Spatial point processes: Theory and practice illustrated with R

Ege Rubak

Department of Mathematical Sciences
Aalborg University

Lecture II, Feburary 17, 2011

Contents of lecture II

- Properties for the Poisson point process.
- Maximum likelihood estimation for Poisson point processes.
- Investigating dependence between points using functional summary statistics.
- Relevant commands in `spatstat` (remember to load the package when a new session is started):

  ```
  > library(spatstat)
  ```

Homogeneous Poisson processes

- The homogeneous Poisson process with intensity λ is defined by two properties:
  - $N(A)$ is Poisson distributed with mean $\lambda|A|$, for all A.
  - Conditional on $N(A) = n$ the n points are independent and uniformly distributed in A.
- The counts $N(A)$ and $N(B)$ are independent for disjoint A, B.
- We will use the homogeneous Poisson process as the basis for creating more complicated point process models.

Density for point processes

- A point process $\mathbf{X}$ in the window W has density $f(\cdot)$ with respect to the unit rate Poisson process if

$$\mathbb{E}[h(\mathbf{X})] = \mathbb{E}[h(\mathbf{Y})f(\mathbf{Y})] \tag{1}$$

  for all functionals h, where $\mathbf{Y}$ is a unit rate Poisson process (i.e. $\lambda = 1$).
- In particular the homogeneous Poisson process with intensity λ has density

$$f(\mathbf{x}) = e^{(1-\lambda)|W|}\lambda^{n(\mathbf{x})}. \tag{2}$$

- The maximum likelihood estimate $\hat{\lambda}$ of the intensity λ is

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

Inhomogeneous Poisson processes

- The inhomogeneous Poisson process with intensity function $\lambda(\cdot)$ is defined by two properties:
  - $N(A)$ is Poisson distributed with mean $\int_A \lambda(u)\,\mathrm{d}u$, for all A.
  - Conditional on $N(A) = n$ the n points are independent and identically distributed in A with density proportional to $\lambda(\cdot)$.
- The counts $N(A)$ and $N(B)$ are still independent for disjoint A, B.
- The density of an inhomogeneous Poisson process on W with intensity λ_θ parametrised by θ is

$$f(\mathbf{x}) = \exp\left(\int_W (1 - \lambda_\theta(u))\,\mathrm{d}u\right) \prod_i \lambda_\theta(x_i). \qquad (3)$$

- The MLE $\widehat{\theta}$ is not analytically tractable, so it must be computed using numerical algorithms which will be discussed further in next weeks lectures.

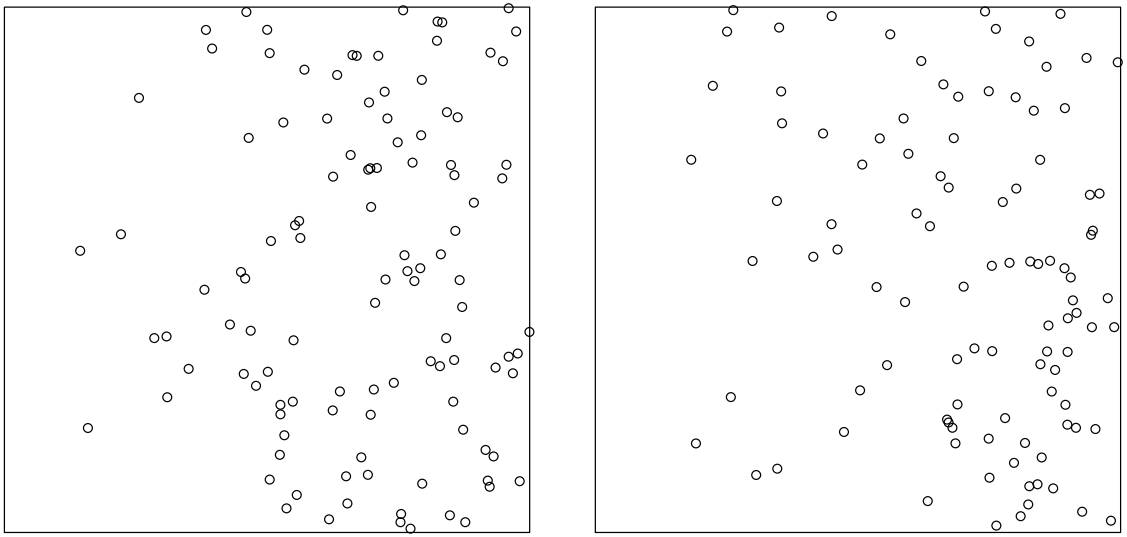
Inhomogeneous Poisson processes

- For simulation a useful technique is independent thinning: Suppose that $\mathbf{X}$ is a Poisson process with intensity function $\beta(\cdot)$, and that each point of $\mathbf{X}$ is either deleted or retained, independently of other points. If the retention probability at location u is $p(u)$, then the resulting process of retained points is Poisson, with intensity function $\lambda(u) = \beta(u)p(u)$, for all u.
- To simulate an inhomogeneous Poisson process with intensity function $\lambda(\cdot)$ we start by simulating a homogeneous Poisson process with intensity $\lambda_{\max} = \max_u\{\lambda(u)\}$. Then a thinning is performed with retention probabilies $p(u) = \lambda(u)/\lambda_{\max}$ to obtain the inhomogeneous process.

Inhomogeneous Poisson processes

To simulate an inhomogeneous Poisson process we can supply the intensity as a function or a pixel image:

```
> intenfun <- function(x, y) 200 * x
> plot(rpoispp(intenfun, lmax = 200), main = "")
> intenim <- as.im(intenfun, dimyx = c(100, 100),
+     W = unit.square())
> plot(rpoispp(intenim), main = "")
```



Model-fitting in spatstat

- The spatstat implementation requires the intensity function $\lambda_\theta(u)$ to be loglinear in the parameter θ:

$$\log \lambda_\theta(u) = \theta \cdot S(u) \tag{4}$$

  where $S(u)$ is a real-valued or vector-valued function of location u. The form of S is arbitrary so this is not much of a restriction.
- In practice $S(u)$ could be a function of the spatial coordinates of u, or an observed covariate, or a mixture of both.

Model-fitting in spatstat

- The fitting function is called `ppm` ('point process model') and is very closely analogous to the model fitting functions in R such as `lm` and `glm`.
- The statistic $S(u)$ is specified by an R language formula, like the formulas used to specify the systematic relationship in a linear model or generalised linear model.
- The basic syntax is:

  `> ppm(X, ~trend)`

  where `X` is the point pattern dataset, and `~trend` is an R formula with no left-hand side. This should be viewed as a model with log link, so *the formula* `~trend` *specifies the form of the* **logarithm** *of the intensity function*.

Model-fitting in spatstat

Fitting a Poisson model with an intensity that is log-linear in the cartesian coordinates, i.e. $\lambda_\theta((x,y)) = \exp(\theta_0 + \theta_1 x + \theta_2 y)$:

```
> data(bei)
> ppm(bei, ~x + y)
```

```
Nonstationary Poisson process

Trend formula: ~x + y

Fitted coefficients for trend formula:
  (Intercept)             x             y
-4.7245290274 -0.0008031288  0.0006496090
```

In this case the fitted intensity function is

$$\lambda_\theta((x,y)) = \exp\left(-4.724529 + -0.000803\, x + 0.00065\, y\right).$$

Model-fitting in spatstat

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(bei, ~polynom(x, y, 2))

Nonstationary Poisson process

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

Fitted coefficients for trend formula:
         (Intercept)   polynom(x, y, 2)[x]
       -4.275762e+00         -1.609187e-03
polynom(x, y, 2)[y] polynom(x, y, 2)[x^2]
       -4.895166e-03          1.625968e-06
polynom(x, y, 2)[x.y] polynom(x, y, 2)[y^2]
       -2.836387e-06          1.331331e-05
```

Model-fitting in spatstat

To fit a model with constant but unequal intensities on each side of the vertical line $x = 500$, the explanatory variable $S(u)$ should be a factor with two levels, `Left` and `Right` say, taking the value `Left` when the location u is to the left of the line $x = 500$.

```
> side <- function(z) factor(ifelse(z < 500, "left",
+     "right"))
> ppm(bei, ~side(x))
```

```
Nonstationary Poisson process

Trend formula: ~side(x)

Fitted coefficients for trend formula:
 (Intercept) side(x)right
  -4.8026460   -0.2792705
```

Model-fitting in spatstat

For the rainforest data we may want to fit the inhomogeneous Poisson model with intensity which is a loglinear function of slope, i.e.

$$\lambda(u) = \exp(\beta_0 + \beta_1 Z(u)) \tag{5}$$

where β_0, β_1 are parameters and $Z(u)$ is the slope at location u:

```
> slope <- bei.extra$grad
> ppm(bei, ~slope, covariates = list(slope = slope))

Nonstationary Poisson process

Trend formula: ~slope

Fitted coefficients for trend formula:
(Intercept)       slope
  -5.390553    5.022021
```

Model-fitting in spatstat

It might be more appropriate to fit the model

$$\lambda(u) = \beta Z(u) \tag{6}$$

where again $Z(u)$ is the slope at u. Equivalently

$$\log \lambda(u) = \log \beta + \log Z(u). \tag{7}$$

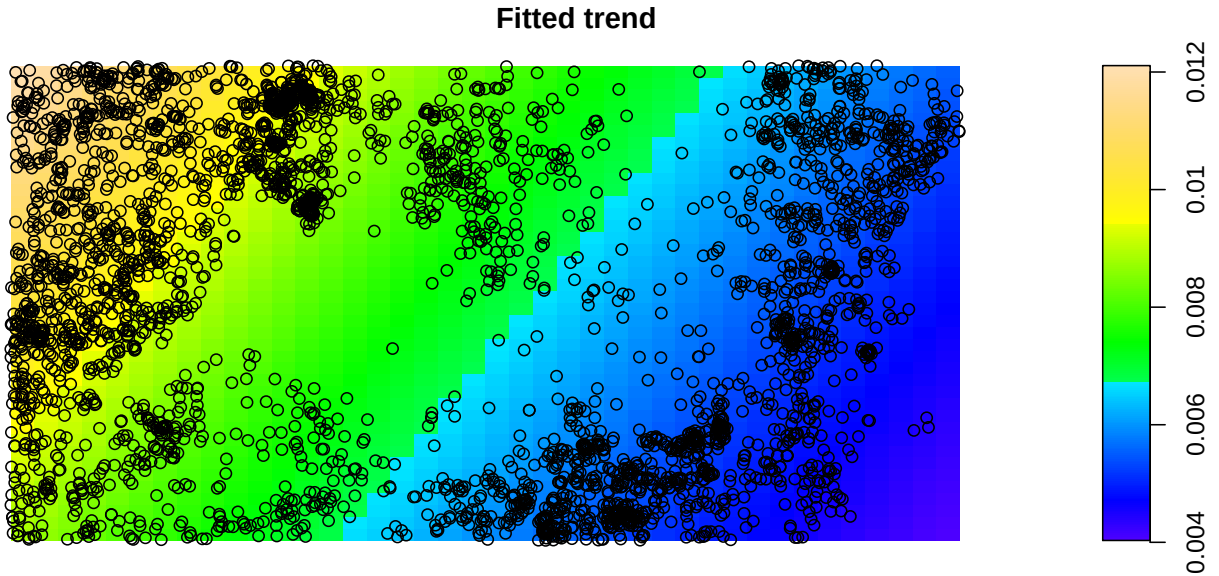
There is no coefficient in front of the term $\log Z(u)$ in (7), so this term is an 'offset'. To fit this model,

```
> ppm(bei, ~offset(log(slope)), covariates = list(slope = s
Nonstationary Poisson process

Trend formula: ~offset(log(slope))

Fitted coefficients for trend formula:
(Intercept)
  -2.427127
```

Predicting the intensity function

The values of the intensity funcion can be calculated on a grid in the window for a fitted model via the function `predict`. Alternatively, we can simply plot the fitted object which calls the `predict`-function and plots the intensity function:

```
> fit <- ppm(bei, ~x + y)
> plot(fit, how = "image", se = FALSE, pause = FALSE)
```



Variance of estimates

For Poisson models it is possible to calculate the variance-covariance matrix of $\widehat{\theta}$:

```
> vcov(fit)
```

```
             (Intercept)             x             y
(Intercept)  1.854091e-03 -1.491267e-06 -3.528289e-06
x           -1.491267e-06  3.437842e-09  1.208410e-14
y           -3.528289e-06  1.208410e-14  1.338955e-08
```

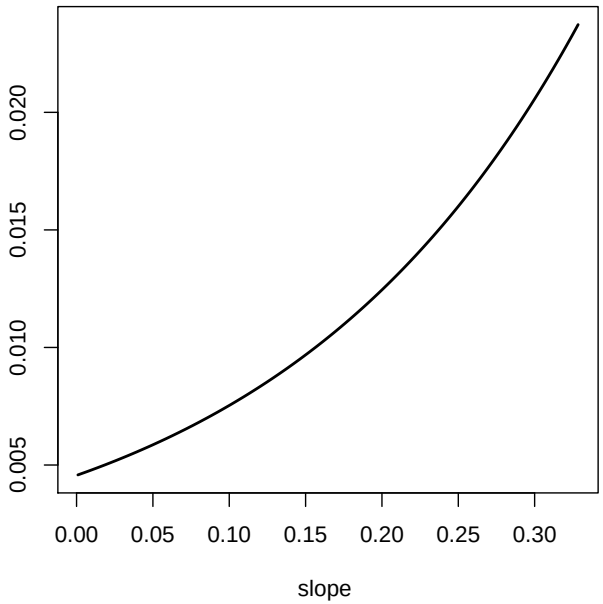
This can be used to give standard deviations on the estimates:

i	θ_i	$\mathrm{Var}(\widehat{\theta}_i)$	standard deviation
0	-4.724529	0.001854091	0.04305915
1	-0.0008031288	3.437842e-09	5.863311e-05
2	0.000649609	1.338955e-08	0.0001157132

Covariate effect plot

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. This is chiefly useful if the model only has one covariate.

```
> fit <- ppm(bei, ~slope, covariates = list(slope = slope))
> plot(effectfun(fit, "slope"), main = "")
```



Analysis of deviance

Analysis of deviance for **nested** Poisson point process models is done by using the function anova:

```
> fitnull <- update(fit, ~1)
> anova(fitnull, fit, test = "Chi")

Analysis of Deviance Table

Model 1: .mpl.Y ~ 1
Model 2: .mpl.Y ~ slope
  Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1     20507      18728
2     20506      18346  1   382.25 < 2.2e-16 ***
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.0
```

Clearly, the model including slope is favored.

Akaike Information Criterion

The homogeneous Poisson process is not nested in the model (6), in which intensity is proportional to slope. One possibility here is to use the Akaike Information Criterion **AIC** for model selection:

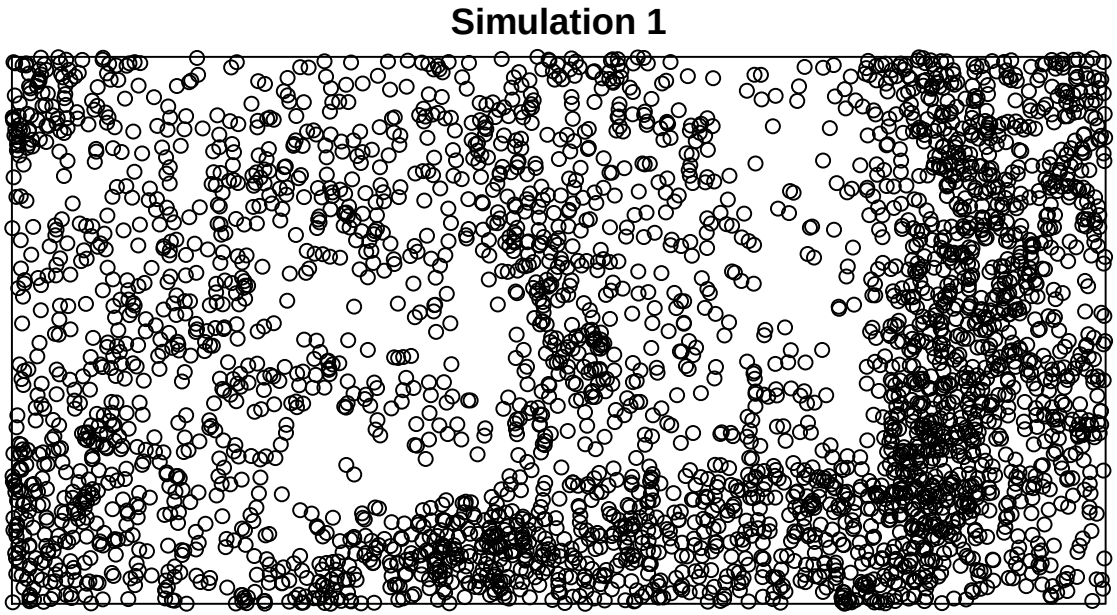
```
> fit <- ppm(bei, ~offset(log(slope)), covariates = list(sl
> fitnull <- ppm(bei, ~1)
> AIC(fit)

[1] 42496.65

> AIC(fitnull)

[1] 42763.92
```

The smaller AIC favours the model (6) with intensity proportional to slope.

Simulation of fitted model

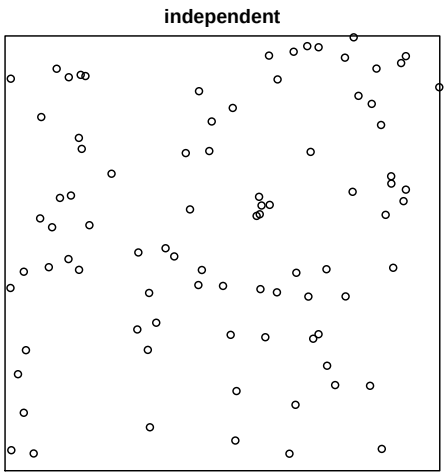
A fitted Poisson model can be simulated automatically using the `simulate`-function:

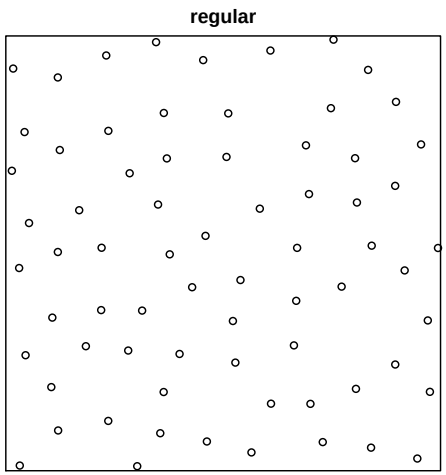
```
> X <- simulate(fit)
> plot(X, main = "")
```

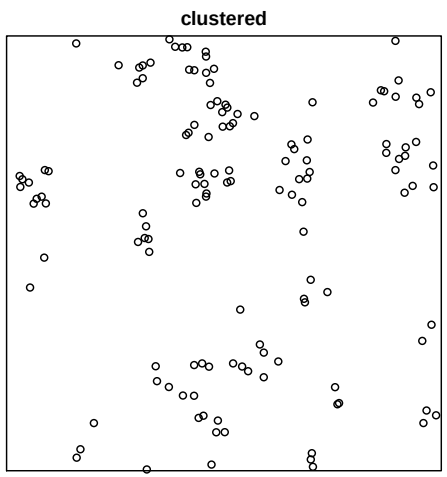


Independence, regularity and clustering

Suppose that a point pattern appears to have constant intensity, and we wish to assess whether the pattern is Poisson. The alternative is that the point pattern is either regular or clustered.

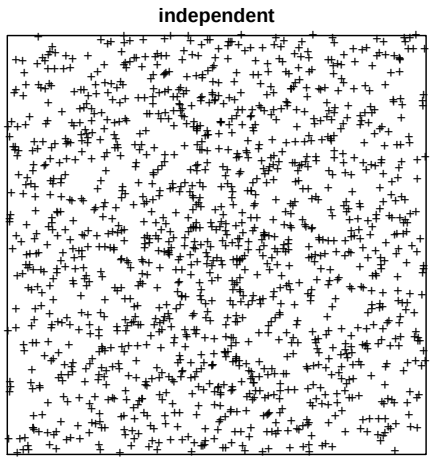


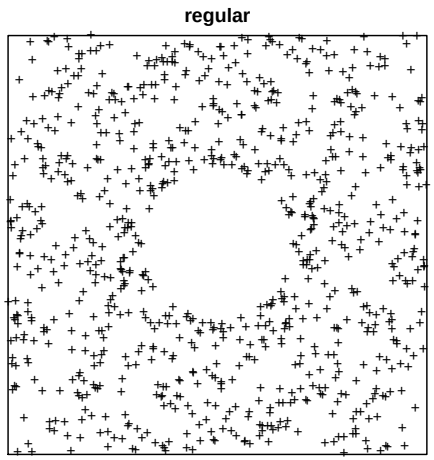


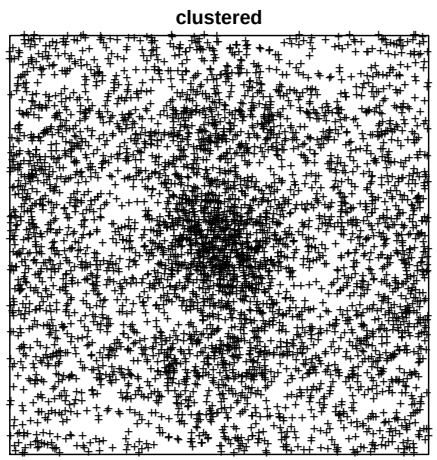


Fry plot

The *Fry plot* is a scatterplot of the vector differences $x_j - x_i$ between all pairs of distinct points in the pattern. It is computed by the command `fryplot`.







Distances

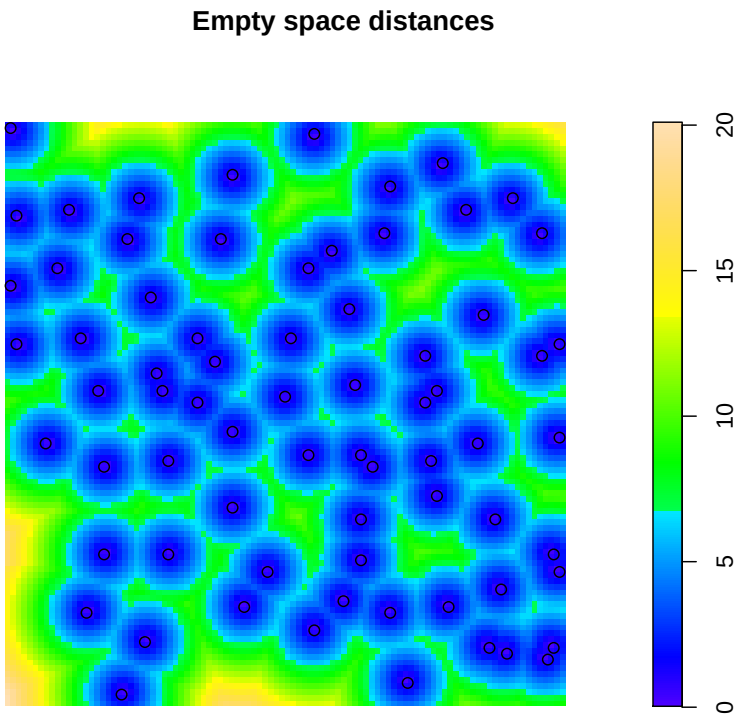
Classical techniques for investigating interpoint interaction are *distance methods*, based on measuring the distances between points. Specifically we may consider

- **pairwise distances** $s_{ij} = ||x_i - x_j||$ between all distinct pairs of points x_i and x_j $(i \neq j)$ in the pattern;
- **nearest neighbour distances** $t_i = \min_{j \neq i} s_{ij}$, the distance from each point x_i to its nearest neighbour;
- **empty space distances** $d(u) = \min_i ||u - x_i||$, the distance from a fixed reference location u in the window to the nearest data point.

These are avalible in `spatstat` as the functions `pairdist`, `nndist` and `distmap` respectively.

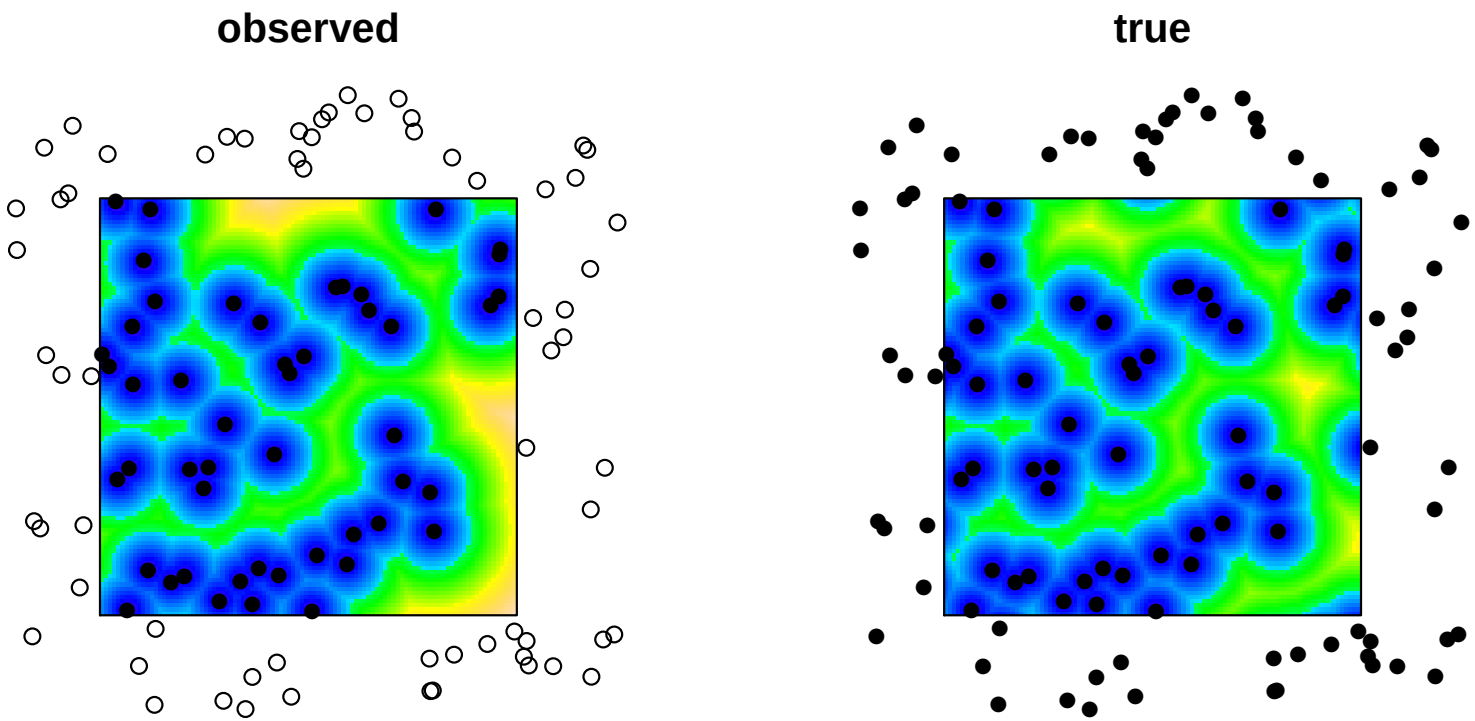
Empty space distances

```
> data(swedishpines)
> emp <- distmap(swedishpines)
> plot(emp, main = "Empty space distances")
> plot(swedishpines, add = TRUE)
```



Edge effects

Confining observations to a window W implies that the observed distance $d(u, \mathbf{x}) = d(u, \mathbf{X} \cap W)$ to the nearest data point inside W, may be greater than the true distance $d(u, \mathbf{X})$ to the nearest point of the complete point process $\mathbf{X}$.



Empty space function F

- Assuming $\mathbf{X}$ is *stationary* (statistically invariant under translations), we can define the cumulative distribution function of the empty space distance

$$F(r) = \mathbb{P}\{d(u, \mathbf{X}) \leq r\} \tag{8}$$

where u is an arbitrary reference location. If the process is stationary then this definition does not depend on u.

- The empirical cumulative distribution function (ecdf) of the observed empty space distances on a grid of locations u_j, $j = 1, \ldots, m$,

$$F^*(r) = \frac{1}{m} \sum_j \mathbf{1}\{d(u_j, \mathbf{x}) \leq r\} \tag{9}$$

is a negatively biased estimator of $F(r)$, as explained above.

Edge-correction

- Typically edge-corrected estimates of F are weighted versions of the ecdf,

$$\widehat{F}(r) = \sum_j e(u_j, r)\mathbf{1}\left\{d(u_j, \mathbf{x}) \leq r\right\} \tag{10}$$

  where $e(u, r)$ is an edge correction weight designed so that $\widehat{F}(r)$ is unbiased.
- The edge effect problem can also be regarded as a form of censoring (analogous to right-censoring in survival data), and a counterpart of the Kaplan-Meier estimator is available.
- Thus, *assuming that the point process is homogeneous*, we are able to compute an unbiased and reasonably accurate estimate of the empty space function F defined by (8).

Empty space function: Poisson case

- Notice that $d(u, \mathbf{X}) > r$ if and only if there are no points of X in the disc $b(u, r)$ of radius r centred on u. For a homogeneous Poisson process with intensity λ the probability that there are no points in this region is $\exp(-\lambda\pi r^2)$. Hence for a Poisson process

$$F_p(r) = 1 - \exp(-\lambda\pi r^2). \tag{11}$$

- Typically we compare $\widehat{F}(r)$ with the value of $F_p(r)$ obtained by plugging in the estimated intensity $\hat{\lambda} = n(\mathbf{x})/\text{area}(W)$. Values $\widehat{F}(r) > F_p(r)$ suggest that empty space distances in the point pattern are shorter than for a Poisson process, suggesting a regularly space pattern; while values $\widehat{F}(r) < F_p(r)$ suggest a clustered pattern.

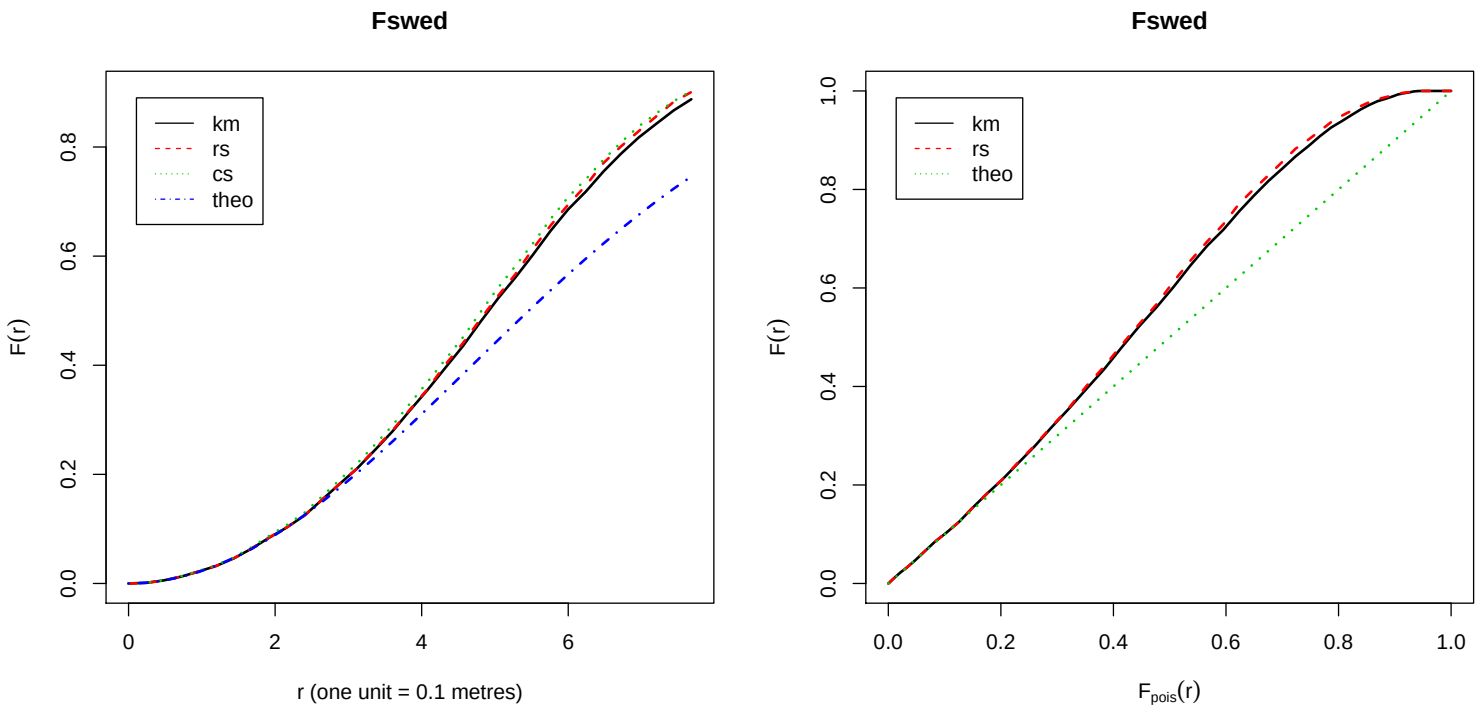
The empty space function in spatstat

The function Fest computes estimates of $F(r)$ using several edge corrections, and the benchmark value for the Poisson process.

```
> Fswed <- Fest(swedishpines)
> Fswed
Function value object (class âĂŸfvâĂŹ)
for the function r -> F(r)
Entries:
id       label         description
--       -----         -----------
r        r             distance argument r
theo     F[pois](r)    theoretical Poisson F(r)
cs       F[cs](r)      Chiu-Stoyan estimate of F(r)
rs       F[bord](r)    border corrected estimate of F(r)
km       F[km](r)      Kaplan-Meier estimate of F(r)
hazard   hazard(r)     Kaplan-Meier estimate of hazard funct
----------------------------------------
```

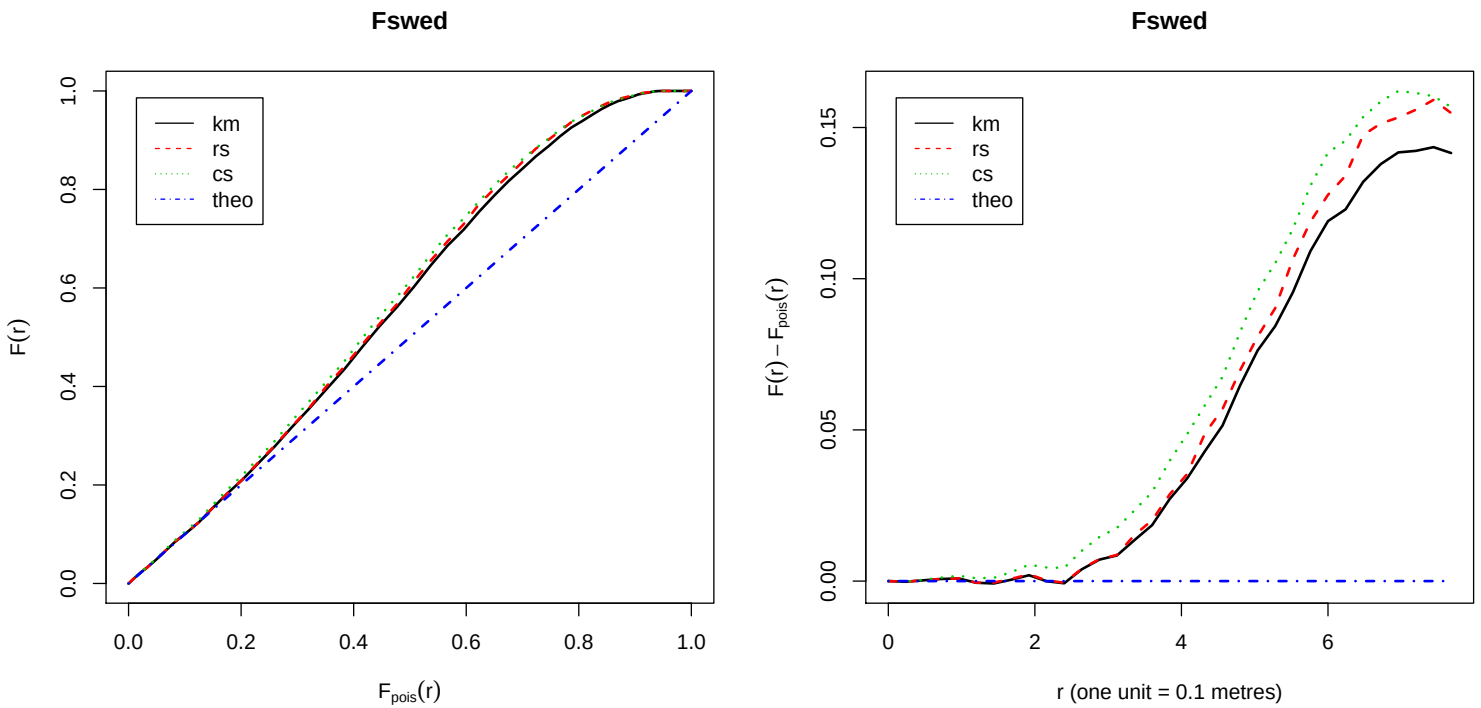
Plotting the empty space function

```
> plot(Fswed)
> plot(Fswed, cbind(km, rs, theo) ~ theo)
```



Plotting the empty space function

```
> plot(Fswed, . ~ theo)
> plot(Fswed, . - theo ~ r)
```



Thank your for your attention.