Inhomogeneous Poisson point process

Jaione Macicior and Olga Marín

2024-01-26

Introduction

In the paper it presents a way to model the first-order intensity of spatio-temporal point pattern data, considering the intensity as a parametric log-linear function of spatial, temporal, and spatio-temporal covariates. However, even though this method can really provide good and interpretable results, basing on the contents learnt in class, we will work only with spatial data, and we will try to fit a point process model in order to get predictions and some interpretations.

As far as the regularization is concerned, the dataset used has only two variables so variable selection will not be needed as we will see, so the regularization part is not implemented either.

For this demonstration the Beilschmedia data will be used.

Beilschmedia data

This dataset installed in the spatstat package represents a point pattern indicating the locations of Beilschmiedia trees. Additional covariate data are provided in a distinct object called bei.extra. This object is a list containing two entries: elev and grad. These entries correspond to pixel images that convey the values of terrain elevation and terrain slope, respectively.

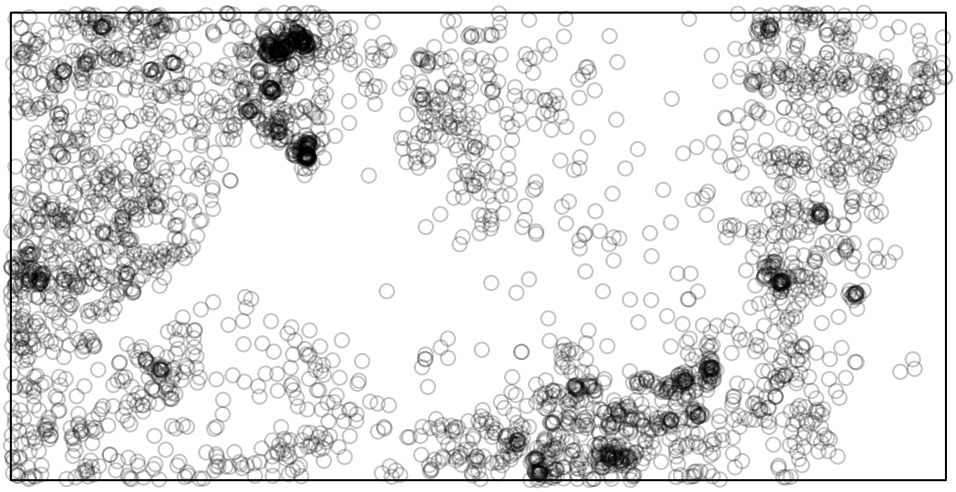
```
bei
```

```
## Planar point pattern: 3604 points
## window: rectangle = [0, 1000] x [0, 500] metres
```

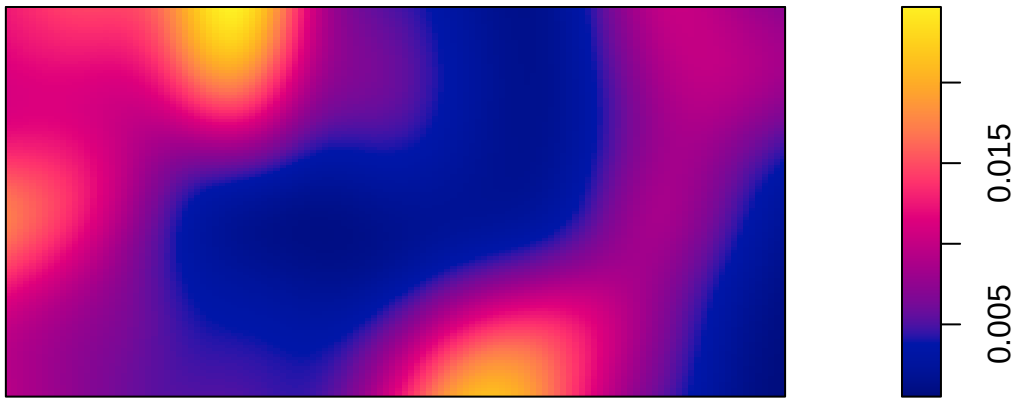
```
bei.extra
```

```
## List of pixel images
##
## elev:
## real-valued pixel image
## 101 x 201 pixel array (ny, nx)
## enclosing rectangle: [-2.5, 1002.5] x [-2.5, 502.5] metres
##
## grad:
## real-valued pixel image
## 101 x 201 pixel array (ny, nx)
## enclosing rectangle: [-2.5, 1002.5] x [-2.5, 502.5] metres
```

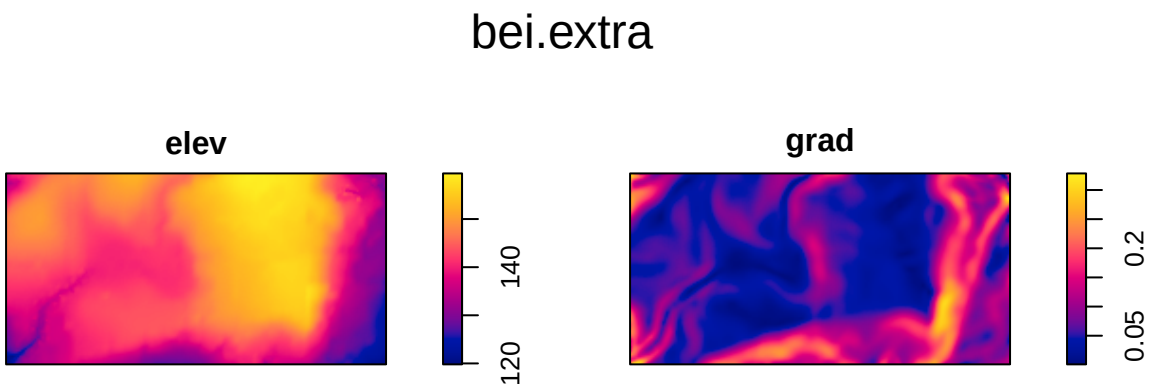
```
plot(bei)
```




```
plot(density(bei))
```




```
plot(bei.extra)
```



We apply the Kolmogorov-Smirnov test to see whether the dataset depends on the covariates:

```
cdf.test.ppp(bei, bei.extra$elev, test = "ks")
```

```
##
##  Spatial Kolmogorov-Smirnov test of CSR in two dimensions
##
## data:  covariate 'bei.extra$elev' evaluated at points of 'bei'
##      and transformed to uniform distribution under CSR
## D = 0.10629, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
cdf.test.ppp(bei, bei.extra$grad, test = "ks")
```

```
##
##  Spatial Kolmogorov-Smirnov test of CSR in two dimensions
##
## data:  covariate 'bei.extra$grad' evaluated at points of 'bei'
##      and transformed to uniform distribution under CSR
## D = 0.19484, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

As both p-values are nearly zero, the covariates can be used to fit the model of the dataset.

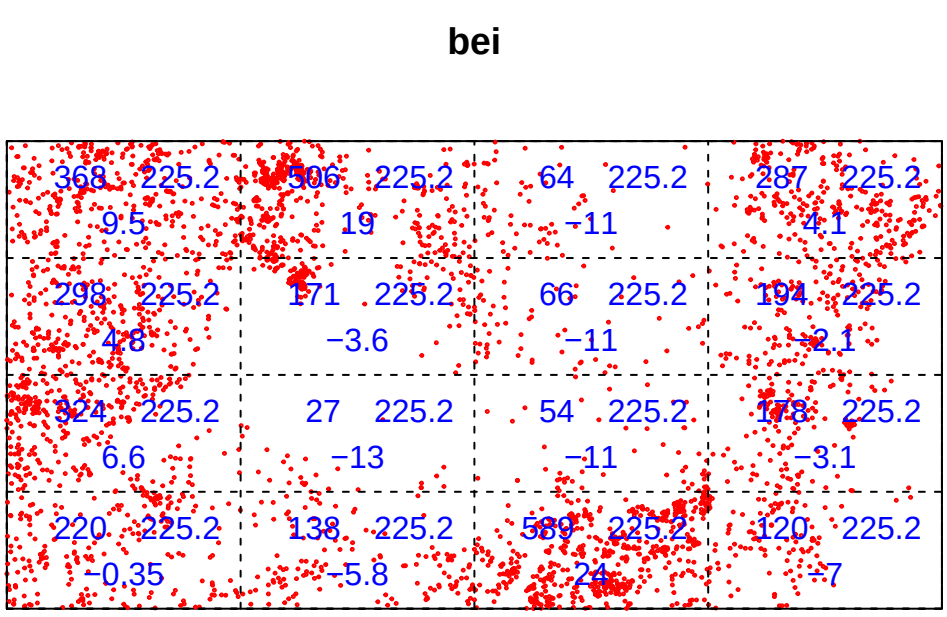
Clustering

Now we are going to see whether we can observe clustering in the data with the following test:

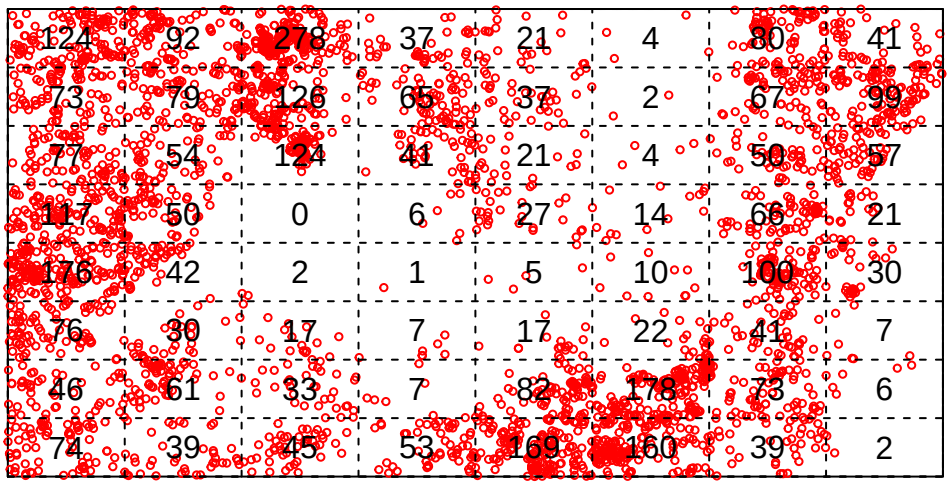
```
tx2 = quadrat.test(bei, nx = 4, ny = 4, alternative = 'clustered');tx2 #clustered
```

```
##
##  Chi-squared test of CSR using quadrat counts
##
## data:  bei
## X2 = 1754.6, df = 15, p-value < 2.2e-16
## alternative hypothesis: clustered
##
## Quadrats: 4 by 4 grid of tiles
```

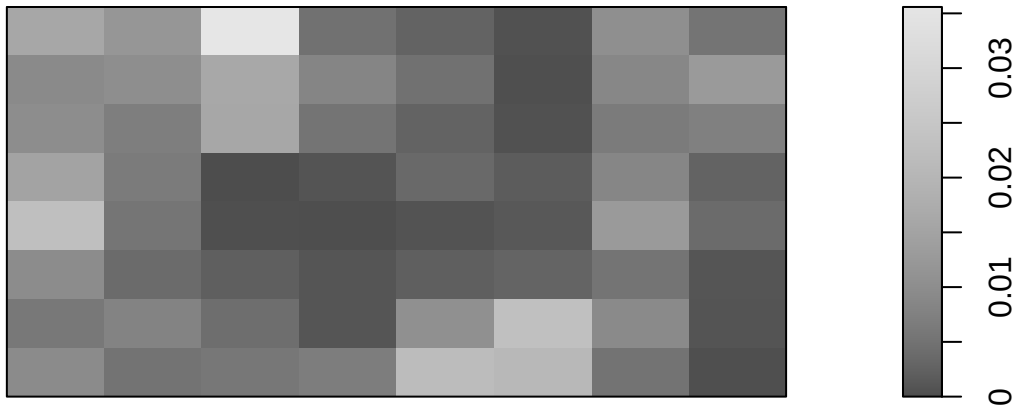
```
plot(bei, cols = "red", cex = 0.2)
plot(tx2, add = TRUE, col = "blue", lty = 2)
```



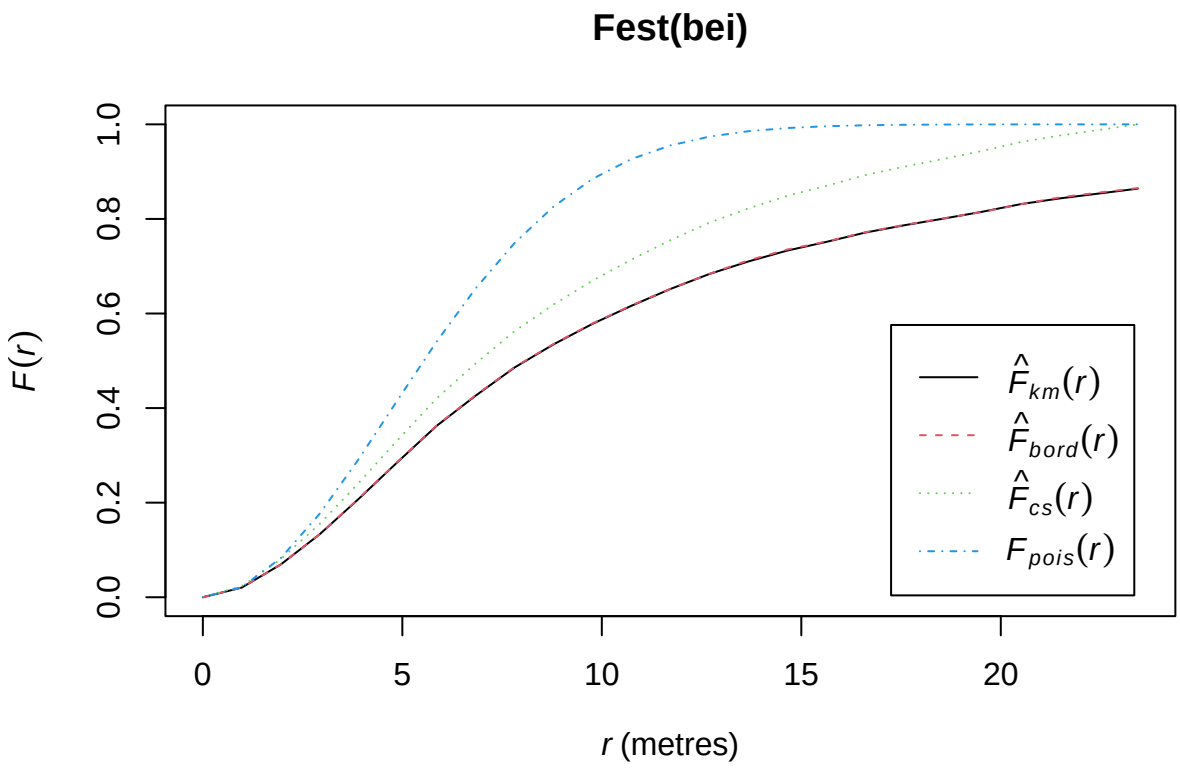

```
q1 = quadratcount(bei, nx = 8, ny = 8)
iq1 = intensity(q1, image = TRUE)

plot(bei, cols = "red", cex = 0.5)
plot(q1, add = TRUE, lty = 2)
```



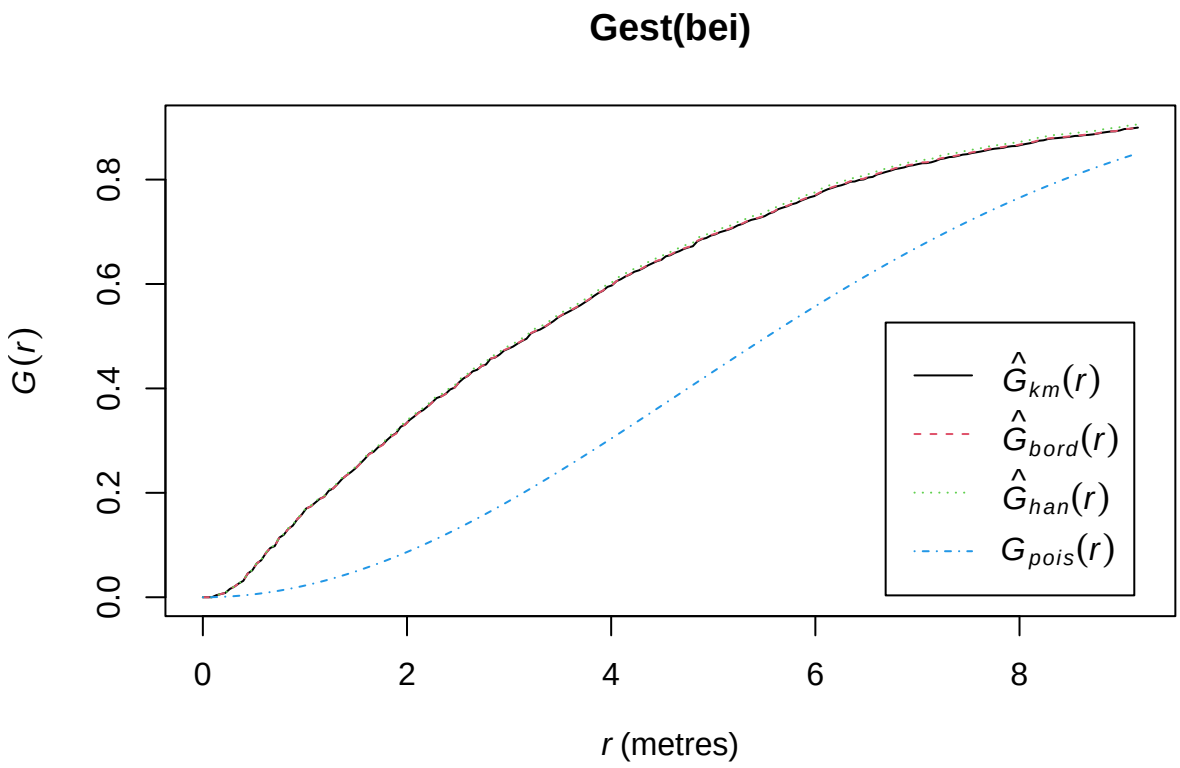

```
plot(iq1, col = grey.colors(12312))
```



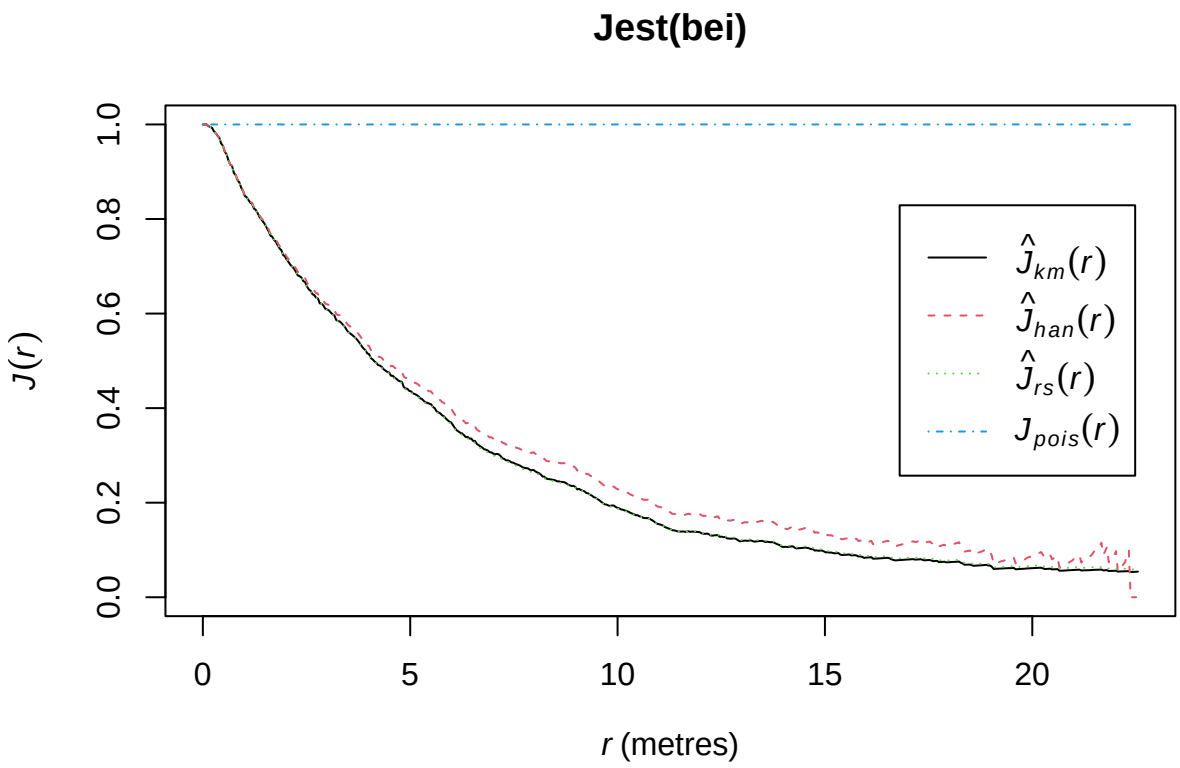

```
plot(Fest(bei)) #empty space function
```




```
plot(Gest(bei)) #nearest neighbourhood distribution
```

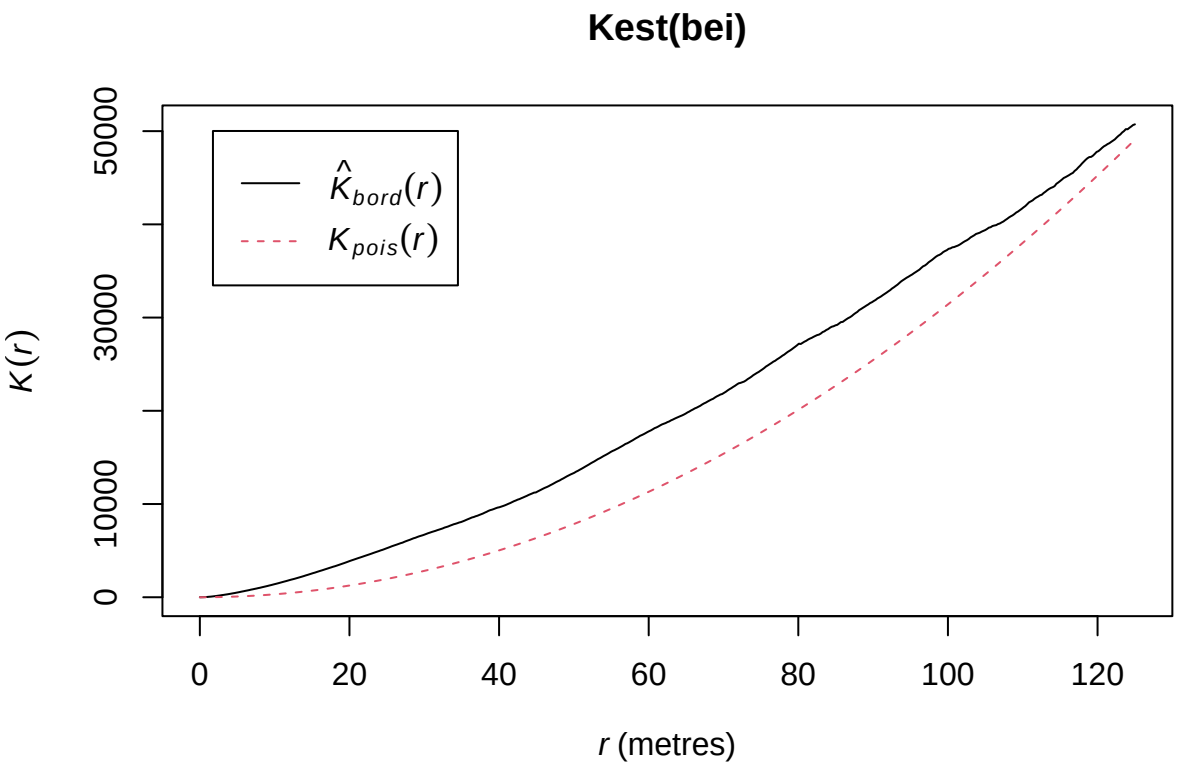



```
plot(Jest(bei))
```




```
plot(Kest(bei))
```

```
## number of data points exceeds 3000 - computing border correction estimate only
```




```
par(mfrow = c(2,2))

res1 = envelope(bei, Fest, nsim = 20)
```

```
## Generating 20 simulations of CSR  ...
## 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
## 20.
##
## Done.
```

```
plot(res1)

res2 = envelope(bei, Gest, nsim = 20)
```

```
## Generating 20 simulations of CSR  ...
## 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
## 20.
##
## Done.
```

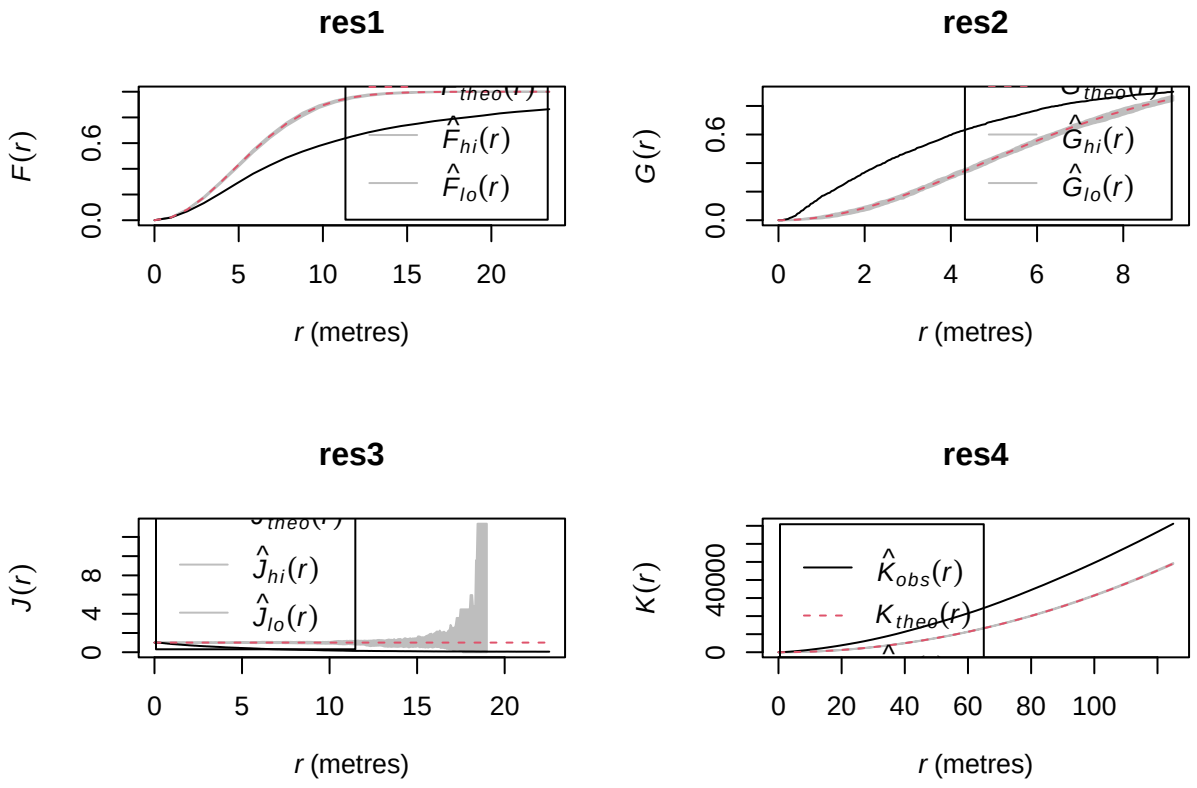
```
plot(res2)

res3 = envelope(bei, Jest, nsim = 20)
```

```
## Generating 20 simulations of CSR  ...
## 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
## 20.
##
## Done.
```

```
plot(res3)

res4 = envelope(bei, Kest, nsim = 20)
```

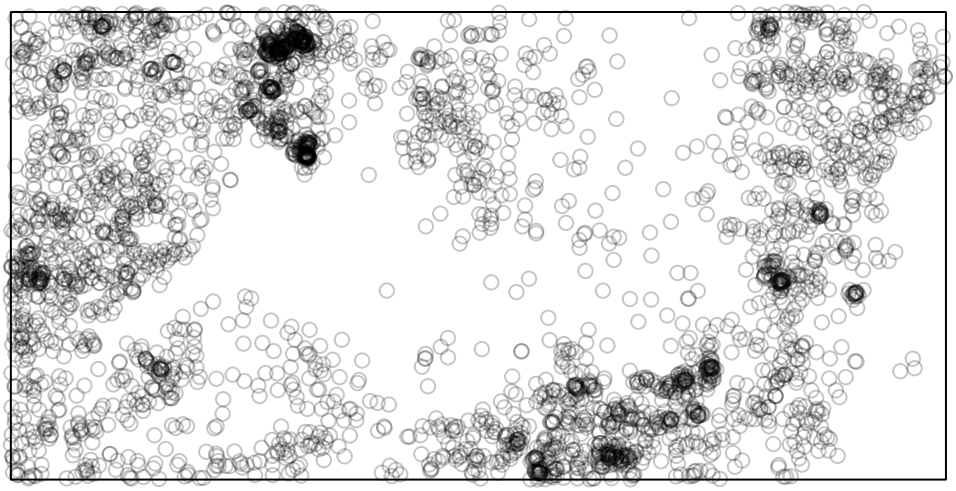
```
## Generating 20 simulations of CSR  ...
## 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
## 20.
##
## Done.
```

```
plot(res4)
```



This tests and plots indicate that there is clustering in the data as it can be observed:

```
plot(bei)
```



Fitting a Loglinear model

This is the general loglinear model:

$$\lambda_\beta(u) = \exp\left(B(u) + \beta^\top Z(u)\right) = \exp\left(B(u) + \beta_1 Z_1(u) + \beta_2 Z_2(u) + \ldots + \beta_p Z_p(u)\right)$$

where $B(u)$ and $Z_1(u), ..., Z_p(u)$ are known functions and $\beta_1, ..., \beta_p$ are parameters to be estimated.

Additive model

The Beilschmedia dataset has two numerical covariates: the terrain elevation and slope. The simplest loglinear model including these covariates is: $\lambda(u) = exp(\beta_0 + \beta_1 E(u) + \beta_2 S(u))$ where the coefficients β_i are the parameters to be estimated, and $E(u)$ and $S(u)$ are the terrain elevation in metres and slope at location u respectively. This model is an **additive model**.

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

```
## Nonstationary Poisson process
## Fitted to point pattern dataset 'bei'
##
## Log intensity:  ~elev + grad
##
```

```
## Fitted trend coefficients:
## (Intercept)        elev        grad
## -8.56355220  0.02143995  5.84646680
##
##                Estimate        S.E.     CI95.lo     CI95.hi Ztest       Zval
## (Intercept) -8.56355220 0.341113849 -9.23212306 -7.89498134   *** -25.104675
## elev         0.02143995 0.002287866  0.01695581  0.02592408   ***   9.371155
## grad         5.84646680 0.255781018  5.34514522  6.34778838   ***  22.857313
```

We obtain the following model:

$$bei = exp(-8.563 + 0.022 * elev + 5.846 * grad)$$

The interpretation of the fitted model is quite straightforward as both covariates are continuous numerical values. On a flat piece of land (slope zero) at sea level (elevation zero), the expected density of Beilschmiedia trees is $\exp(-8.563) = 0.000191$ trees per square meter.

With each additional meter of terrain elevation, the density increases by approximately 2.2% $\exp(0.022) = 1.022$. For every 0.1-unit increase in slope, the density increases by a factor of $\exp(5.846) = 1.793$. These effects are additive on the linear predictor scale: if elevation increases by 1 meter and slope increases by 0.1 units, the log density increases by the sum of the elevation effect (1×0.02141) and the slope effect (0.1×5.841).

```
min(bei.extra$grad); max(bei.extra$grad)
```

```
## [1] 0.0008663357
```

```
## [1] 0.3284767
```

```
min(bei.extra$elev); max(bei.extra$elev)
```

```
## [1] 119.81
```

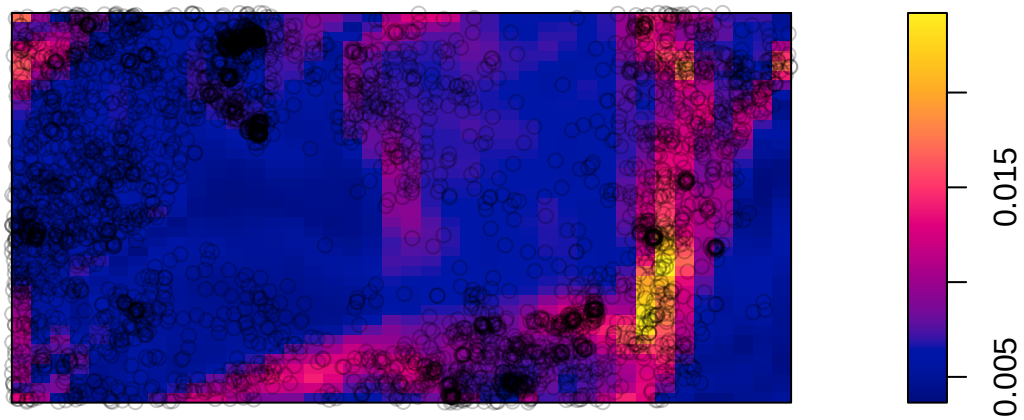
```
## [1] 159.48
```

To assess the relative 'importance' of the slope and elevation variables, we consider the range of values for each variable. The slope varies from nearly zero to its maximum value (max(grad) = 0.3285), leading to a slope effect variation by a factor of $\exp(5.841 \times (0.3285\text{-}0)) = 6.813$ between the flattest and steepest slopes. On the other hand, terrain elevation ranges from 119.8 to 159.5 meters, resulting in an elevation effect variation by a factor of $\exp(0.022 \times (159.47\text{-}119.81)) = 2.34$ between the lowest and highest elevations. Therefore, in terms of magnitude, the slope effect is more 'important.'

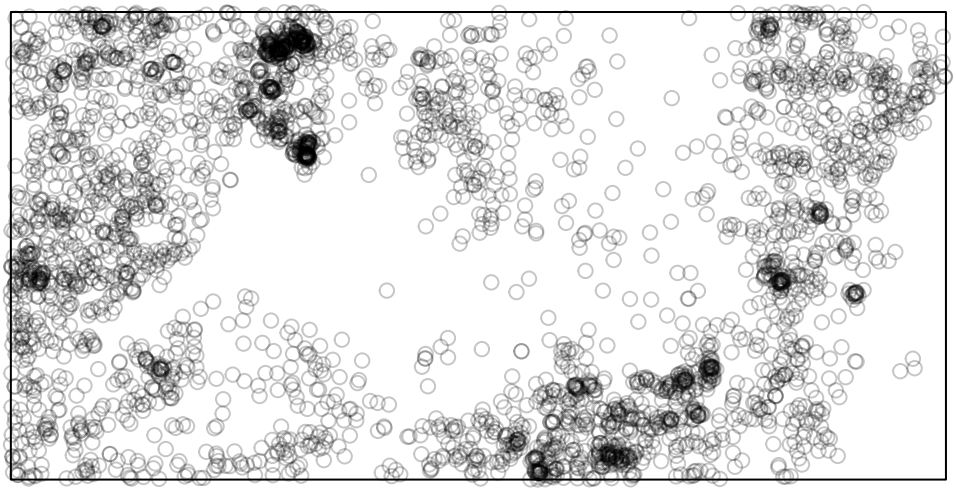
Statistical inference

We analyse the fitted model:

```
plot(model, how = 'image', se = FALSE)
```




```
plot(bei)
```




```
print(model)
```

```
## Nonstationary Poisson process
## Fitted to point pattern dataset 'bei'
##
## Log intensity:  ~elev + grad
##
## Fitted trend coefficients:
## (Intercept)        elev        grad
## -8.56355220  0.02143995  5.84646680
##
##                Estimate        S.E.     CI95.lo     CI95.hi Ztest       Zval
## (Intercept) -8.56355220 0.341113849 -9.23212306 -7.89498134   *** -25.104675
## elev         0.02143995 0.002287866  0.01695581  0.02592408   ***   9.371155
## grad         5.84646680 0.255781018  5.34514522  6.34778838   ***  22.857313
```

We can observe that all the parameters are clearly significant.

Standard error and confidence intervals These are the estimated standard errors of the individual parameter estimates:

```
sqrt(diag(vcov(model)))
```

```
## (Intercept)        elev        grad
## 0.341113849 0.002287866 0.255781018
```

```
confint(model)
```

```
##                   2.5 %      97.5 %
## (Intercept) -9.23212306 -7.89498134
## elev         0.01695581  0.02592408
## grad         5.34514522  6.34778838
```

Regarding the confidence intervals, the values of the estimated parameters may vary and this can change the value of the intensity, but overall the intervals are not too big.

The estimated correlation between individual parameter estimates can be useful in detecting collinearity and confounding:

```
co <- vcov(model, what="corr"); round(co,2)
```

```
##             (Intercept)  elev  grad
## (Intercept)        1.00 -1.00 -0.41
## elev              -1.00  1.00  0.34
## grad              -0.41  0.34  1.00
```

This suggests a strong negative correlation between the intercept parameter estimate and the estimate of the coefficients of elev. As far as the estimated coefficients of the covariates are concerned, they are not too correlated but thir correlation is positive.

```
step(model) # no change (?)
```

```
## Start:  AIC=42295.11
## ~elev + grad
##
##        Df   AIC
## <none>    42295
## - elev  1 42383
## - grad  1 42760
```

```
## Nonstationary Poisson process
## Fitted to point pattern dataset 'bei'
##
## Log intensity:  ~elev + grad
##
## Fitted trend coefficients:
## (Intercept)        elev        grad
## -8.56355220  0.02143995  5.84646680
##
##                Estimate        S.E.     CI95.lo     CI95.hi Ztest       Zval
## (Intercept) -8.56355220 0.341113849 -9.23212306 -7.89498134   *** -25.104675
## elev         0.02143995 0.002287866  0.01695581  0.02592408   ***   9.371155
## grad         5.84646680 0.255781018  5.34514522  6.34778838   ***  22.857313
```

```
stepAIC(model) # no change
```

```
## Start:  AIC=42295.11
## ~elev + grad
##
##        Df   AIC
## <none>    42295
## - elev  1 42383
## - grad  1 42760

## Nonstationary Poisson process
## Fitted to point pattern dataset 'bei'
##
## Log intensity:  ~elev + grad
##
## Fitted trend coefficients:
## (Intercept)        elev        grad
## -8.56355220  0.02143995  5.84646680
##
##                Estimate        S.E.     CI95.lo     CI95.hi Ztest       Zval
## (Intercept) -8.56355220 0.341113849 -9.23212306 -7.89498134   *** -25.104675
## elev         0.02143995 0.002287866  0.01695581  0.02592408   ***   9.371155
## grad         5.84646680 0.255781018  5.34514522  6.34778838   ***  22.857313
```

```
drop1(model) # no change
```

```
## Single term deletions
##
## Model:
## ~elev + grad
##        Df   AIC
## <none>    42295
## elev    1 42383
## grad    1 42760
```

```
AIC(model)
```

```
## [1] 42295.11
```

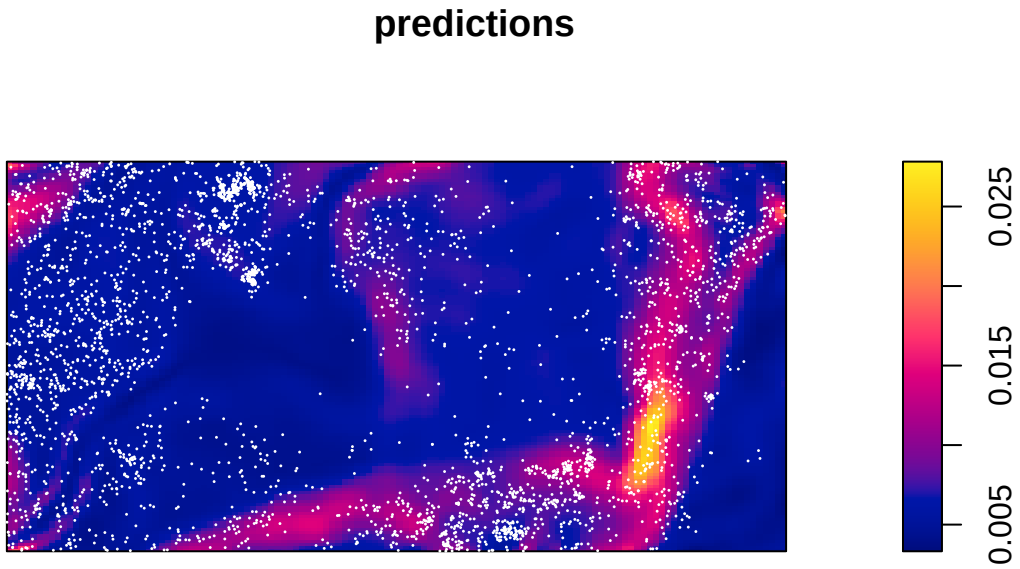
Regarding the variable selection methods, we can conclude that the two covariates are needed to fit the model.

Prediction

Intensity In any Poisson model with intensity $\lambda(u) = \lambda_\beta(u)$, where β represents a parameter vector, the estimated intensity or predicted intensity is denoted as $\lambda_{\hat{\beta}}(u)$. This is derived by substituting the fitted parameter estimates into the intensity formula.

```
# compute the fitted intensity at a regular grid
pred <- predict(model)
```

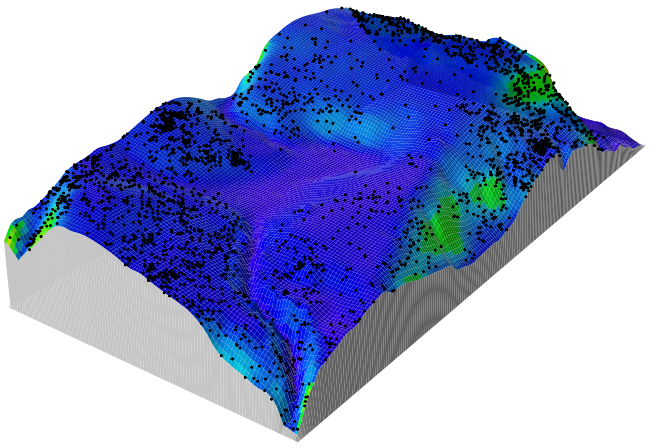
```
plot(pred, main = 'predictions')
plot(bei, add = TRUE, cols = 'white', cex = 0.2, pch = 16)
```



We can see that the model has predicted some clusters as the one in the right side of the plot. However, we also can observe that in the left of the plot there are a lot of trees but the intensity is quite low.

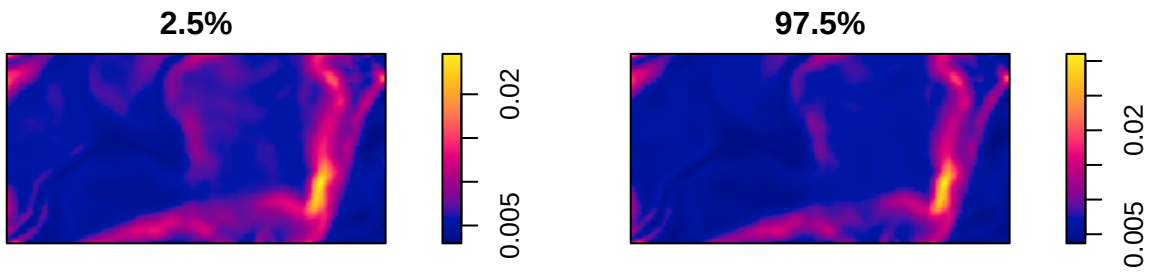
We can plot the results in 3D to see it more clearly:

```
M <- persp(bei.extra$elev, colin=pred, colmap=topo.colors, shade=0.4, theta=-55, phi=25, expand=6, box=

perspPoints(bei, Z=bei.extra$elev, M=M, pch=20, cex=0.1)
```



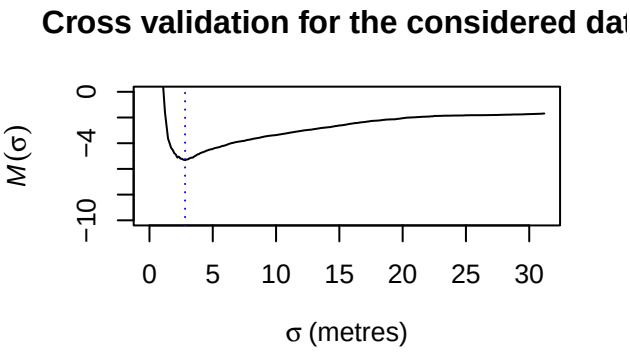
A confidence interval for the true value of $\lambda(u)$ at each location u can be computed by:

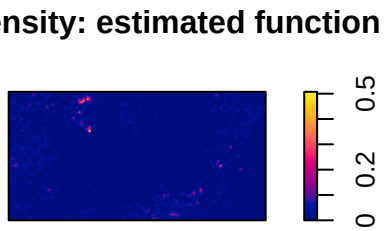
```
plot(predict(model, interval='confidence'), main = 'Interval confidence')
```

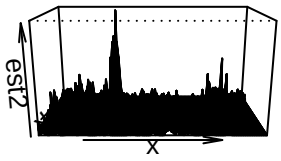


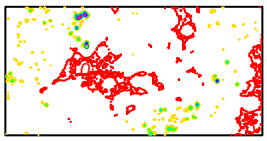
In point processes, kernel smoothing is a technique employed to estimate the underlying intensity function. Comparing predictions with kernel smoothing in point process models is like checking how well our model predicts events over time or space. Kernel smoothing gives us a smooth curve that represents the actual event patterns. By comparing our model's predictions to this smooth curve, we can see where our model does well and where it might struggle. It's a way to make sure our model is capturing the real trends and variations in the data, helping us trust the predictions it makes about when and where events will happen. We use Diggle's procedure in order to select a smoothing bandwidth for the kernel estimation of point process intensity using cross-validation.

```
opt.sg = bw.diggle(bei) #2.813112
par(mfrow = c(2,2))
plot(opt.sg, ylim = c(-10, 0), main = "Cross validation for the considered data")
est2 = density(bei, sigma = opt.sg)
plot(est2, main = "Intensity: estimated function")
persp(est2, main = "Estimated intensity: perspective", col = "green")
contour(est2, main = "Estimated intensity: contour", col = rainbow(7))
```

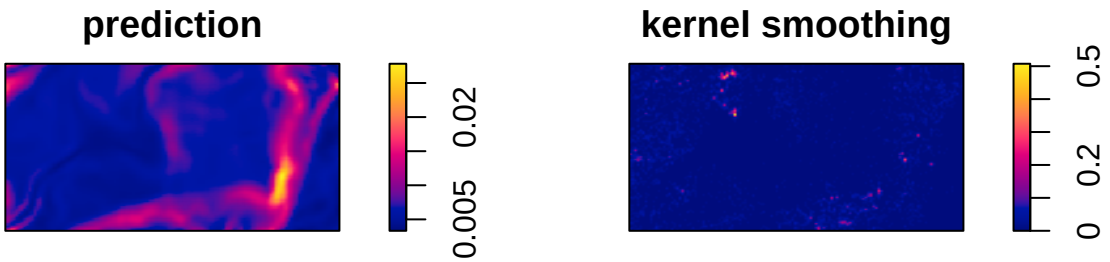









```
l <- solist(pred, est2)
plot(l, equal.ribbon = FALSE, main = "",
     main.panel = c("prediction", "kernel smoothing"))
```



We can conclude that kernel smoothing does not provide a lot of information as the values of the intensity are very similar.

Number of trees in a region We predict the expected number of trees in a region B as $\hat{\mu}(B) = \int_B \hat{\lambda}(u)du$. To find the expected number of trees at elevations below an altitude. As expected, as the minimum value of the covariate is 119,81, the expected number of trees will be zero if x is less than 119,81.

```
x <- 119.81
# expected number of trees
B <- levelset(bei.extra$elev, x)

predict(model, B, type="count")
```

```
## [1] 0.1454755
```

```
# confidence interval
predict(model, B, type="count", se=TRUE, interval='confidence')
```

```
## $confidence
##      2.5%     97.5%
## 0.1287352 0.1622157
##
## $se
## [1] 0.008541091
```

```r
# prediction interval
predict(model, B, type="count", se=TRUE, interval='prediction')
```

```
## $prediction
##  2.5% 97.5%
##     0     0
##
## $se
## [1] 0.008541091
```

With this dataset this is not very useful but in other contexts, for instance, if the point pattern is a record of accidents in a year, this can be useful: if we think about 'repeating the experiment' over time, there's a 95% chance that the number of accidents next year in the same area will be within the calculated limits.

Numer of trees in tiles We also can predict the number of trees

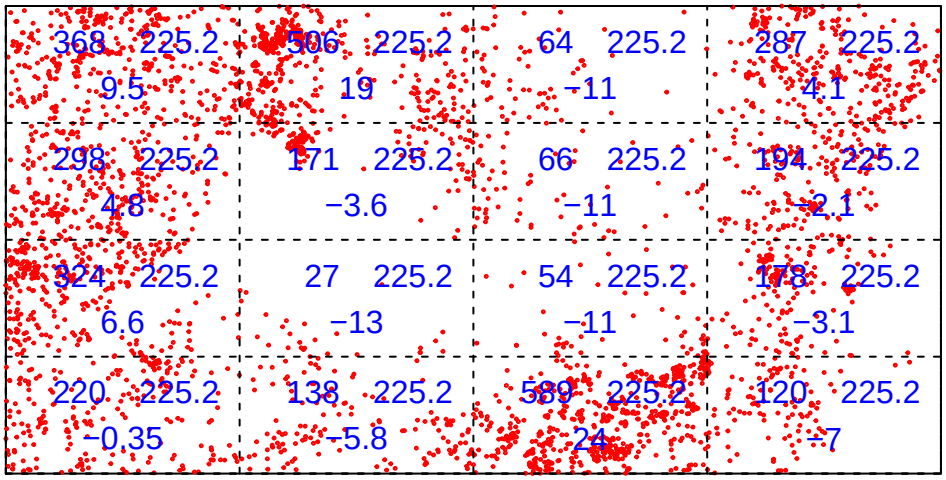
```r
pred <- predict(model, window=quadrats(bei, 4), type="count")
pred
```

```
## Tile row 1, col 1 Tile row 1, col 2 Tile row 1, col 3 Tile row 1, col 4
##          226.6723          218.8753          219.4225          284.4650
## Tile row 2, col 1 Tile row 2, col 2 Tile row 2, col 3 Tile row 2, col 4
##          182.4285          177.4767          208.1796          263.6230
## Tile row 3, col 1 Tile row 3, col 2 Tile row 3, col 3 Tile row 3, col 4
##          173.0021          157.9391          208.0413          300.3089
## Tile row 4, col 1 Tile row 4, col 2 Tile row 4, col 3 Tile row 4, col 4
##          192.9246          251.9437          270.1201          268.2639
```

```r
tx2 = quadrat.test(bei, nx = 4, ny = 4);tx2 #clustered
```

```
##
##  Chi-squared test of CSR using quadrat counts
##
## data:  bei
## X2 = 1754.6, df = 15, p-value < 2.2e-16
## alternative hypothesis: two.sided
##
## Quadrats: 4 by 4 grid of tiles
```

```r
plot(bei, cols = "red", cex = 0.2)
plot(tx2, add = TRUE, col = "blue", lty = 2)
```



Significance of the covariates in the model

Even though we applied some methods to perform variable selection, it can be useful to test the significance of each term in the model. To do that, we can apply a *significance test* comparing a null model and the model including that covariate. For that, **analysing of deviance** will be used.

```
m0 <- ppm(bei ~ 1) # null model
m1 <- ppm(bei ~ grad, data = bei.extra) # model with the covariate

anova(m0, m1, test= 'LR') # significance test
```

Grad

```
## Analysis of Deviance Table
##
## Model 1: ~1   Poisson
## Model 2: ~grad    Poisson
##   Npar Df Deviance  Pr(>Chi)
## 1    1
## 2    2  1   383.11 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The p-value is effectively zero, indicating that the covariate is significative in the model.

Elev For testing the significance of this variable, this covariate will be added to the prior models, and then the same test will be applied:

```
m0 <- ppm(bei~elev, data = bei.extra)
m1 <- update(m1, .~.+elev)

anova(m0, m1, test= 'LR') # significance test
```

```
## Analysis of Deviance Table
##
## Model 1: ~elev    Poisson
## Model 2: ~grad + elev     Poisson
##   Npar Df Deviance  Pr(>Chi)
## 1    2
## 2    3  1   467.38 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

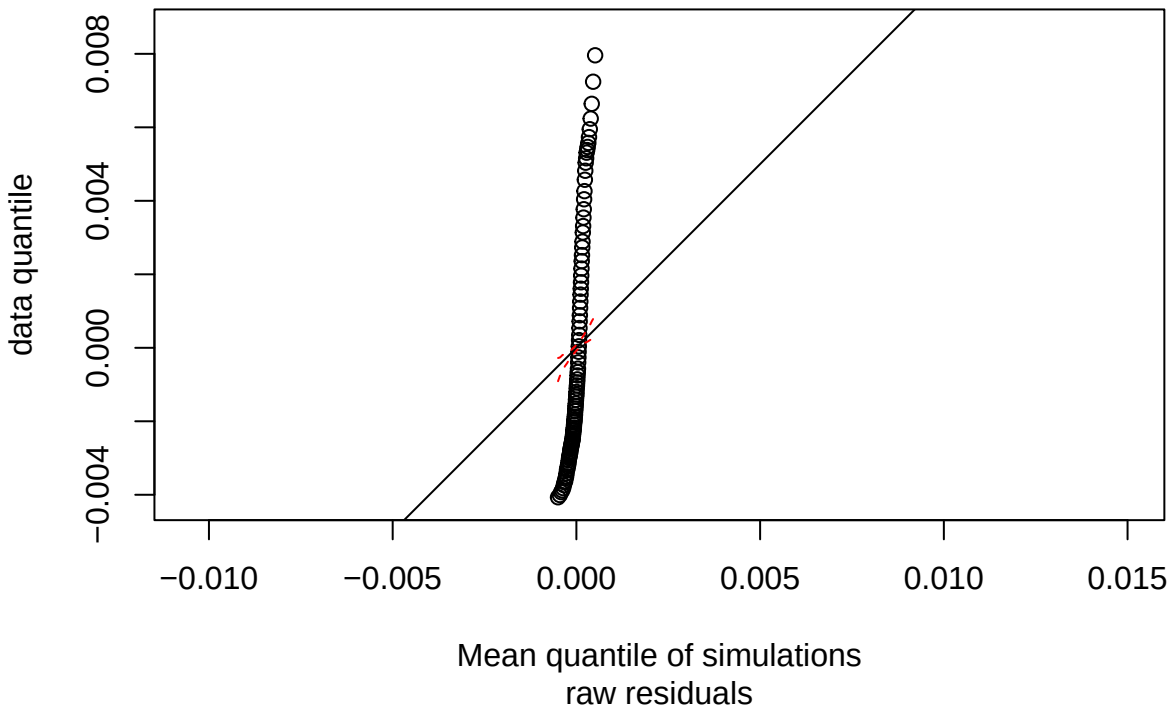
We conclude that as the p-value os technically zero, this covariate is also significative.

Validation of the model

```
rA = residuals(model, type = "raw")
```

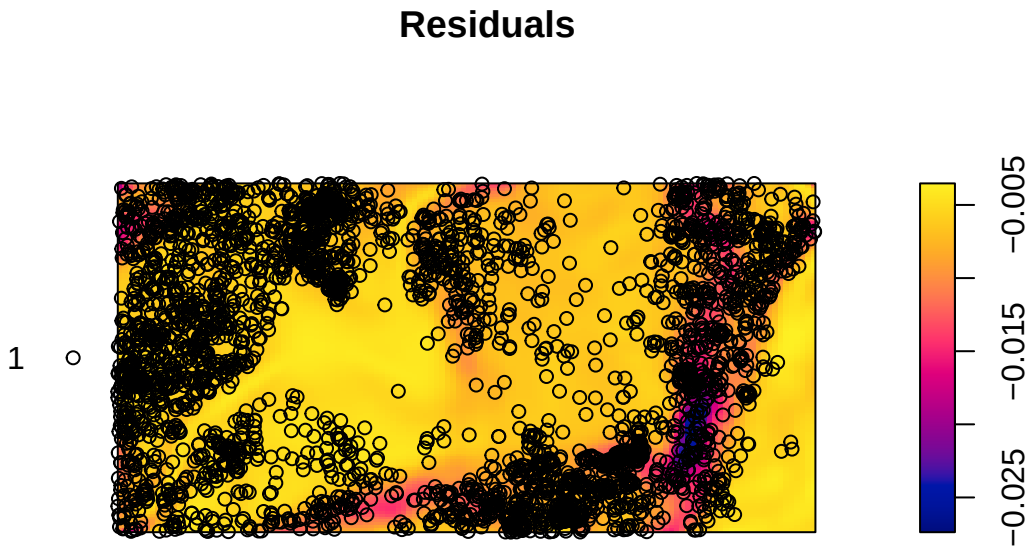
```
qA = qqplot.ppm(model, type = "raw")
```

```
## Extracting model information...Evaluating trend...done.
## Simulating 100 realisations... 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
## 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,
## 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60,
## 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80,
## 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99,
## 100.
##
## Diagnostic info:
##  simulated patterns contained an average of 3608.9 points.
## Calculating quantiles...averaging.....Done.
```



The Q-Q plot for the residuals is a straight line with all points lying on the 0 which is an indication that the distribution of residuals are consistent with the theoretical distribution that we are comparing it to. Besides, it could also mean that the residuals are similar from each other.

```
plot(rA, main = "Residuals")
```



We can observe that the residuals are closer to zero when there are less trees even though when there are clusters they are not that big.

In conclusion, we can validate the model as the residuals have a desirable distribution.