Point Processes 4: Model Validation

Michael Noonan

DATA 589: Spatial Statistics

Table of contents

Review

Review

Last lecture we saw that, for a homogeneous point process, the number of points falling in any region can be treated as Poisson distributed random variable.

This property allowed us to go beyond descriptives and define a formal framework for modelling point processes, which in turn allows us to make general inference about our study system, and also make predictions.

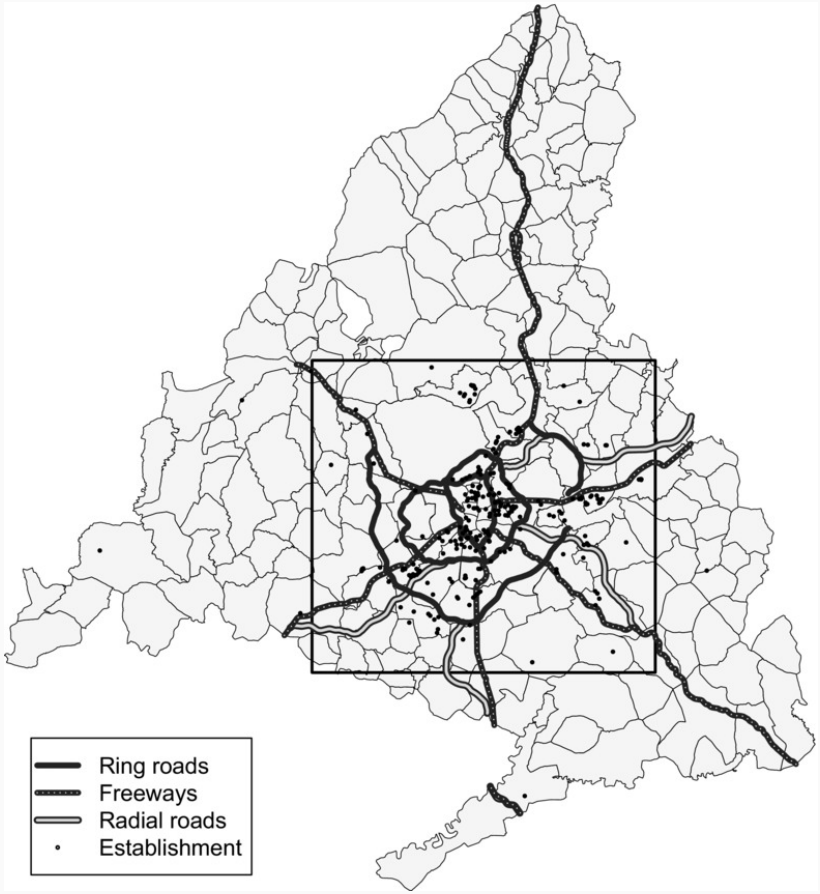
We also saw that we can visualise fitted models and perform model selection to identify the best fit model for the data at hand... but we didn't cover methods for validating our models.

Today we will focus on tools for validating PPP models, and we will finish with some general guidance on where to go from here.

Applied Points Pattern Analysis

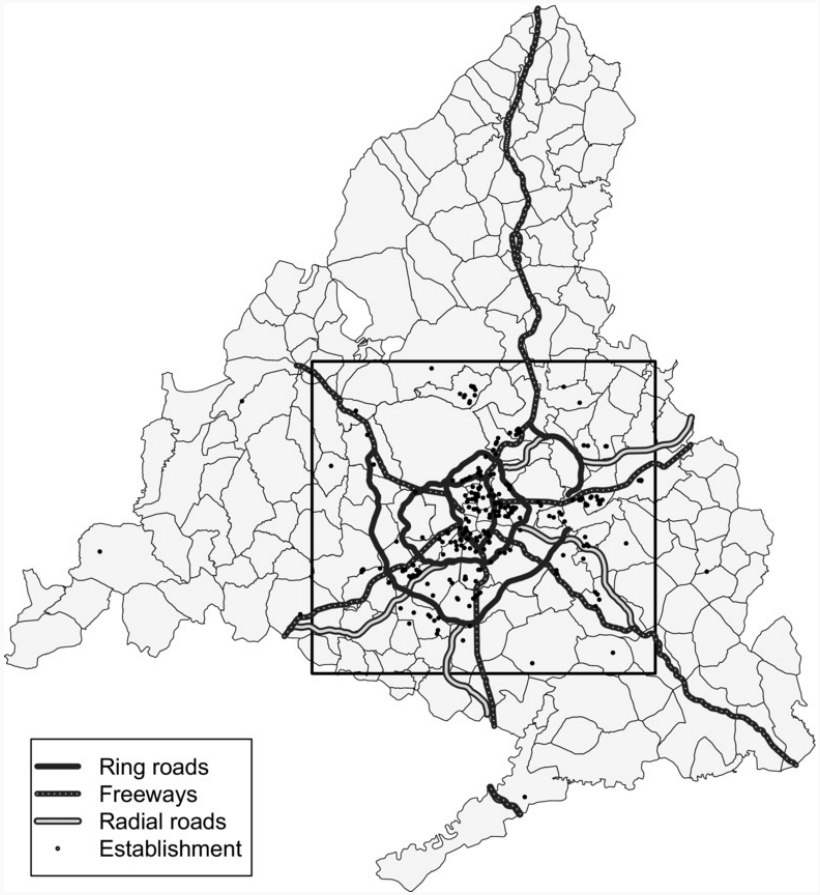
Electronics industry cohesion

Sweeney & Gómez-Antonio (2016) used a Gibbs point process to model the first (intensity) and second (correlation) moments of the electronics manufacturing industry in Madrid.



Electronics industry cohesion

Road networks and public transport were important drivers of manufacturer locations, but electronics manufacturers showed strong cohesion after accounting for the first moment effects.

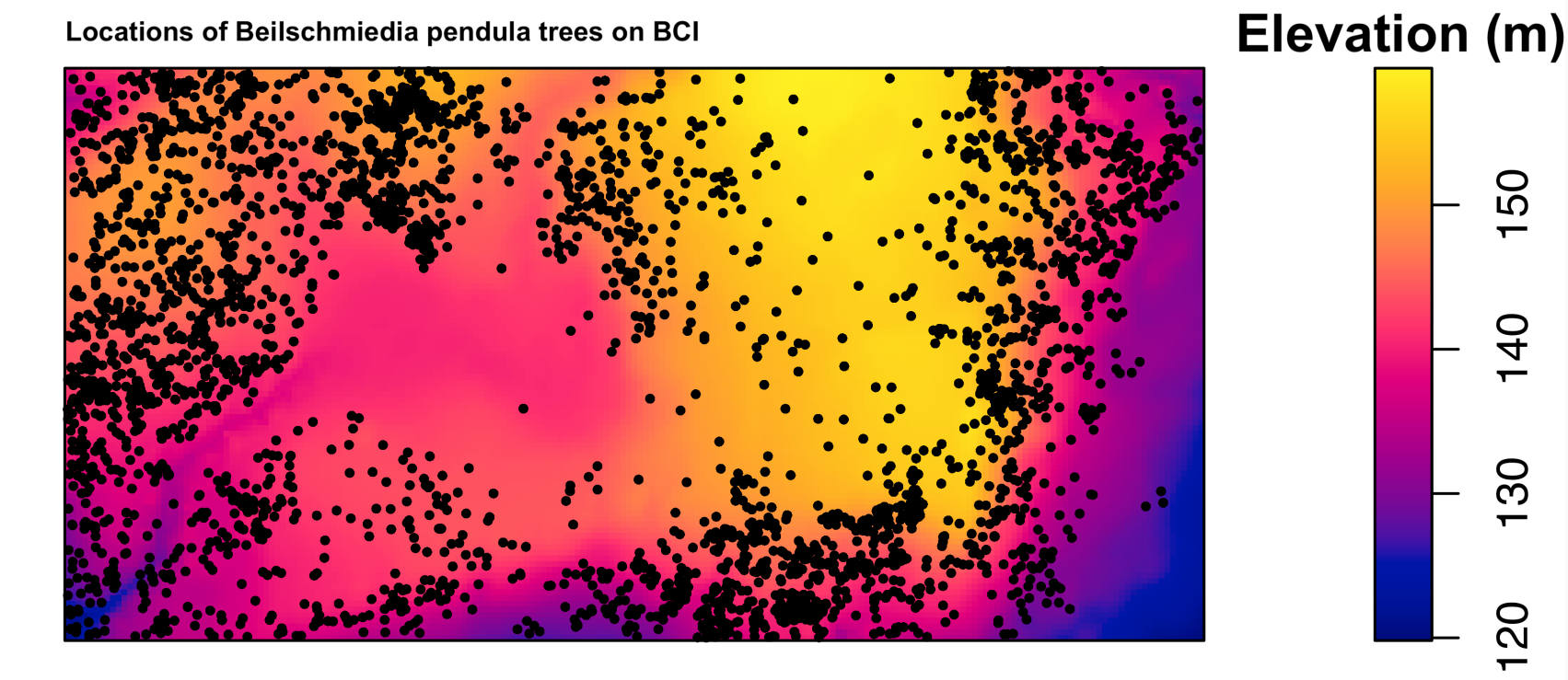


Motivation

Beilschmiedia pendula dataset

Last lecture we were fitting a Poisson point processes to data on the locations of *Beilschmiedia pendula*.

We finished with the model $\hat{\lambda}(u) = e^{-138+1.85\times\text{elev}(u)-0.0064\times\text{elev}(u)^2}$

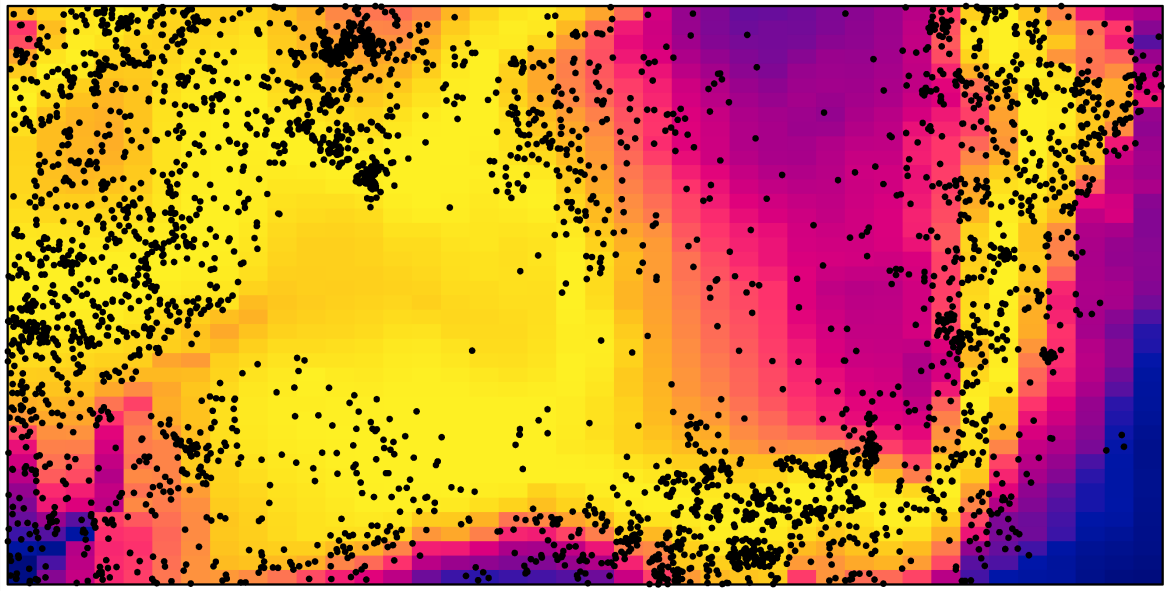


Source: `spatstat` package

Beilschmiedia pendula dataset cont.

We saw that a LRT favoured a quadratic relationship with elevation over a linear relationship.

Model selection can tell us which models from a pool of candidates have the best support given our observations, but it doesn't tell us anything about how well our model does at predicting the occurrence of *B. pendula*.

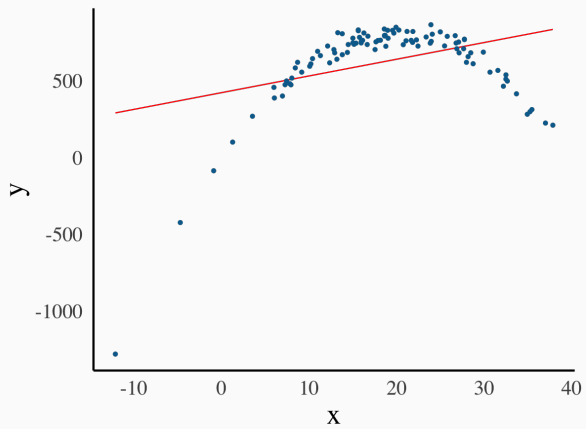


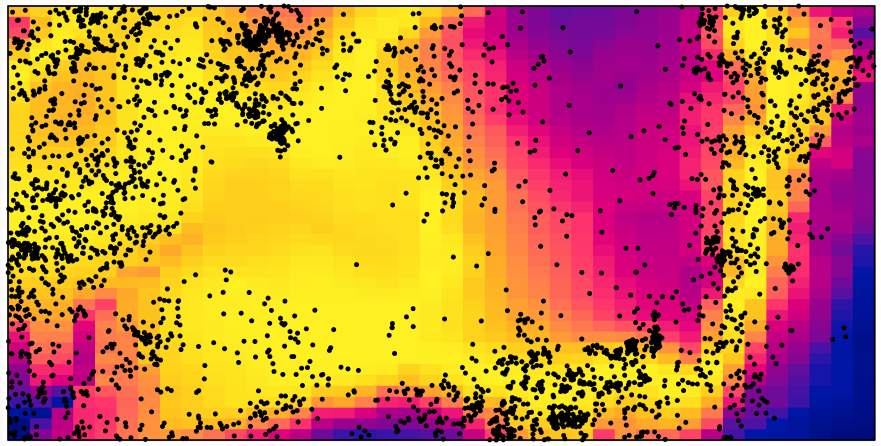
Model validation

When we fit a model to some data we are always assuming that the model has been correctly specified.

When we use software to fit a model to some data it will always estimate some coefficients even if the model is a poor fit to the data.

The same is true for point processes, but because we are working in multiple dimensions it becomes challenging to diagnose our models.





e.g., statistically these figs. are identical (fit vs. obs.), but are they equally interpretable?

Quadrat Counting Tests

Quadrat counting

Earlier in the course we saw how quadrat counting could be used as a test a set of observed points for deviations from homogeneity (or CSR), but the concept can be generalised to an inhomogeneous point process.

After fitting a model, we are saying that the intensity at any location u is $\lambda_\theta(u)$, where θ are the values of our parameters.

Under a null hypothesis that the intensity is $\lambda_\theta(u)$, then the expected number of points falling in each quadrats, B_j, is μ_j, where

$$\hat{\mu}_j = \int_{B_j} \lambda_{\hat{\theta}}(u) du$$

Quadrat counting cont.

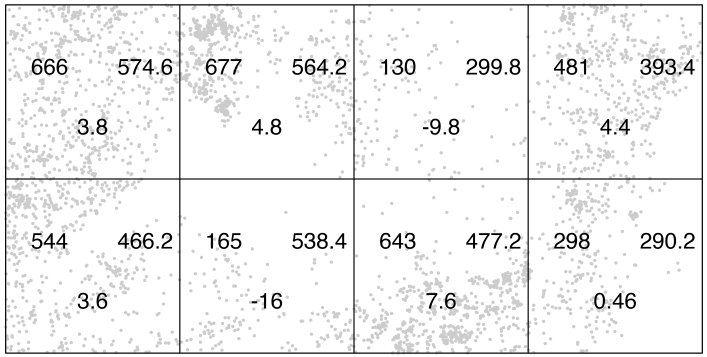
We can therefore test for significant deviations from $\lambda_\theta(u)$ using a χ^2 test

$$\chi^2 = \sum_j \frac{(\text{observed} - \text{expected})^2}{\text{expected}} = \sum_j \frac{(n_j - \hat{\mu}_j)^2}{\hat{\mu}_j}$$

Test of $\lambda_\theta(u)$ in R

As before, this test can be performed using the `quadrat.test()` function from the `spatstat` package.

```
#Fit a model with a quadratic effect for elevation
fit_quad <- ppm(bei ~ elev + I(elev^2),
                data = bei.extra)

#Run the quadrat test
quadrat.test(fit_quad, nx = 4, ny = 2)

  Chi-squared test of fitted Poisson model   fit _quad
         using quadrat counts

data:  data from fit_quad
X2 = 482.6, df = 5, p-value < 2.2e-16
alternative hypothesis: two.sided

Quadrats: 4 by 2 grid of tiles
```



...which suggests that there's a significant deviation from our model's predictions.

χ^2 test

666 574.6 3.8	677 564.2 4.8	130 299.8 -9.8	481 393.4 4.4
544 466.2 3.6	165 538.4 -16	643 477.2 7.6	298 290.2 0.46

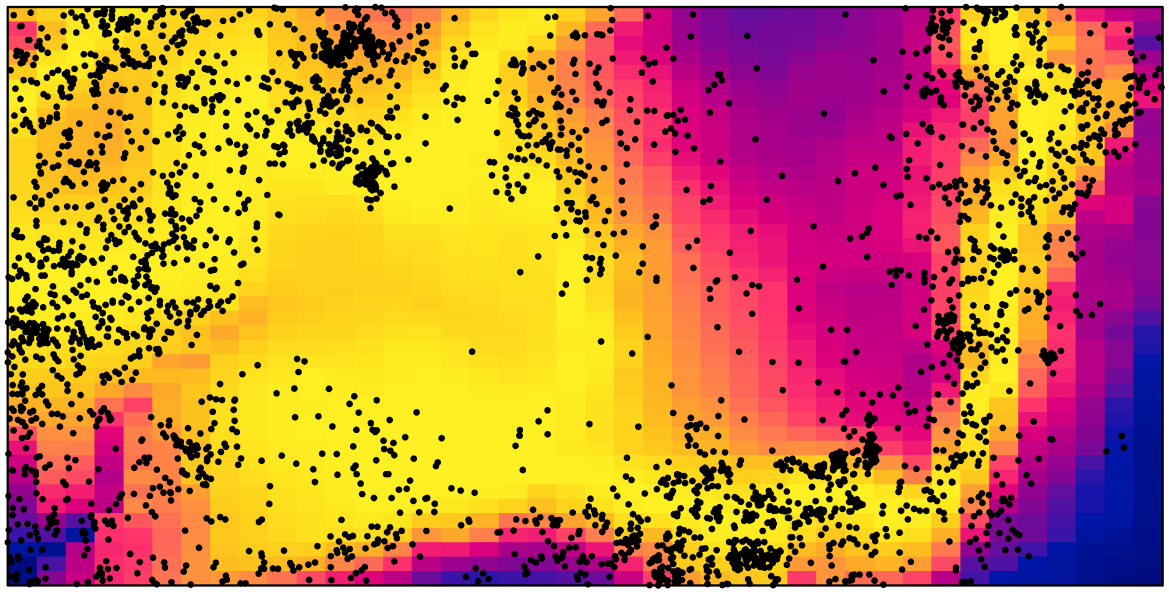
Top left = observed, top right = expected, bottom = Pearson residuals

$$\text{Pearson residual} = \frac{\text{observed} - \text{expected}}{\sqrt{\text{expected}}} = \frac{n_j - \hat{\mu}_j}{\sqrt{\hat{\mu}_j}}$$

Pearson residuals have a μ of 0 and σ of 1, so anything > 2 is unusual.

χ^2 test cont.

666 574.6 3.8	677 564.2 4.8	130 299.8 -9.8	481 393.4 4.4
544 466.2 3.6	165 538.4 -16	643 477.2 7.6	298 290.2 0.46



χ^2 test of fit caveats

This test can tell us if there are significant deviations from the predictions made by $\lambda_\theta(u)$ and the observed point data, but the p-value doesn't provide any information on the cause of the deviations.

Significant deviations from $\lambda_\theta(u)$ can be due to missing parameters, model mispecification (e.g., polynomial vs. linear), a lack of independence, non-stationarity, etc... (i.e., tells us if we have a problem, but provides no information on how to fix it).

As before, the result is sensitive to the size of the quadrats.

Poisson Point Process Residuals

Model Residuals

A model isn't always a perfect representation of what's going on in the real world, and there will be deviations between what actually happened (i.e., the observed values), and what the model predicted would happen (i.e., the predicted values).

The difference between the predicted and observed value is called the residual:

$$\text{Residual} = \text{Observed} - \text{Predicted}$$

Because residuals are supposed to have very specific behaviour (e.g., $\mathcal{N}(0, \sigma^2)$ for linear regression), they're a useful tool for evaluating models.

Poisson point processes residuals

For a point process with parameters θ, our predictions are the estimated intensity function $\lambda_\theta(u)$, and our observations are points, how can we subtract these from each other?

A point process residual is the observed number of points falling in any region B, $n(\mathrm{x} \cap B)$, minus the expected number of points, $\int_B \lambda_{\hat{\theta}}(u)du$:

$$\text{Residual} = \text{Observed} - \text{Predicted}$$

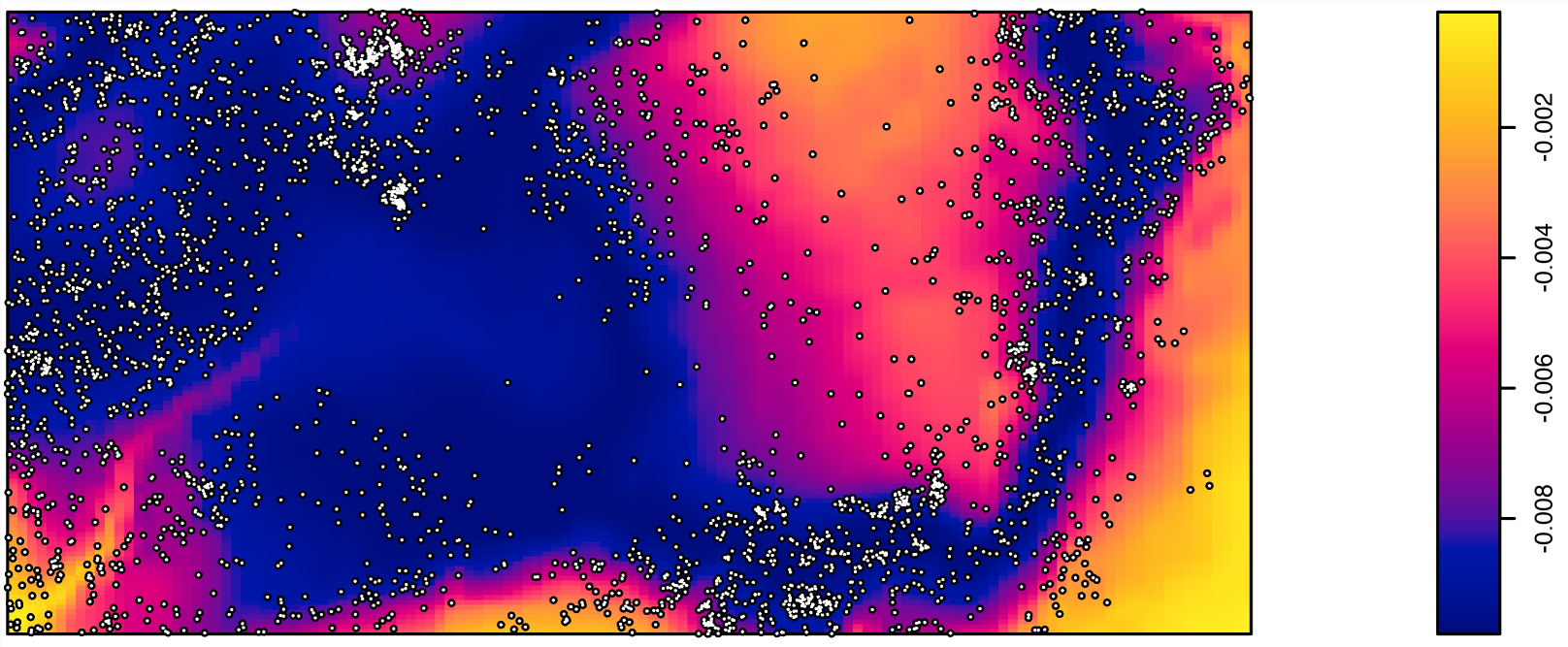
$$\mathcal{R}(B) = n(\mathrm{x} \cap B) - \int_B \lambda_{\hat{\theta}}(u)du$$

Similar to the quadrat count residuals we just saw, but defined for any 'region' B.

PPP residuals

The residuals of a PPP model are calculated using the `spatstat::residuals()` function.

```
#Calculate the residuals
res <- residuals(fit_quad)

plot(res)
```

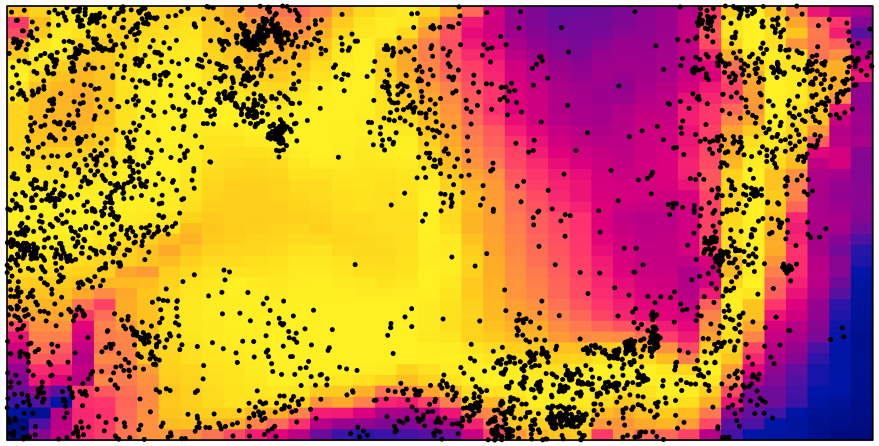


PPP residuals cont.

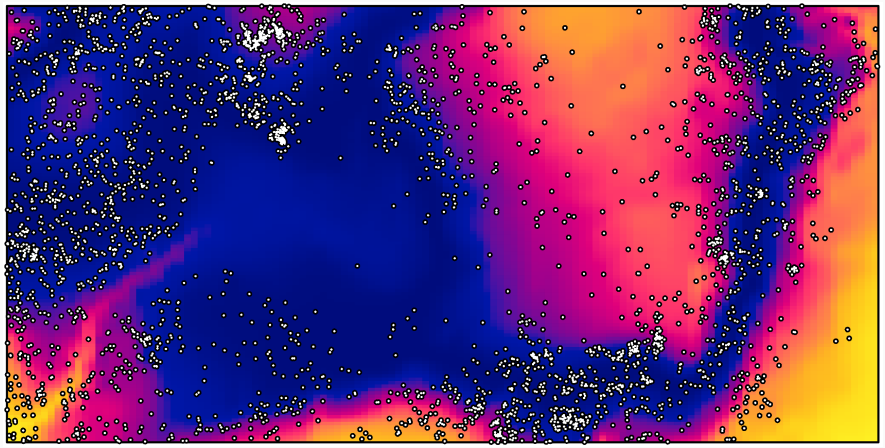
Comparing the model predictions to the residuals provides information on where the model is well/poorly-behaved.

Cause is still unknown, but there is a clear spatial pattern, which suggests an unmodelled spatial covariate.

Predictions

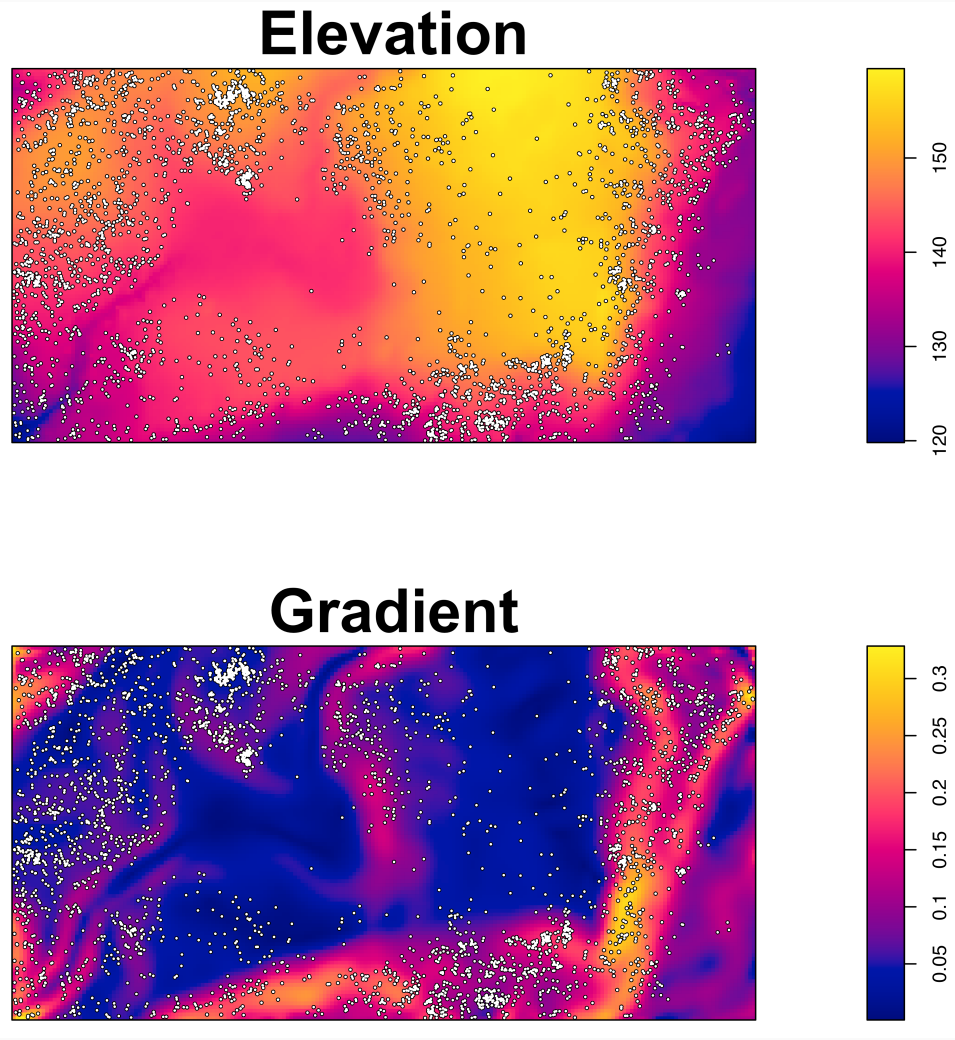


Residuals



PPP residuals cont.

This dataset also has information the gradient (slope). Thoughts?



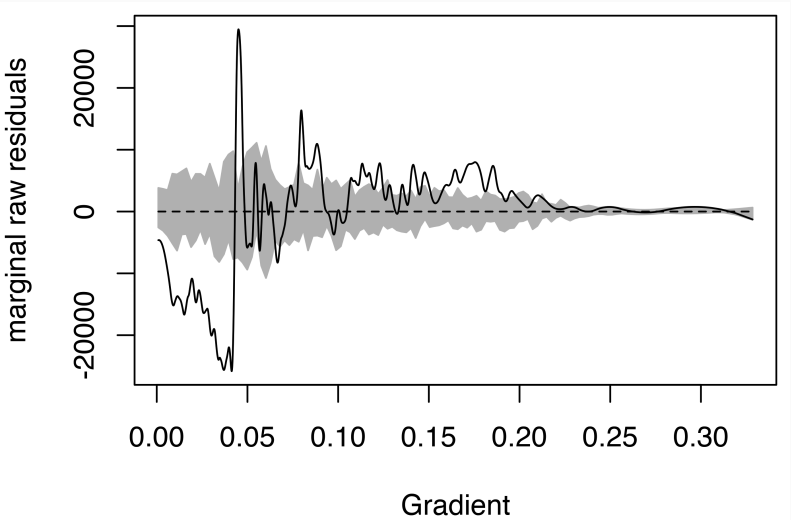
Lurking variable plot

Visual inspection suggests that gradient may play an important role in governing the spatial distribution of *B. pendula*.

To get a better feeling of whether it's worth considering an additional covariate we can use 'lurking variable' plots.

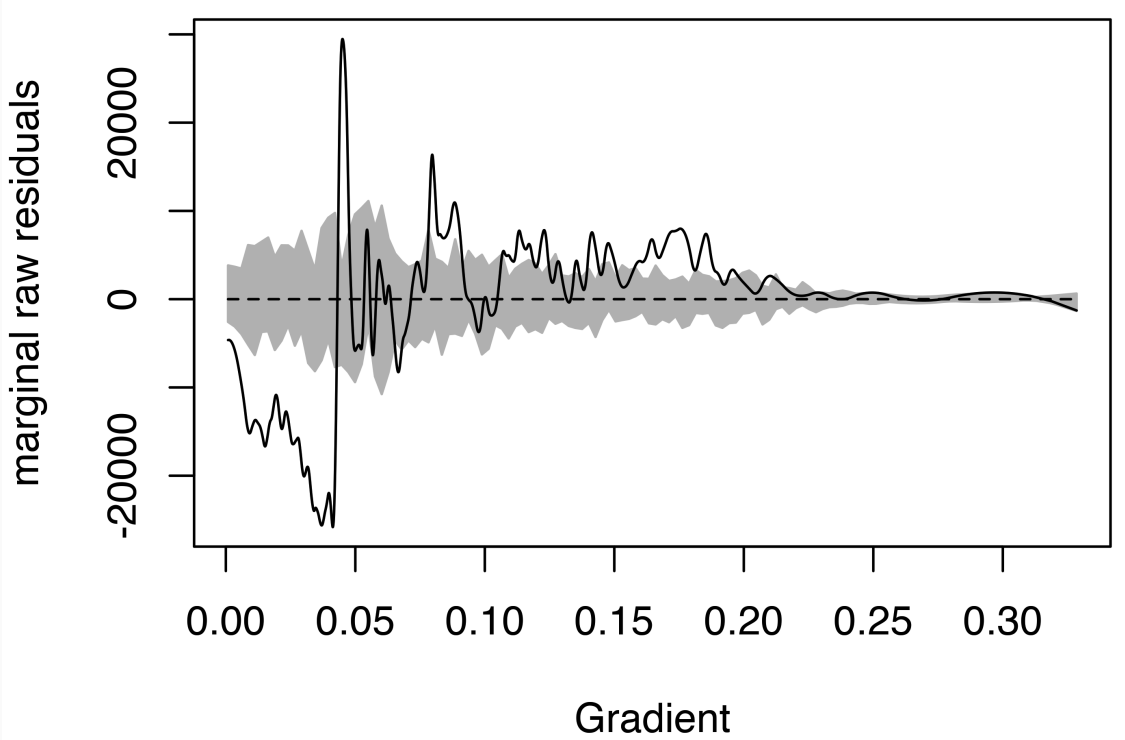
Lurking variable plot: For each possible covariate value we sum the residuals and visualise the results. Should be ~ 0 if the covariate is unrelated to trends in the residuals.

```
#Lurking variable plot

lurking(fit_quad,
        bei.extra$grad,
        type = "raw",
        cumulative = F,
        envelope = T,
        xlab = "Gradient")
```



Lurking variable plot cont.

The lurking variable plots suggests that *B. pendula* density is over-estimated at low gradients (negative residuals on average), and over-estimated at intermediate gradients (positive residuals on average).



PPP with multiple covariates

It looks like gradient is important, so we can try adding a gradient term to the model.

```
#Fit the PPP model
fit_quad <- ppm(bei ~ elev + I(elev^2) + grad, data = bei.extra)

---- Intensity: ----

Log intensity: ~elev + I(elev^2) + grad
Model depends on external covariates    elev    and    grad
Covariates provided:
  elev: im
  grad: im

Fitted trend coefficients:
  (Intercept)          elev     I(elev^2)          grad
-1.411708e+02  1.869641e+00 -6.422991e-03  5.402510e+00

                 Estimate         S.E.       CI95.lo       CI95.hi Ztest      Zval
(Intercept) -1.411708e+02 6.9555996924 -154.80356045 -1.275381e+02   *** -20.29600
elev         1.869641e+00 0.0963620230    1.68077473  2.058507e+00   ***  19.40226
I(elev^2)   -6.422991e-03 0.0003334752   -0.00707659 -5.769392e-03   *** -19.26078
grad         5.402510e+00 0.2507537766    4.91104113  5.893978e+00   ***  21.54508
```

And our new fitted model is of the form:

$$\lambda(u) = e^{-141.2+1.87\times\text{Elevation}(u)-0.0064\times\text{Elevation}(u)^2+5.4\times\text{Gradient}(u)}$$

LRT on PPP models

The term is significant, but is the additional complexity warranted?

```
#Conduct a likelihood ratio test on the gradient term
anova(fit_quad, fit, test = "LRT")

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   419.24 < 2.2e-16 ***
---
Signif. codes:  0    ***    0.001    **    0.01    *    0.05    .    0.1         1
```

The p-value is tiny, so we reject the simple model in favour of the more complex model that includes a gradient effect.

Partial residual plot

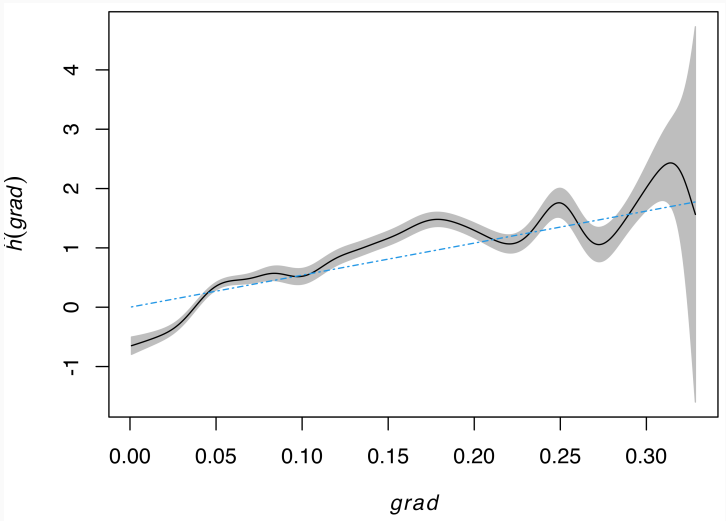
Including information on both elevation and gradient improved our model's performance, but is the gradient effect linear, or non-linear?

To answer this question we can use the partial residuals (details in sec. 11.4.3 of Baddeley *et al.* (2015) if you're interested).

Partial residual plot: shows the fitted effect of a covariate alongside the observed effect.

```
#Calculate the partial Residuals
par_res <- parres(fit, "grad")

#Visualise
plot(par_res)
```

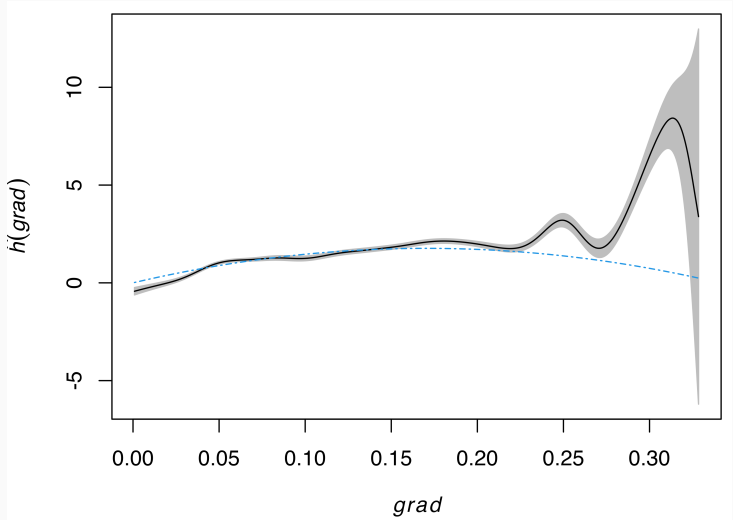
Suggests a non-linear relationship.



Refining the PPP model

Using the residuals as a diagnostic tool, we can refine our fitted model.

```
#Fit a model with a quadratic effects for
# both elevation and gradient
fit_quad_2 <- ppm(bei ~ elev + I(elev^2) +
                  grad + I(grad^2),
                  data = bei.extra)


anova(fit, fit_quad_2, test = "LRT")

Analysis of Deviance Table

Model 1: ~elev + I(elev^2) + grad    Poisson
Model 2: ~elev + I(elev^2) + grad + I(grad^2)
       Poisson
  Npar Df Deviance  Pr(>Chi)
1    4
2    5  1   238.52 < 2.2e-16 ***


#Visualise
plot(parres(fit_quad_2, "grad"))
```



The fit looks better, but is still not perfect (higher order polynomial?, missing covariate?).

In practice, we would iterate through this refinement process until we were happy with our model.

Coordinates as covariates

Normally (i.e., not spatially) when fitting regression models, there are two main ways to improve our fit:

1. Increase the complexity of exiting covariates (e.g., GAMS or higher order polynomials).

```
fit <- ppm(bei ~ elev + I(elev^2) + I(elev^3) + I(elev^4), data = bei.extra)
```

2. Add new covariates.

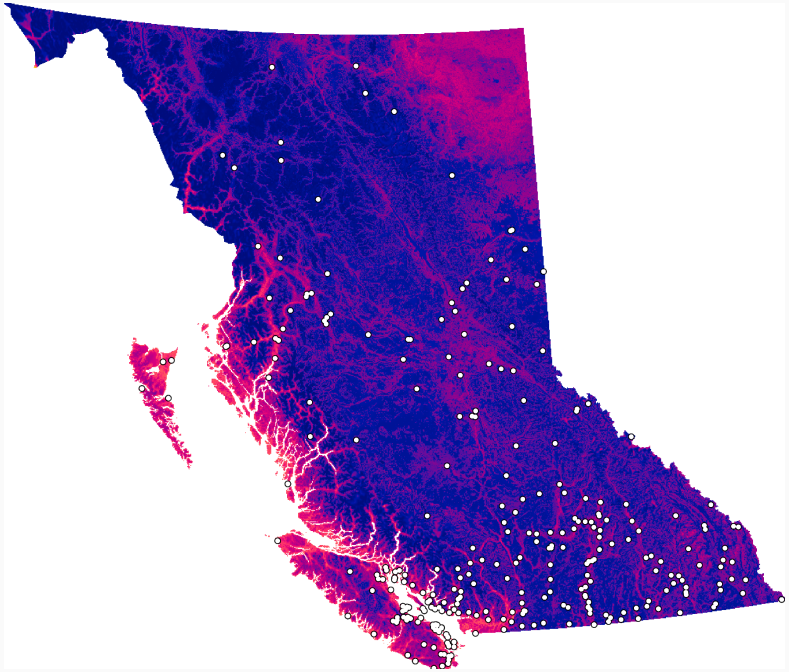
```
fit <- ppm(bei ~ elev + grad, data = bei.extra)
```

We can always increase the complexity of a model, but we often have no way of including additional covariates (can't generate the missing data *post hoc*).

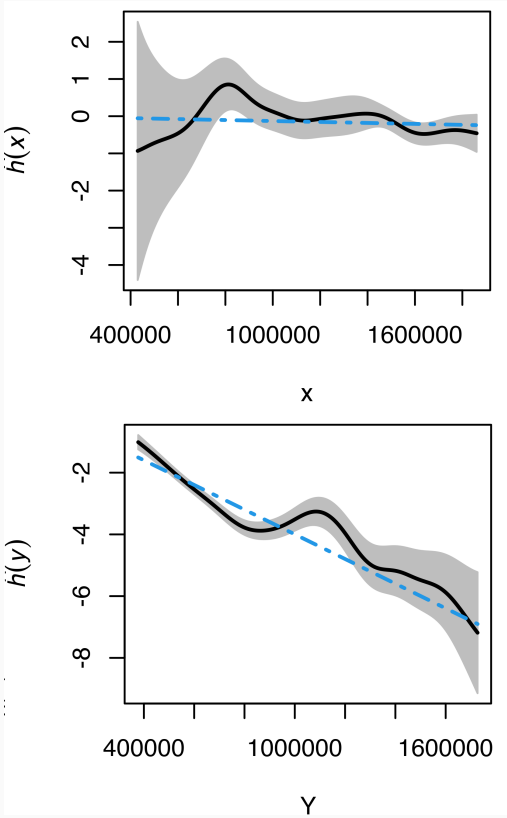
Coordinates as covariates cont.

With spatial data, we can include the Cartesian coordinates as covariates

A model predicting the distribution of BC Parks might over-predict in coastal and northern regions.



```
fit_coord <- ppm(parks_ppp ~ x + y,
                 data = DATA)
```



Coordinates as covariates cont.

Using the Cartesian coordinates as covariates is very useful when there are clear spatially varying trends in the data but no (or missing) covariates.

The process is unlikely to be responding to 'x' and 'y', so these are proxy variables (should aim to replace them if possible).

Because they are proxies, they can include the effects of multiple factors, so try not to overthink these terms (i.e., use them, but use them wisely).

Validation take home messages

Model selection techniques can tell us if one model outperforms another, but provide minimal information on the goodness of fit.

Model validation is a key step in any analysis. Because point process residuals are functionally identical to linear regression residuals, they can aid in the model validation process.

Model validation and refinement is an iterative process that is part science (based on theory), part art (based on experience), the more models you fit and diagnose, the better modeller you will be (not just PPPs!).

There are many additional diagnostic tools that we didn't cover (see chapters 10-11 in Baddeley *et al.* (2015) if you're interested).

Where to from here?

What we didn't cover

The methods we covered will get you through most standard spatial point pattern analyses, but there are a few things we didn't cover in detail:

1. Modifying the quadrature scheme via the `quadscheme()` function (section 9.8 in Baddeley *et al.*, 2015);

2. Cox processes (i.e., doubly stochastic models);

3. Gibbs processes (i.e., first and second moment models).

Cox point processes

The standard Poisson point process assumes there is some true and *fixed* $\lambda(u)$ value that we are estimating.

The Cox process generalises this by assuming that our local intensity is itself a random variable $\Lambda(u)$ (i.e., a stochastic process)

$$\mathbb{E}[n(\mathsf{X} \cup B)] = \mathbb{E}(\Lambda)|B|$$

Cox models allow for over-dispersion or clustering... but are difficult to fit.

Note: it's impossible to distinguish a Cox point process from a Poisson point process from only one realisation, and there are currently no model validation tools for these models.

Gibbs point processes

The standard Poisson point process assumes the intensity if only a function of first moment effects on $\lambda(u)$.

The Gibbs point process generalises this by explicitly defining interactions between points (i.e., second moment effects).

$$\lambda(u|\mathrm{x}) = \begin{cases} \beta & \text{if } u \text{ is permissible} \\ 0 & \text{if } u \text{ is not permissible} \end{cases}$$

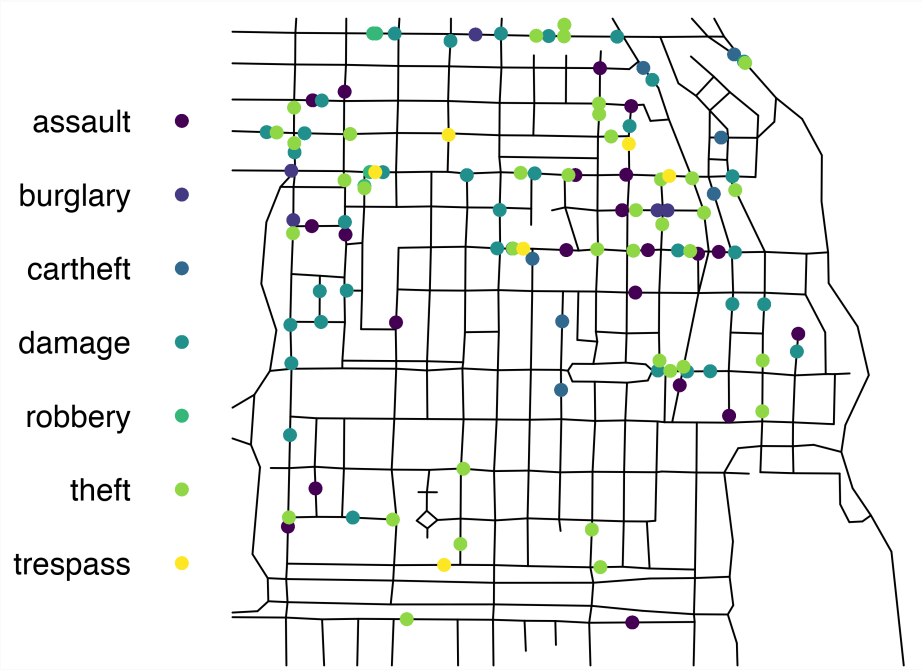
Gibbs models generally model avoidance or inhibition.

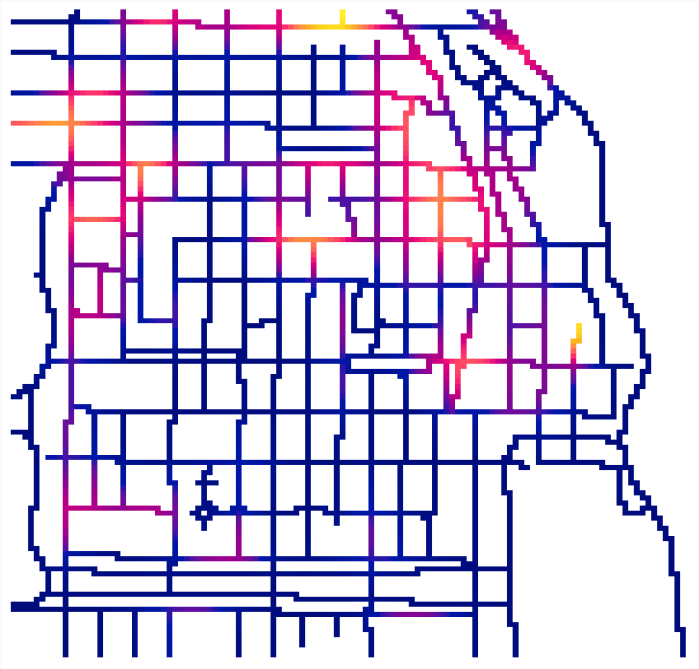
Fit via the `interaction` argument in `ppm` e.g., `ppm(parks ~ 1 + Strauss(10))`

Additional considerations

If you are dealing with points on a linear network, you will need a completely different set of functions (workflow is the same).

```
data("chicago")

lambda <- density(unmark(chicago), sigma= 60)

plot(lambda)

fit <- lppm(chicago ~ x + y)

...
```





Additional considerations cont.

We have mostly been ignoring the marks, but they can be included as covariates (exact workflow depends on structure of the marks and analytical goals, see Ch 14 in Baddeley *et al.*, 2015).

We have been working in two dimensions, but the methods extend to three dimensions (see Ch 15 in Baddeley *et al.*, 2015).

Tools for more complicated data structures (e.g., replicated data, spatial time series) are still being developed.

Take home messages

The methods we covered will get you through most standard spatial point pattern analyses.

A typical workflow begins with data visualisation, and calculating first and second moment descriptive statistics.

Once you have a good feeling of the properties of the data, building models to describe the system will allow you to make generalisable inference about the point process you're modelling.

The rest of the course will focus on situations where the locations of our data are arbitrary, but where the spatial context is still critical for understanding our system.

References

Baddeley, A., Rubak, E. & Turner, R. (2015). *Spatial point patterns: methodology and applications with R*. CRC press.

Sweeney, S. & Gómez-Antonio, M. (2016). Localization and industry clustering econometrics: An assessment of gibbs models for spatial point processes. *Journal of Regional Science*, 56, 257–287.