Species distribution modelling with Poisson point processes

Ian Flint and Roozbeh Valavi

2020-02-18

Species distribution modelling

Envrionmental data

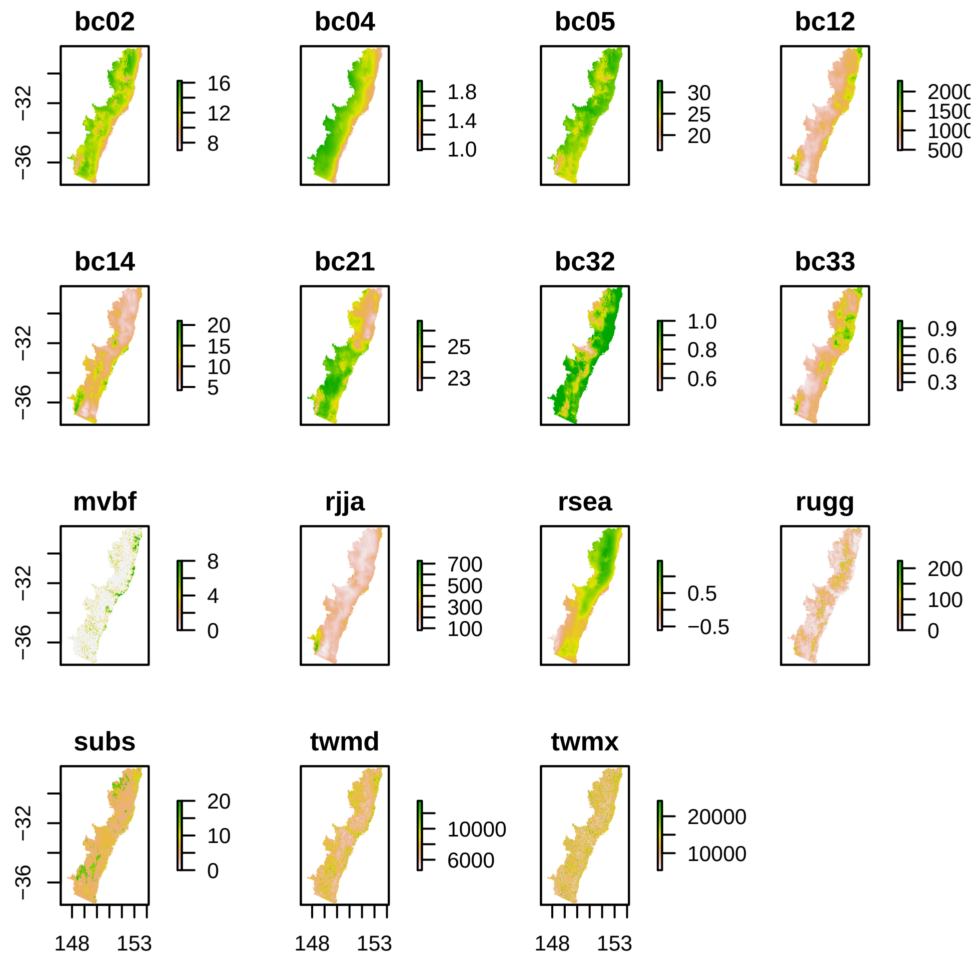
```
# library(maptools)
library(raster)
library(sf)

grid_dir <- "data/grids"
vars <- list.files(grid_dir, pattern = ".tif$", full.names = TRUE)

# the border shapefile
ibra <- st_read("data/ibraone.shp", crs = 4283, quiet = TRUE)

# read the raster layers as a raster stack
nsw_stack <- stack(vars)
# set the coordinate system
crs(nsw_stack) <- CRS("+init=epsg:4283")

plot(nsw_stack)
```



Species data

The species data comes form Fithian and others (2015).

All the species starts with `euca` ...

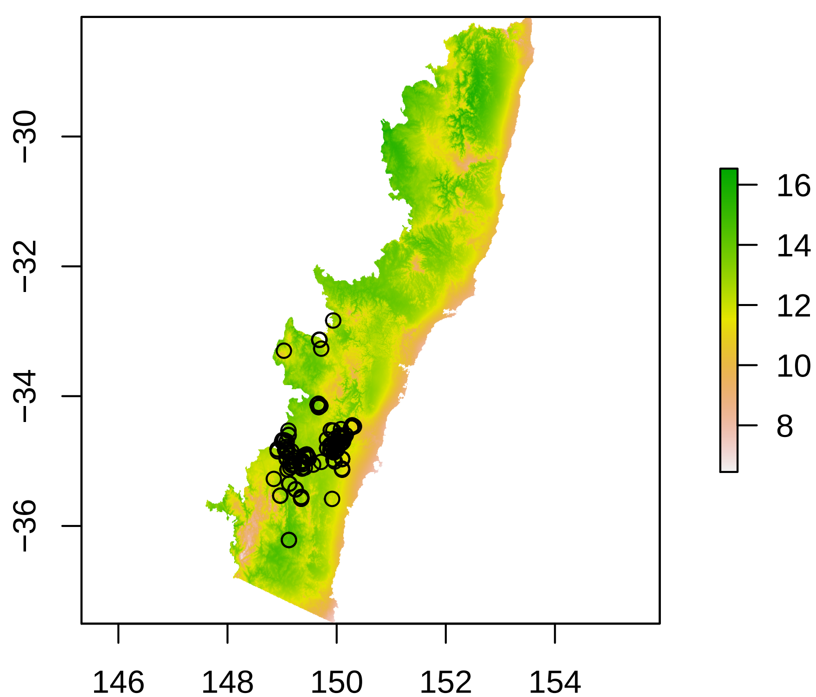
```r
# load Fithian et al (2015) species data
load("data/moddat.RData")

head(eucacine_moddat)
```

```
##   id     long       lat year accuracy  longlat   cells     bc01     bc02
## 1  1 150.0862 -34.60553 2000      100 15008622 6279811 13.63983 12.23118
## 2  2 150.2854 -34.44967 2000      100 15028535 6126927 13.24499 11.36682
## 3  3 150.0125 -34.80654 2000      100 15001250 6474021 13.71813 12.41187
## 4  4 150.0997 -34.96775 2000      100 15009972 6631876 13.09869 11.85747
## 5  5 149.8980 -34.85114 2007      100 14989800 6517680 13.29461 12.52573
## 6  6 149.7147 -35.01311 1999      100 14971468 6675426 12.81009 12.73738
##       bc04     bc05       bc06     bc12      bc14     bc21      bc28      bc32
## 1 1.645262 27.27752  1.09317458 693.4518  9.081451 26.44496 0.5211285 0.7648971
## 2 1.596420 26.22510  1.17330492 780.2014 10.506531 25.97984 0.6106081 0.8590701
## 3 1.652387 27.46763  1.06588304 681.5583  9.605835 26.38567 0.5021659 0.7421684
## 4 1.613784 26.41990  0.87655205 788.7050 10.496765 25.96847 0.5949574 0.8564287
```

```
## 5 1.699777 27.45464  0.58909386 653.2943  9.165375 26.40723 0.4897964 0.7353666
## 6 1.749456 27.40740 -0.01114328 677.7914 10.394196 26.34236 0.5267761 0.8032433
##        bc33      d2r      d2t     elev ibgd ibsb mvbf     rdjf     rjja
## 1 0.3000291   0.0000 76729.03 590.4653   98  394    3 188.9394 144.8310
## 2 0.3601428   0.0000 55561.12 692.9059   93  384    0 214.0642 160.9149
## 3 0.2900780   0.0000 91516.44 550.2057   98  394    0 183.8921 140.0840
## 4 0.3360529   0.0000 91472.76 639.3138   98  394    0 206.9004 166.6616
## 5 0.2770475 742.9186 84018.19 623.7888   98  394    0 175.7002 134.8407
## 6 0.2836685   0.0000 59406.70 696.1747   98  392    3 177.5476 147.8208
##        rsea      rugg sandsubs     slpe stre subs twmd  twmx      wpot xveg
## 1 0.2742345 10.927129        0 1.216894 0.76    5 6260  7542 -1.288309    0
## 2 0.2579768 18.860123        0 3.167662 0.00    7 8023 11071 -1.171590    0
## 3 0.2861401 19.935825        0 1.852355 0.00    5 7474  9066 -1.308515    1
## 4 0.2572912 15.468473        0 2.161766 0.00    5 7278  9110 -1.118788    1
## 5 0.2735702  8.366111        0 1.390608 1.00   15 8569 13911 -1.360653    0
## 6 0.2330509  5.964362        0 1.133822 0.00    6 8172 10061 -1.218756    1
```

```r
plot(nsw_stack[[1]])
points(eucacine_moddat$long, eucacine_moddat$lat)
```



Modelling with Maxent

Modelling with point processes

The spatstat package is the most widely used to work with point processes. Covariates are usually specified in their image objects spatstat::im. Internally, this is represented as a large pixel matrix, so conversion from rasters and other image objects is usually straightforward.

```
library(spatstat)

covariates <- lapply(as.list(nsw_stack), function(element) maptools::as.im.RasterLayer(element))
names(covariates) <- names(nsw_stack)
```

Spatstat also needs to be told what the observation region is. The required object type is spatstat::owin. Common ways to construct an owin is to either take a fixed rectangle, i.e. window <- owin(c(0, 100), c(0, 100)), or to use an existing covariate or raster to construct the window. The latter technique is what we will use here.

Although it would be possible to do window <- spatstat::as.owin(covariates[[1]]), it will be easier to work on a window with a lower resolution, as shown next.

```
window <- spatstat::as.owin(as.mask(covariates[[1]], eps = 0.01))
```

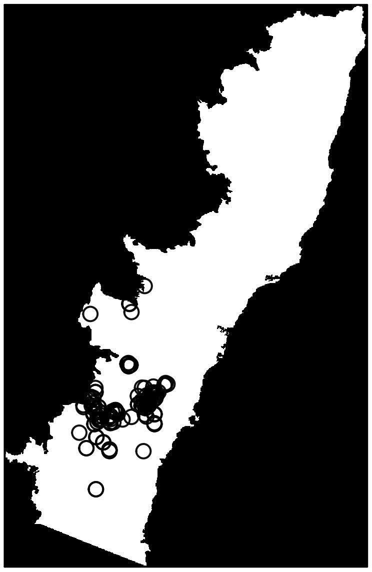
Locations of individuals are represented via a point pattern object spatstat::ppp, and consist in coordinates along with a window in which the species has been observed.

```
configuration <- spatstat::ppp(x = eucacine_moddat$long, y = eucacine_moddat$lat,
    window = window)
```

Point patterns can easily be plotted.

```
plot(configuration)
```

configuration

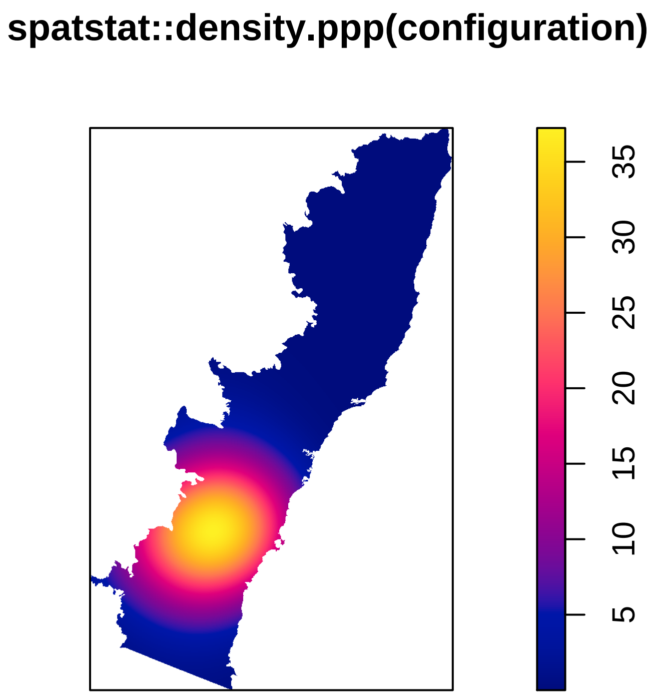


It is usually a good idea to start by a "static'' analysis of the point pattern, without yet involving covariates.

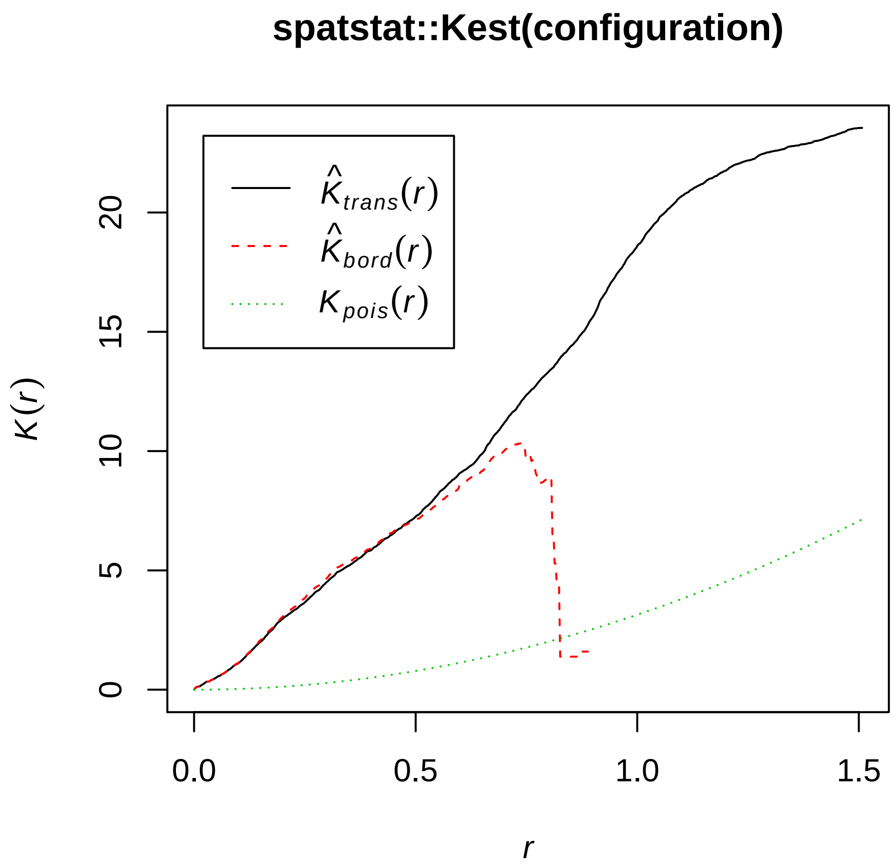
```
summary(configuration)
```

```
## Planar point pattern:  171 points
## Average intensity 9.396613 points per square unit
##
## Coordinates are given to 6 decimal places
##
## binary image mask
## 935 x 604 pixel array (ny, nx)
## pixel size: 0.00998 by 0.01 units
## enclosing rectangle: [147.6075, 153.6375] x [-37.505, -28.1575] units
##                      (6.03 x 9.348 units)
## Window area = 18.198 square units
## Fraction of frame area: 0.323
```

```
plot(spatstat::density.ppp(configuration))
```




```
plot(spatstat::Kest(configuration))
```




```
# The line below takes 3 min to execute and is not crucial to the analysis.
# plot(spatstat::envelope(configuration,Kest))
```

Doing inference on the point pattern is just as easy as setting up a `glm` regression. Start by writing the formula, essentially `formula <- "configuration ~ covariates`

```
formula <- paste0("configuration ~ ", paste0(names(covariates), collapse = " + "))
print(formula)
```

```
## [1] "configuration ~ bc02 + bc04 + bc05 + bc12 + bc14 + bc21 + bc32 + bc33 + mvbf + rjja + rsea + ru
```

The fitting function (analogue of `glm`) is `spatstat::ppm` and is used as follows.

```
fit <- spatstat::ppm(as.formula(formula), covariates = covariates)
```

The fitted regression is manipulated in the same way as a `glm` fit is, so for example you can have a look at the summary

```
summary(fit)
```

```
## Point process model
## Fitting method: maximum likelihood (Berman-Turner approximation)
## Model was fitted using glm()
## Algorithm converged
## Call:
## ppm.formula(Q = as.formula(formula), covariates = covariates)
```

```
## Edge correction: "border"
##  [border correction distance r = 0 ]
## --------------------------------------------------------------------------------
## Quadrature scheme (Berman-Turner) = data + dummy + weights
##
## Data pattern:
## Planar point pattern:  171 points
## Average intensity 9.4 points per square unit
## binary image mask
## 935 x 604 pixel array (ny, nx)
## pixel size: 0.00998 by 0.01 units
## enclosing rectangle: [147.6075, 153.6375] x [-37.505, -28.1575] units
##                      (6.03 x 9.348 units)
## Window area = 18.198 square units
## Fraction of frame area: 0.323
##
## Dummy quadrature points:
##      151 x 55 grid of dummy points, plus 4 corner points
##      dummy spacing: 0.03993377 x 0.16995455 units
##
## Original dummy parameters: =
## Planar point pattern:  2968 points
## Average intensity 163 points per square unit
## binary image mask
## 935 x 604 pixel array (ny, nx)
## pixel size: 0.00998 by 0.01 units
## enclosing rectangle: [147.6075, 153.6375] x [-37.505, -28.1575] units
##                      (6.03 x 9.348 units)
## Window area = 18.198 square units
## Fraction of frame area: 0.323
## Quadrature weights:
##      (counting weights based on 151 x 55 array of rectangular tiles)
## All weights:
##  range: [9.98e-05, 0.00679]  total: 18.2
## Weights on data points:
##  range: [0.000566, 0.00339]  total: 0.279
## Weights on dummy points:
##  range: [9.98e-05, 0.00679]  total: 17.9
## --------------------------------------------------------------------------------
## FITTED MODEL:
##
## Nonstationary Poisson process
##
## ---- Intensity: ----
##
## Log intensity: ~bc02 + bc04 + bc05 + bc12 + bc14 + bc21 + bc32 + bc33 + mvbf +
## rjja + rsea + rugg + subs + twmd + twmx
## Model depends on external covariates 'bc02', 'bc04', 'bc05', 'bc12', 'bc14',
## 'bc21', 'bc32', 'bc33', 'mvbf', 'rjja', 'rsea', 'rugg', 'subs', 'twmd' and
## 'twmx'
## Covariates provided:
##  bc02: im
##  bc04: im
##  bc05: im
```

```
##  bc12: im
##  bc14: im
##  bc21: im
##  bc32: im
##  bc33: im
##  mvbf: im
##  rjja: im
##  rsea: im
##  rugg: im
##  subs: im
##  twmd: im
##  twmx: im
##
## Fitted trend coefficients:
##   (Intercept)          bc02          bc04          bc05          bc12
## -6.973843e+01 -7.532535e-01  9.769080e+00 -7.456425e-01  2.930845e-02
##          bc14          bc21          bc32          bc33          mvbf
##  4.374920e-02  3.496929e+00 -6.106670e+00 -2.648364e+01 -1.296194e-01
##          rjja          rsea          rugg          subs          twmd
## -6.709563e-02 -4.212680e+00 -7.344646e-03 -2.012431e-02 -1.646928e-04
##          twmx
##  3.709840e-05
##
##                  Estimate         S.E.       CI95.lo       CI95.hi Ztest
## (Intercept) -6.973843e+01 1.998915e+01 -1.089164e+02 -3.056042e+01   ***
## bc02        -7.532535e-01 2.662334e-01 -1.275061e+00 -2.314456e-01    **
## bc04         9.769080e+00 2.159955e+00  5.535646e+00  1.400251e+01   ***
## bc05        -7.456425e-01 1.182665e-01 -9.774406e-01 -5.138443e-01   ***
## bc12         2.930845e-02 3.719012e-03  2.201932e-02  3.659758e-02   ***
## bc14         4.374920e-02 1.187564e-01 -1.890091e-01  2.765075e-01
## bc21         3.496929e+00 6.324668e-01  2.257317e+00  4.736541e+00   ***
## bc32        -6.106670e+00 2.628885e+00 -1.125919e+01 -9.541505e-01     *
## bc33        -2.648364e+01 8.499642e+00 -4.314263e+01 -9.824648e+00    **
## mvbf        -1.296194e-01 5.709847e-02 -2.415303e-01 -1.770846e-02     *
## rjja        -6.709563e-02 1.322165e-02 -9.300958e-02 -4.118167e-02   ***
## rsea        -4.212680e+00 3.299336e+00 -1.067926e+01  2.253901e+00
## rugg        -7.344646e-03 6.012075e-03 -1.912810e-02  4.438804e-03
## subs        -2.012431e-02 2.024376e-02 -5.980134e-02  1.955272e-02
## twmd        -1.646928e-04 7.891568e-05 -3.193647e-04 -1.002095e-05     *
## twmx         3.709840e-05 2.972548e-05 -2.116248e-05  9.535928e-05
##                   Zval
## (Intercept) -3.4888147
## bc02        -2.8292972
## bc04         4.5228164
## bc05        -6.3047626
## bc12         7.8807082
## bc14         0.3683944
## bc21         5.5290312
## bc32        -2.3229128
## bc33        -3.1158536
## mvbf        -2.2701030
## rjja        -5.0746794
## rsea        -1.2768264
## rugg        -1.2216492
```

```
## subs        -0.9940996
## twmd        -2.0869470
## twmx         1.2480333
##
## ----------- gory details -----
##
## Fitted regular parameters (theta):
##   (Intercept)          bc02          bc04          bc05          bc12
## -6.973843e+01 -7.532535e-01  9.769080e+00 -7.456425e-01  2.930845e-02
##          bc14          bc21          bc32          bc33          mvbf
##  4.374920e-02  3.496929e+00 -6.106670e+00 -2.648364e+01 -1.296194e-01
##          rjja          rsea          rugg          subs          twmd
## -6.709563e-02 -4.212680e+00 -7.344646e-03 -2.012431e-02 -1.646928e-04
##          twmx
##  3.709840e-05
##
## Fitted exp(theta):
##  (Intercept)         bc02         bc04         bc05         bc12         bc14
## 5.163991e-31 4.708322e-01 1.748467e+04 4.744294e-01 1.029742e+00 1.044720e+00
##         bc21         bc32         bc33         mvbf         rjja         rsea
## 3.301390e+01 2.227958e-03 3.149928e-12 8.784297e-01 9.351058e-01 1.480664e-02
##         rugg         subs         twmd         twmx
## 9.926823e-01 9.800768e-01 9.998353e-01 1.000037e+00
## Problem:
##  Values of the covariates 'twmd', 'twmx' were NA or undefined at 0.19% (6 out of 3139) of the quadra
```

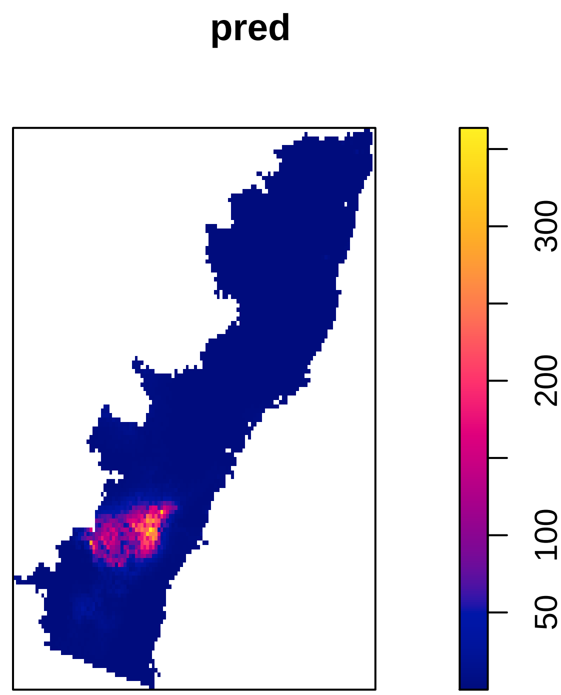
or do an ANOVA.

```
formula_without_bc04 <- paste0("configuration ~ ", paste0(names(covariates)[-2],
    collapse = " + "))
fit_without_bc04 <- spatstat::ppm(as.formula(formula_without_bc04), covariates = covariates)
anova(fit, fit_without_bc04)
```

```
## Analysis of Deviance Table
##
## Model 1: ~bc02 + bc04 + bc05 + bc12 + bc14 + bc21 + bc32 + bc33 + mvbf + rjja + rsea + rugg + subs +
## Model 2: ~bc02 + bc05 + bc12 + bc14 + bc21 + bc32 + bc33 + mvbf + rjja + rsea + rugg + subs + twmd +
##   Npar Df Deviance
## 1   16
## 2   15 -1  -19.831
```

To look at the predicted intensity, you use the `spatstat::predict.ppm` function.

```
pred <- spatstat::predict.ppm(fit, covariates = covariates)
plot(pred)
```



Spatstat can handle many different types of correlation structures between individuals of the species. You would usually supply an `interaction` parameter to 'spatstat::ppm". However, initial analysis suggested attraction between the individuals, in which case a doubly-stochastic (Cox) point process is more appropriate. Fitting such point processes uses another function, as shown below.

```
fit_cox <- spatstat::kppm(as.formula(formula), covariates = covariates, clusters = "LGCP")
summary(fit_cox)
```

```
## Inhomogeneous Cox point process model
## Fitted to point pattern dataset 'configuration'
## Fitted by minimum contrast
##  Summary statistic: inhomogeneous K-function
## Minimum contrast fit (object of class "minconfit")
## Model: Log-Gaussian Cox process
##   Covariance model: exponential
## Fitted by matching theoretical K function to configuration
##
## Internal parameters fitted by minimum contrast ($par):
##     sigma2      alpha
## 6.13869198 0.04268364
##
## Fitted covariance parameters:
##        var      scale
## 6.13869198 0.04268364
```

```
## Fitted mean of log of random intensity:  [pixel image]
##
## Converged successfully after 97 function evaluations
##
## Starting values of parameters:
##     sigma2      alpha
## 1.00000000 0.05169264
## Domain of integration: [ 0 , 1.508 ]
## Exponents: p= 2, q= 0.25
##
## ----------- TREND MODEL -----
## Point process model
## Fitting method: maximum likelihood (Berman-Turner approximation)
## Model was fitted using glm()
## Algorithm converged
## Call:
## ppm.ppp(Q = X, trend = trend, rename.intercept = FALSE, covariates = covariates,
##     covfunargs = covfunargs, use.gam = use.gam, forcefit = TRUE,
##     nd = nd, eps = eps)
## Edge correction: "border"
##  [border correction distance r = 0 ]
## --------------------------------------------------------------------------------
## Quadrature scheme (Berman-Turner) = data + dummy + weights
##
## Data pattern:
## Planar point pattern:  171 points
## Average intensity 9.4 points per square unit
## binary image mask
## 935 x 604 pixel array (ny, nx)
## pixel size: 0.00998 by 0.01 units
## enclosing rectangle: [147.6075, 153.6375] x [-37.505, -28.1575] units
##                      (6.03 x 9.348 units)
## Window area = 18.198 square units
## Fraction of frame area: 0.323
##
## Dummy quadrature points:
##      151 x 55 grid of dummy points, plus 4 corner points
##      dummy spacing: 0.03993377 x 0.16995455 units
##
## Original dummy parameters: =
## Planar point pattern:  2968 points
## Average intensity 163 points per square unit
## binary image mask
## 935 x 604 pixel array (ny, nx)
## pixel size: 0.00998 by 0.01 units
## enclosing rectangle: [147.6075, 153.6375] x [-37.505, -28.1575] units
##                      (6.03 x 9.348 units)
## Window area = 18.198 square units
## Fraction of frame area: 0.323
## Quadrature weights:
##      (counting weights based on 151 x 55 array of rectangular tiles)
## All weights:
##  range: [9.98e-05, 0.00679]  total: 18.2
## Weights on data points:
```

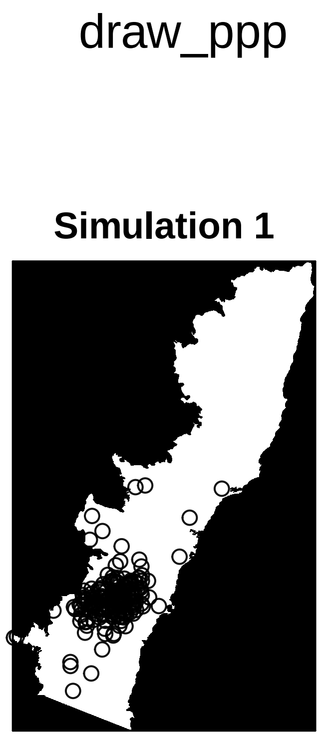
```
##  range: [0.000566, 0.00339]  total: 0.279
## Weights on dummy points:
##  range: [9.98e-05, 0.00679]  total: 17.9
## --------------------------------------------------------------------------------
## FITTED MODEL:
##
## Nonstationary Poisson process
##
## ---- Intensity: ----
##
## Log intensity: ~bc02 + bc04 + bc05 + bc12 + bc14 + bc21 + bc32 + bc33 + mvbf +
## rjja + rsea + rugg + subs + twmd + twmx
## Model depends on external covariates 'bc02', 'bc04', 'bc05', 'bc12', 'bc14',
## 'bc21', 'bc32', 'bc33', 'mvbf', 'rjja', 'rsea', 'rugg', 'subs', 'twmd' and
## 'twmx'
## Covariates provided:
##  bc02: im
##  bc04: im
##  bc05: im
##  bc12: im
##  bc14: im
##  bc21: im
##  bc32: im
##  bc33: im
##  mvbf: im
##  rjja: im
##  rsea: im
##  rugg: im
##  subs: im
##  twmd: im
##  twmx: im
##
## Fitted trend coefficients:
##   (Intercept)          bc02          bc04          bc05          bc12
## -6.973843e+01 -7.532535e-01  9.769080e+00 -7.456425e-01  2.930845e-02
##          bc14          bc21          bc32          bc33          mvbf
##  4.374920e-02  3.496929e+00 -6.106670e+00 -2.648364e+01 -1.296194e-01
##          rjja          rsea          rugg          subs          twmd
## -6.709563e-02 -4.212680e+00 -7.344646e-03 -2.012431e-02 -1.646928e-04
##          twmx
##  3.709840e-05
##
##                  Estimate         S.E.       CI95.lo       CI95.hi Ztest
## (Intercept) -6.973843e+01 1.998915e+01 -1.089164e+02 -3.056042e+01   ***
## bc02        -7.532535e-01 2.662334e-01 -1.275061e+00 -2.314456e-01    **
## bc04         9.769080e+00 2.159955e+00  5.535646e+00  1.400251e+01   ***
## bc05        -7.456425e-01 1.182665e-01 -9.774406e-01 -5.138443e-01   ***
## bc12         2.930845e-02 3.719012e-03  2.201932e-02  3.659758e-02   ***
## bc14         4.374920e-02 1.187564e-01 -1.890091e-01  2.765075e-01
## bc21         3.496929e+00 6.324668e-01  2.257317e+00  4.736541e+00   ***
## bc32        -6.106670e+00 2.628885e+00 -1.125919e+01 -9.541505e-01     *
## bc33        -2.648364e+01 8.499642e+00 -4.314263e+01 -9.824648e+00    **
## mvbf        -1.296194e-01 5.709847e-02 -2.415303e-01 -1.770846e-02     *
## rjja        -6.709563e-02 1.322165e-02 -9.300958e-02 -4.118167e-02   ***
```

```
## rsea        -4.212680e+00 3.299336e+00 -1.067926e+01  2.253901e+00
## rugg        -7.344646e-03 6.012075e-03 -1.912810e-02  4.438804e-03
## subs        -2.012431e-02 2.024376e-02 -5.980134e-02  1.955272e-02
## twmd        -1.646928e-04 7.891568e-05 -3.193647e-04 -1.002095e-05     *
## twmx         3.709840e-05 2.972548e-05 -2.116248e-05  9.535928e-05
##                   Zval
## (Intercept) -3.4888147
## bc02        -2.8292972
## bc04         4.5228164
## bc05        -6.3047626
## bc12         7.8807082
## bc14         0.3683944
## bc21         5.5290312
## bc32        -2.3229128
## bc33        -3.1158536
## mvbf        -2.2701030
## rjja        -5.0746794
## rsea        -1.2768264
## rugg        -1.2216492
## subs        -0.9940996
## twmd        -2.0869470
## twmx         1.2480333
##
## ----------- gory details -----
##
## Fitted regular parameters (theta):
##   (Intercept)          bc02          bc04          bc05          bc12
## -6.973843e+01 -7.532535e-01  9.769080e+00 -7.456425e-01  2.930845e-02
##          bc14          bc21          bc32          bc33          mvbf
##  4.374920e-02  3.496929e+00 -6.106670e+00 -2.648364e+01 -1.296194e-01
##          rjja          rsea          rugg          subs          twmd
## -6.709563e-02 -4.212680e+00 -7.344646e-03 -2.012431e-02 -1.646928e-04
##          twmx
##  3.709840e-05
##
## Fitted exp(theta):
##  (Intercept)         bc02         bc04         bc05         bc12         bc14
## 5.163991e-31 4.708322e-01 1.748467e+04 4.744294e-01 1.029742e+00 1.044720e+00
##         bc21         bc32         bc33         mvbf         rjja         rsea
## 3.301390e+01 2.227958e-03 3.149928e-12 8.784297e-01 9.351058e-01 1.480664e-02
##         rugg         subs         twmd         twmx
## 9.926823e-01 9.800768e-01 9.998353e-01 1.000037e+00
## Problem:
##  Values of the covariates 'twmd', 'twmx' were NA or undefined at 0.19% (6 out of 3139) of the quadra
##
##
## ----------- COX MODEL -----------
## Model: log-Gaussian Cox process
##
##  Covariance model: exponential
## Fitted covariance parameters:
##        var      scale
## 6.13869198 0.04268364
## Fitted mean of log of random intensity: [pixel image]
```

```
##
## Final standard error and CI
## (allowing for correlation of Cox process):
##                  Estimate S.E. CI95.lo CI95.hi Ztest Zval
## (Intercept) -6.973843e+01   NA      NA      NA  <NA>   NA
## bc02        -7.532535e-01   NA      NA      NA  <NA>   NA
## bc04         9.769080e+00   NA      NA      NA  <NA>   NA
## bc05        -7.456425e-01   NA      NA      NA  <NA>   NA
## bc12         2.930845e-02   NA      NA      NA  <NA>   NA
## bc14         4.374920e-02   NA      NA      NA  <NA>   NA
## bc21         3.496929e+00   NA      NA      NA  <NA>   NA
## bc32        -6.106670e+00   NA      NA      NA  <NA>   NA
## bc33        -2.648364e+01   NA      NA      NA  <NA>   NA
## mvbf        -1.296194e-01   NA      NA      NA  <NA>   NA
## rjja        -6.709563e-02   NA      NA      NA  <NA>   NA
## rsea        -4.212680e+00   NA      NA      NA  <NA>   NA
## rugg        -7.344646e-03   NA      NA      NA  <NA>   NA
## subs        -2.012431e-02   NA      NA      NA  <NA>   NA
## twmd        -1.646928e-04   NA      NA      NA  <NA>   NA
## twmx         3.709840e-05   NA      NA      NA  <NA>   NA
```

A nice way to appreciate the difference in the underlying model is to draw from the fitted distribution. This can easily be done for the fitted Poisson point process.

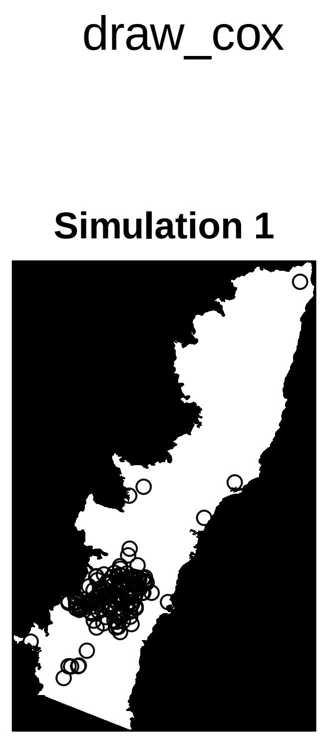
```
draw_ppp <- spatstat::simulate.ppm(fit)
plot(draw_ppp)
```



Drawing from a Cox point process requires you to use another library, but it essentially works in the same way.

```
library(RandomFields)
library(RandomFieldsUtils)

draw_cox <- spatstat::simulate.kppm(fit_cox)
plot(draw_cox)
```



Making goodness-of-fit tests is straightforward, we refer in particular to the functions `spatstat::quadrat.test`, `spatstat::cdf.test`, `spatstat::dclf.test` and `spatstat::mad.test`. A lot of these functions rely on multiple simulations of the point process, which is going to be exeedingly slow for the Cox process. Instead, we show what a goodness-of-fit test looks like with a simple fit with a Poisson point process.

```
dclf.test(fit)
```

```
## Generating 99 simulated realisations of fitted Poisson model  ...
## 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, 2
## 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, 6
## 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98,  99.
##
## Done.
```

```
##
##  Diggle-Cressie-Loosmore-Ford test of fitted Poisson model
##  Monte Carlo test based on 99 simulations
##  Summary function: K(r)
##  Reference function: sample mean
##  Alternative: two.sided
##  Interval of distance values: [0, 1.5075]
##  Test statistic: Integral of squared absolute deviation
##  Deviation = leave-one-out
##
```

```
## data:  fit
## u = 2.0884, rank = 7, p-value = 0.07
```

Comparing the results