0$ where the expectation is taken over both **X** and **D**. The logistic score (9.70) is an unbiased estimating function. This implies that, under reasonable conditions, the estimate of $\boldsymbol{\theta}$ obtained from logistic regression is consistent and asymptotically normal [38].

If we rearrange (9.70) as

$$\mathbf{U}(\boldsymbol{\theta}; \mathbf{x}, \mathbf{d}) = \sum_{u \in \mathbf{x}} \mathbf{Z}(u) - \sum_{u \in \mathbf{x} \cup \mathbf{d}} \frac{\lambda_{\boldsymbol{\theta}}(u)}{\lambda_{\boldsymbol{\theta}}(u) + \delta(u)} \mathbf{Z}(u) \tag{9.73}$$

and apply Campbell's formula to the last term in (9.73), we obtain

$$\mathbb{E} \sum_{u \in \mathbf{X} \cup \mathbf{D}} \frac{\lambda_{\boldsymbol{\theta}}(u)}{\lambda_{\boldsymbol{\theta}}(u) + \delta(u)} \mathbf{Z}(u) = \int_W \mathbf{Z}(u) \lambda_{\boldsymbol{\theta}}(u) \, \mathrm{d}u. \tag{9.74}$$

SAMPLE

Thus, if the last term in (9.73) is replaced by its expectation, the score of the full point process likelihood is obtained. Hence the score of the conditional logistic regression may be viewed as a Monte Carlo approximation of the full maximum likelihood score to which it converges (in mean square) when $\inf_{u \in W} \delta(u) \to \infty$.

It is also possible to use dummy points generated by a point process other than a Poisson process, with appropriate adjustments to the likelihood [38].

9.10.5 Implementation in `spatstat`

If `ppm` is called with the option `method="logi"`, it will fit the point process model using conditional logistic regression.

```
> fitM <- ppm(bei ~ grad, data=bei.extra)
> fitL <- ppm(bei ~ grad, data=bei.extra, method="logi")
> coef(fitM)
(Intercept)        grad
     -5.391       5.022
> coef(fitL)
(Intercept)        grad
     -5.415       5.294
```

Dummy points will be generated at random, by default, but can be generated in several ways, including a deterministic grid. The default is a stratified random pattern, in which the window is divided into equal tiles and each tile contains a fixed number of independent random points.

In the usual case where the left-hand side of the model formula is a point pattern `X`, the function `ppm` will call `quadscheme.logi(X)` to assemble a 'quadrature scheme' object in which all data and dummy points have weight equal to 1.

Alternatively the left-hand side of the model formula may be such a quadrature scheme which the user has constructed by calling `quadscheme.logi(X, ...)`. The user may provide the dummy points by calling `quadscheme.logi(X, D)`, but this would be unusual: normally they are generated randomly in `quadscheme.logi` (ultimately by calling `dummy.logi`) taking account of the optional arguments `dummytype`, `nd`, and `mark.repeat`.

9.10.6 Logistic regression in case-control studies

In a spatial case-control study of disease we have two types of points: cases (people diagnosed with the disease of interest) and controls (people who are similar to the cases except that they do not have the disease).

Figure 1.12 on page 9 shows the Chorley-Ribble cancer data of Diggle [224] giving the residential locations of new cases of cancer of the larynx (58 cases) and cancer of the lung (978 cases) in the Chorley and South Ribble Health Authority of Lancashire, England, between 1974 and 1983. The location of a disused industrial incinerator is also given. The aim is to assess evidence for an increase in the incidence of laryngeal cancer close to the incinerator. The lung cancer cases can arguably be treated as 'controls', that is, as a sample from the susceptible population. Data analysis in [224, 236, 55] concluded there is significant evidence of an incinerator effect.

Assume that the susceptible population has some unknown spatial density $s(u)$ (people per square kilometre). As explained in Section 9.3.7, the null model of constant risk assumes that the cases are a Poisson process with intensity $\lambda(u) = \rho s(u)$ (cases per square kilometre) where ρ is the disease risk per head of population. Common alternative models assume that the cases are Poisson with intensity $\lambda(u) = r(u, \boldsymbol{\theta}) s(u)$ where $r(u, \boldsymbol{\theta})$ is a spatially varying risk function.

Suppose we treat the locations y_i of cases and controls as fixed, and consider only the disease

SAMPLE

status indicators, $I_i = 1$ if y_i is a case and $I_i = 0$ if it is a control. Then, by the principle explained in Section 9.10.2, the disease status indicators satisfy a binary regression

$$\log \frac{\mathbb{P}\{I_i = 1\}}{\mathbb{P}\{I_i = 0\}} = \log \frac{p_i}{1-p_i} = \log r(y_i, \boldsymbol{\theta}).$$

If the risk function $r(u, \boldsymbol{\theta})$ is loglinear in $\boldsymbol{\theta}$, the relationship is a logistic regression. This is a well-known principle in epidemiology. Diggle and Rowlingson [236] argue the advantages of this approach in a spatial context, which include not having to estimate the population density.

For the Chorley-Ribble dataset we can carry out such an analysis as follows:

```
> X <- split(chorley)$larynx
> D <- split(chorley)$lung
> incin <- as.ppp(chorley.extra$incin, W = Window(chorley))
> dincin <- distfun(incin)
> Q <- quadscheme.logi(X,D)
> fit <- ppm(Q~dincin)
```

For similar applications in forestry see [684, 145].

9.11 Approximate Bayesian inference

Recently Rajala [561] suggested using the conditional logistic regression likelihood as the basis for approximate Bayesian inference for point process models. In this setup (9.68) is treated as the true loglikelihood of the model and a multivariate Gaussian distribution is used as the prior for $\boldsymbol{\theta}$. The corresponding posterior density is not easy to derive, but applying a tangential lower bound to the log terms in (9.68) leads to a multivariate Gaussian distribution as a lower bound to the posterior. The mean of this lower bound distribution is used as the parameter estimate. However, the lower bound depends on so-called variational parameters and the estimate depends on how these are chosen. The best variational parameters are the ones that maximise the lower bound since then we are as close as possible to the true posterior distribution. In [561] the EM algorithm is used to find this optimal choice of the variational parameters. Rajala's code has been incorporated in `ppm` as the method `"VBlogi"`:

```
> fitVB <- ppm(bei ~ grad, data=bei.extra, method="VBlogi")
> coef(fitVB)
(Intercept)        grad
     -5.417       5.305
```

The parameter estimates are very close to those obtained when `method="logi"` in Section 9.10.5. However, the results depend on the chosen Gaussian prior distribution, which by default is chosen to be somewhat 'uninformative'. It is recommended to set the prior mean and variance-covariance matrix explicitly, so that the dependence on the prior is more apparent. Suppose for example that our prior belief is that the terrain slope (`grad`) should have very little effect on the abundance of trees, while we have no strong prior knowledge about the intercept term (underlying abundance of trees). It would be appropriate to take the prior mean of both parameters to be zero, with large prior variance for the intercept parameter, and small prior variance for the coefficient of slope:

```
> fitVBp <- ppm(bei ~ grad, data=bei.extra,
                prior.mean = c(0,0), prior.var = diag(c(10000,0.01)))
```

SAMPLE

```
> coef(fitVBp)
(Intercept)        grad
    -4.9794      0.5438
```

In this example our strong belief in the absence of a slope effect has substantially changed the estimate for the slope coefficient.

At the time of writing it is not possible to calculate the posterior variance-covariance matrix in `spatstat`, so there is currently no way to assess the uncertainty of this approximate Bayesian estimate.

SAMPLE

9.12 Non-loglinear models

This section covers methods for fitting a *general* (not loglinear) Poisson point process model. It describes the model-fitting functions `ippm` (implementing Newton's method for differentiable models) and `profilepl` (applying brute force estimation for general models, such as those involving a threshold).

9.12.1 Profile likelihood

In many intensity models, *some* of the parameters appear in loglinear form, while other parameters do not. Such a model is of the form

$$\lambda_{\boldsymbol{\theta}}(u) = \exp(\boldsymbol{\varphi}^\top \mathbf{Z}(u, \boldsymbol{\psi})) \tag{9.75}$$

where $\boldsymbol{\theta} = (\boldsymbol{\varphi}, \boldsymbol{\psi})$ is a partition of the entries of the parameter vector $\boldsymbol{\theta}$ into *regular parameters* $\boldsymbol{\varphi}$ which appear in loglinear form in (9.75) and *irregular parameters* $\boldsymbol{\psi}$ which do not appear in loglinear form.

If we *fix* the values of the irregular parameters $\boldsymbol{\psi}$, then equation (9.75) is loglinear in the remaining parameters $\boldsymbol{\varphi}$. Considered as a model with parameters $\boldsymbol{\varphi}$ only, this is a loglinear Poisson point process model, which can be fitted using the methods described in previous sections.

Thus, for any chosen value of $\boldsymbol{\psi}$, the likelihood $L(\boldsymbol{\theta}) = L(\boldsymbol{\varphi}, \boldsymbol{\psi})$ can easily be maximised over all possible values of $\boldsymbol{\varphi}$, giving us the *profile maximum likelihood estimate*

$$\widehat{\boldsymbol{\varphi}}(\boldsymbol{\psi}) = \operatorname{argmax}_{\boldsymbol{\varphi}} L(\boldsymbol{\varphi}, \boldsymbol{\psi}). \tag{9.76}$$

The achieved maximum value of the likelihood for a given value of $\boldsymbol{\psi}$ is called the *profile likelihood*:

$$\mathrm{pL}(\boldsymbol{\psi}) = \max_{\boldsymbol{\varphi}} L(\boldsymbol{\varphi}, \boldsymbol{\psi}). \tag{9.77}$$

The maximum likelihood estimate of $\boldsymbol{\theta}$ can be obtained by maximising the *profile* likelihood over $\boldsymbol{\psi}$. That is, if $\widehat{\boldsymbol{\psi}} = \operatorname{argmax}_{\boldsymbol{\psi}} \mathrm{pL}(\boldsymbol{\psi})$ is the value of the irregular parameters that maximises the profile likelihood, then $\widehat{\boldsymbol{\theta}} = (\widehat{\boldsymbol{\varphi}}(\widehat{\boldsymbol{\psi}}), \widehat{\boldsymbol{\psi}})$ is the maximum likelihood estimate of $\boldsymbol{\theta}$.

9.12.2 Maximising profile likelihood by brute force

A simple strategy for maximising the profile likelihood is to evaluate $\mathrm{pL}(\boldsymbol{\psi})$ at a grid of test values of $\boldsymbol{\psi}$ and simply choose the value of $\boldsymbol{\psi}$ which yields the maximum.

The function `profilepl` performs this 'brute force' maximisation. Its syntax is

```
profilepl(s, f, ...)
```

The argument `s` is a data frame containing values of the irregular parameters over which the profile likelihood will be computed. The argument `f` should be set to `Poisson` for fitting Poisson processes. Additional arguments `...` are passed to `ppm` to fit the model.

For example, the `nztrees` dataset appears to have a line of trees (perhaps a planted avenue) at right boundary. We fit a threshold model where the intensity is different on either side of the line $x = a$. The threshold value a can be found using `profilepl`:

```
> thresh <- function(x,y,a) { x < a }
> df <- data.frame(a=1:152)
> nzfit <- profilepl(df, Poisson, nztrees ~ thresh, eps=0.5)
```

This calculation may take considerable time. The result, `nzfit`, is an object of class `"profilepl"`. Printing the object gives information about the fitting procedure, the fitted value of the irregular parameter, and the fitted values of the regular parameters.

```
> nzfit
Profile log pseudolikelihood
for model:  ppm(nztrees ~ thresh,  eps = 0.5,  interaction = Poisson)
fitted with rbord = 0
Interaction: Poisson process
Irregular parameter: a in [1, 152]
Optimum value of irregular parameter:  a = 150
```

The `plot` method generates a plot of the profile likelihood, as shown in Figure 9.15. The plot shows a clear preference for a threshold near the right-hand edge of the field. The optimal fitted model can be extracted as `as.ppm(nzfit)`.

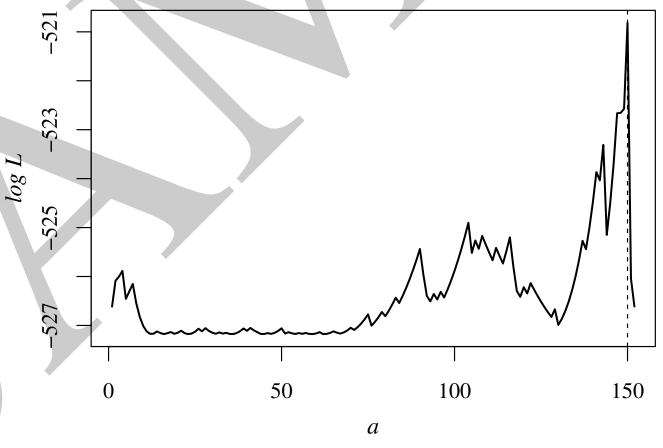


Figure 9.15. *Profile likelihood for changepoint model of NZ Trees data.*

Figure 9.15 shows the typical behaviour of profile likelihood in many applications. The profile likelihood is not continuous as a function of the threshold parameter a: it has discrete jumps at the values a_i which coincide with the horizontal coordinates of data points. Between these jumps, the profile likelihood is continuous and differentiable.

This is a simple instance of a *changepoint* model; the ‘significance’ of the threshold should be assessed using specialised methods for changepoints [147]. The R package `changepoint` may help.

SAMPLE

9.12.3 Maximising profile likelihood by Newton's method

9.12.3.1 Newton's method

If the intensity function is differentiable with respect to the irregular parameters, then a more efficient technique for maximisation is Newton's method of root-finding, applied to the score. If $\lambda_{\boldsymbol{\theta}}(u)$ is differentiable with respect to (all components of) $\boldsymbol{\theta}$, the score is

$$\mathbf{U}(\boldsymbol{\theta};\mathbf{x}) = \sum_{i=1}^{n} z_{\boldsymbol{\theta}}(x_i) - \int_W z_{\boldsymbol{\theta}}(u)\lambda_{\boldsymbol{\theta}}(u)\,\mathrm{d}u \tag{9.78}$$

where $z_{\boldsymbol{\theta}}(u) = (\partial/\partial\boldsymbol{\theta})\log\lambda_{\boldsymbol{\theta}}(u)$, and the *observed* information is

$$H(\widehat{\boldsymbol{\theta}};\mathbf{x}) = -\sum_{i=1}^{n}\kappa_{\widehat{\boldsymbol{\theta}}}(x_i) + \int_W \kappa_{\widehat{\boldsymbol{\theta}}}(u)\lambda_{\widehat{\boldsymbol{\theta}}}(u)\,\mathrm{d}u + \int_W z_{\widehat{\boldsymbol{\theta}}}(u)z_{\widehat{\boldsymbol{\theta}}}(u)^{\top}\lambda_{\widehat{\boldsymbol{\theta}}}(u)\,\mathrm{d}u \tag{9.79}$$

where $\kappa_{\boldsymbol{\theta}}(u) = (\partial/\partial\boldsymbol{\theta})z_{\boldsymbol{\theta}}(u) = (\partial^2/\partial\boldsymbol{\theta}^2)\log\lambda_{\boldsymbol{\theta}}(u)$. In the Newton-Raphson method we repeatedly update our current estimate of $\boldsymbol{\theta}$ by

$$\boldsymbol{\theta}_{m+1} = \boldsymbol{\theta}_m - H(\boldsymbol{\theta}_m;\mathbf{x})^{-1}\mathbf{U}(\boldsymbol{\theta}_m;\mathbf{x}). \tag{9.80}$$

In this case there is some asymptotic theory available for the statistical performance of the estimator [205, sec. 4.5.2].

9.12.3.2 Implementation in `spatstat`

The `spatstat` function `ippm` performs the iterative maximisation (9.80). For example, consider a Poisson point process with intensity function $\lambda(u)$ at u such that

$$\lambda(u) = \exp(\alpha + \beta Z(u))\, f(u,\gamma)$$

where α, β, γ are parameters to be estimated, $Z(u)$ is a spatial covariate function, and f is some known function. Then the parameters α, β are called *regular* because they appear in a loglinear form; the parameter γ is called *irregular*.

To fit this model using `ippm`, we specify the model formula in the usual way for `ppm`. Recall that the right-hand side of the model formula is a representation of the *log* of the intensity. In the above example the log intensity is

$$\log\lambda(u) = \alpha + \beta Z(u) + \log f(u,\gamma)$$

so the model above would be encoded with the trend formula `~Z + offset(log(f))`. Note that the irregular part of the model is an *offset* term, which means that it is included in the log trend as it is, without being multiplied by another regular parameter.

The optimisation runs faster if we specify the derivative of $\log f(u,\gamma)$ with respect to γ. We call this the *irregular score*. To specify this, the user must write an R function that computes the irregular score for any value of γ at any location `(x,y)`.

Thus, to code such a problem, (1) the model formula should define the log intensity, with the irregular part as an `offset`; (2) the argument `start` should be a list containing initial values of each of the irregular parameters; and (3) the argument `iScore`, if provided, must be a list (with one entry for each entry of `start`) of functions with arguments `x,y,...`, that evaluate the partial derivatives of $\log f(u,\gamma)$ with respect to each irregular parameter. An example is given below.

For a general concave loglikelihood, with arbitrary data, the existence and uniqueness of the maximum likelihood estimate (MLE) are not guaranteed, and depend on the absence of 'directions of recession' [584, 697, 616, 10, 282].

SAMPLE

9.12.3.3 Incinerator effect in Chorley-Ribble data

The Chorley-Ribble data (Figure 1.12 on page 9) were discussed in Sections 9.3.7 and 9.10.6.

```
>   lung <- split(chorley)$lung
>   larynx <- split(chorley)$larynx
>   Q <- quadscheme(larynx, eps=0.1)
```

Here we follow Diggle [224] in treating the cases of laryngeal cancer as the response, and taking the lung cancers as a covariate. We apply kernel smoothing [222] to the lung cancer locations to obtain an unnormalised estimate ρ of the spatially varying population density of susceptibles, shown in Figure 9.8 on page 321. This approach is open to critique [224, 236, 55] but is shown here for demonstration purposes.

```
> smo <- density(lung, sigma=0.15, eps=0.1)
> smo <- eval.im(pmax(smo, 1e-10))
```

The null model, of 'constant relative risk', postulates that the larynx cases are a Poisson process with intensity $\lambda(u) = \kappa\,\rho(u)$ at location u. The parameter κ adjusts for the relative abundance of the two types of data points; it is the baseline relative risk of larynx and lung cancer, multiplied by the ratio of sampling fractions used when these data were sampled from the cancer registry. This *could* be used to infer the absolute risk of laryngeal cancer if these sampling fractions were known. We fit the null model:

```
> ppm(Q ~ offset(log(smo)))
```

The fitted coefficient `(Intercept)=-2.824` is the estimate of $\log\kappa$.

In Section 10.5.2 we advocated using the Kolmogorov-Smirnov test or similar tests (Anderson-Darling, Berman-Lawson-Waller) to decide whether a point process depends on a covariate. However in this context, where the covariate is the distance from a fixed point and the alternative is that very small distance values are overrepresented, these tests have weak power, because the null and alternative differ only for a small range of distance values. A parametric modelling approach is more powerful in this context.

Diggle [224] considered alternative 'raised incidence' models that include an effect due to proximity to the incinerator. Assume the intensity of laryngeal cancer cases at a location u is

$$\lambda_{\boldsymbol{\theta}}(u) = \kappa\,\rho(u)\,b_{\alpha,\beta}(d(u)) \tag{9.81}$$

where $d(u)$ is the distance in kilometres to the incinerator from location u. Here $\boldsymbol{\theta} = (\kappa, \alpha, \beta)$ is the parameter vector and

$$b_{\alpha,\beta}(d) = 1 + \alpha\exp(-\beta d^2) \tag{9.82}$$

is the raised risk ratio at a distance d from the incinerator [224, eq. (6)]. The parameters α, β control the magnitude and dropoff rate, respectively, of the incinerator effect. The log intensity is

$$\log\lambda_{\boldsymbol{\theta}}(u) = \log\kappa + \log\rho(u) + \log(1 + \alpha\exp(-\beta d^2)); \tag{9.83}$$

this is not a linear function of the parameters α and β, so these parameters are irregular, and we need to use another algorithm such as `ippm` or `profilepl` to fit the model. Since (9.82) is differentiable with respect to α and β, we can use `ippm`.

We start by defining the squared distance to the incinerator, as this function will be re-used frequently.

```
> d2incin <- function(x, y, xincin=354.5, yincin=413.6) {
     (x - xincin)^2 + (y - yincin)^2
  }
```

SAMPLE

The arguments `xincin,yincin` are the coordinates of the incinerator. By specifying default values for them, we do not have to handle these values again. Next we define the raised incidence term $b_{\boldsymbol{\theta}}(u)$:

```
> raisin <- function(x,y, alpha, beta) {
    1 + alpha * exp( - beta * d2incin(x,y))
  }
```

We then fit the model using `ippm` (this can be slow):

```
> chorleyDfit <- ippm(Q ~offset(log(smo) + log(raisin)),
                      start=list(alpha=5, beta=1))
```

The argument `start` lists the irregular parameters over which the likelihood is to be maximised, and gives their starting values for the iterative algorithm. The parameter names should match the names of arguments of the function `raisin`. As intended, the arguments `xincin, yincin` are held fixed, because they are not listed in `start`.

```
> chorleyDfit
Nonstationary Poisson process
Log intensity:  ~offset(log(smo) + log(raisin))
            Estimate   S.E. CI95.lo CI95.hi Ztest   Zval
(Intercept)   -2.896 0.1313  -3.153  -2.638   *** -22.05
```

The MLE's are $\hat{\alpha} = 22.2$ (dimensionless) and $\hat{\beta} = 0.89\,\mathrm{km}^{-1}$. The `spatstat` generic function `parameters` can be used to extract all the model parameters, including both regular and irregular trend parameters:

```
> unlist(parameters(chorleyDfit))
trend.(Intercept)             alpha              beta
          -2.8958           22.2410            0.8892
```

Note that the iterative maximisation procedure can fail to converge, because the parameters do not appear in loglinear form and the log-likelihood may not be a convex function. In principle there could even be several different maxima of the likelihood. Tweaking the arguments passed to the optimisation function `nlm` usually sidesteps these problems.

9.13 Local likelihood

In local likelihood inference [335, 428] or geographically weighted regression [267], the model parameters are allowed to be spatially varying. More precisely, we start with a homogeneous model; at each spatial location u we fit the same model to data in a neighbourhood of u, obtaining a parameter estimate $\widehat{\boldsymbol{\theta}}(u)$ that will vary between locations.

Suppose we define the neighbourhood of location u as the disc $b(u,R)$ centred at u, with some fixed radius R. At each location u we would fit a model to the data inside $b(u,R)$, obtaining a parameter estimate $\widehat{\boldsymbol{\theta}}(u)$. As we move the disc over the study region, the fitted parameter values will change. This is reminiscent of the scan statistic (page 190).

Instead of including or excluding data according to whether they fall inside or outside a disc, we could weight the data using a smoothing kernel κ. For a Poisson point process model with intensity $\lambda_{\boldsymbol{\theta}}(u)$, the *local log likelihood* or *geographically weighted likelihood* at location u is

$$\log L_u(\boldsymbol{\theta}) = \sum_{i=1}^{n} \kappa(u - x_i) \log \lambda_{\boldsymbol{\theta}}(x_i) - \int_W \kappa(u - v)\, \lambda_{\boldsymbol{\theta}}(v)\, \mathrm{d}v. \tag{9.84}$$

SAMPLE

For each location u, the local log likelihood can be maximised using the same techniques as for the original likelihood, yielding a parameter estimate $\widehat{\theta}(u)$.

Code for local likelihood [33] will be added to `spatstat` soon. The fitting function is `locppm`. For the Queensland copper data:

```
> X <- rotate(copper$Points, pi/2)
> L <- rotate(copper$Lines, pi/2)
> D <- distfun(L)
> copfit <- locppm(X ~ D, eps=1.5, sigma=bw.locppm)
```

Here the smoothing bandwith σ was selected by cross-validation.

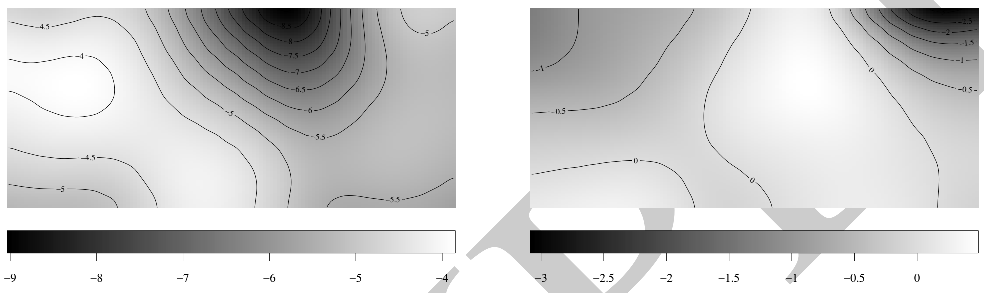


Figure 9.16. *Spatially varying estimates of intercept (*Left*) and slope coefficient (*Right*) from the local likelihood fit of the loglinear model to the Queensland copper data.*

Figure 9.16 is the result of `plot(copfit)`, showing the spatially varying fitted estimates $\widehat{\alpha}(u)$, $\widehat{\beta}(u)$ of the intercept parameter α and slope parameter β in the loglinear model $\lambda(u) = \exp(\alpha + \beta d(u))$ where $d(u)$ is the distance function of the geological lineaments. The estimates of β are close to zero, suggesting a weak relationship, across most of the survey area.

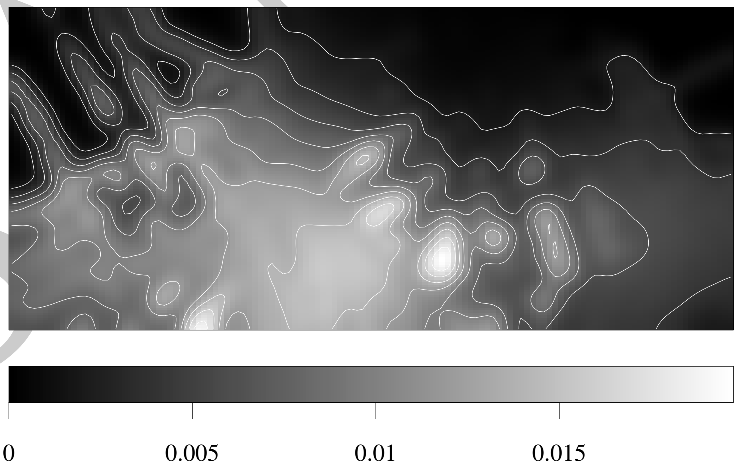


Figure 9.17. *Fitted intensity of copper deposits according to the local likelihood fit of the loglinear model.*

Figure 9.17 shows the fitted intensity $\widehat{\lambda}(u) = \exp(\widehat{\alpha}(u) + \widehat{\beta}(u)d(u))$ of each local model. This strongly suggests there is a higher intensity of copper deposits in the middle of the survey region.

SAMPLE

Figure 9.17 is very similar to the picture that would be obtained simply by kernel smoothing the copper deposits without reference to the lineament pattern. However, the top left of Figure 9.17 shows an area where the predicted intensity under the local model is higher than the kernel-smoothed intensity, because of an apparent association between copper and lineaments in that area.

9.14 FAQ

- *How accurate are the parameter estimates computed by* `ppm` *?*

  This depends on the algorithm settings such as the number of quadrature points, the quadrature weighting method, and the choice of fitting algorithm. See [53, 38] for detailed measurements of accuracy and bias.

  The default settings are chosen to achieve speed rather than high accuracy, and are suitable for initial analysis. (High speed is a requirement of the R package checker.) For accurate or definitive analysis, a larger number of quadrature points should be used, by setting the argument `nd` or `eps` in `ppm`, or by setting `spatstat.options("ndummy.min")`.

- *I get different parameter estimates from different versions of* `spatstat`.

  Later versions of `spatstat` typically give more accurate results. Progressive increases in computational efficiency allow us to change the default settings of `ppm` to increase accuracy.

  To ensure that calculations are exactly reproducible, the algorithm parameters would have to be fixed explicitly. In `ppm` this means the arguments `method` and `correction`, and a specification of the quadrature scheme, such as the arguments `nd` and `nt`.

- *Does* `ppm` *take account of correlation and confounding between the covariates?*

  Like most regression techniques, `ppm` treats the covariates as non-random quantities that are measured without error. The covariates do not have 'correlation' in the statistical sense. Any empirical correlation between the covariates is taken into account when calculating the variance-covariance matrix (`vcov.ppm`), when performing the likelihood ratio test (`anova.ppm`), and so on. If the covariates are confounded in the sense that they are approximately linearly related, the Fisher information matrix will be singular or ill-conditioned, and an error or warning will occur.

- *When I perform pixel logistic regression using* `slrm`*, the fitted presence probabilities in each pixel are tiny numbers like* 10^{-8}*. Is this physically meaningful?*

  Yes. The pixels are tiny, so the probability that a point falls in a given pixel is tiny. In any spatial region of appreciable size, the expected number of points is the sum of the pixel presence probabilities, a sum of a very large number of very small values, yielding a physically meaningful probability. The chance that I win the lottery is tiny; the chance that someone wins the lottery is high.

- *Is it true that a logistic regression model does not assume independence between pixels?*

  The simplest answer is that standard application of logistic regression gives the same result as if we had assumed independence between pixels.

  The logistic regression relationship $\log(p/(1-p)) = \beta_0 + \beta_1 x$ itself does not assume independence between pixels. However, in order to fit a regression model, we must also describe the probability distribution of the observations, including the dependence between them. The standard 'logistic regression model' does assume that observations are independent. More importantly, the standard algorithm for *fitting* a logistic regression assumes independence between pixels (since

SAMPLE

the correlation structure determines the Fisher information matrix which is used in the Iteratively Reweighted Least Squares algorithm).

A logistic regression relationship can still be fitted if we assume some kind of dependence between pixels. However, the fitting algorithm must be adjusted to incorporate this dependence. See Chapter 12.

- *I want to fit a model where the intensity depends on a covariate Z. I don't want to make any restrictive assumptions about the relationship between intensity and Z. As I understand it, a loglinear Poisson point process model is a very specific parametric model, whereas logistic regression and Maxent are flexible non-parametric models. Can you confirm this?*

  Fitting a loglinear Poisson process model using `ppm` is **equivalent** to fitting a spatial (pixel) logistic regression model [34, 691], and also equivalent to fitting a Maxent model [568]. If mathematical proofs don't convince you, try comparing the results of `ppm` and `slrm`, or look at the examples in [34, 691, 568, 567].

  Fitting loglinear Poisson models (or equivalently fitting spatial logistic regressions or fitting Maxent models) is quite a flexible approach, because we are allowed to build models using any transformation of the original covariates. For example, we can easily fit a model in which the log intensity is a cubic function of the covariate Z. The limitation is that the user must specify the transformed covariates.

  If you believe that $\lambda(u)$ depends *only* on the covariate $Z(u)$, then the best place to start is a truly non-parametric estimator based on smoothing, such as `rhohat` (Section 6.6.3). Plotting the estimated relationship may suggest the form of a model which can then be fitted using `ppm`.

  Very flexible regression relationships can also be fitted using generalised additive models as explained in Section 9.6. This also works when there are several covariates.

- *How do I obtain standard errors for the estimates of irregular parameters fitted by* `ippm` *or* `profilepl`*?*

  At the time of writing, the only generally-applicable technique is a bootstrap approach in which the model is re-fitted to simulated realisations from the model, and the sample standard deviation of the parameter estimates is computed. A code snippet is available at `www.spatstat.org`.

SAMPLE

SAMPLE