# Kimboto dataset

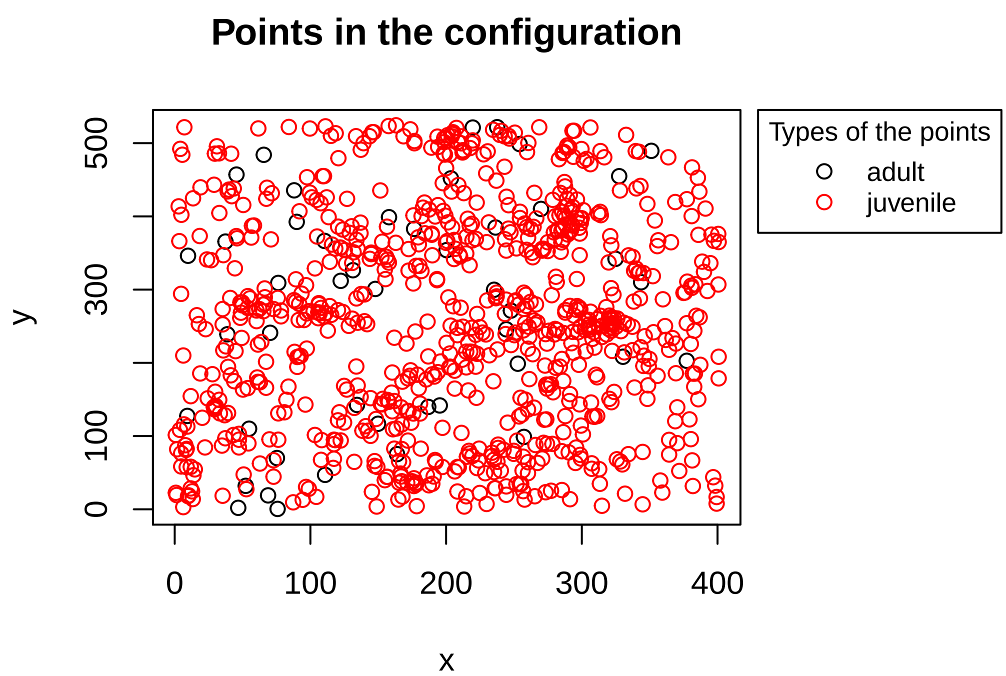
```r
library(ppjsdm)
#> Registered S3 method overwritten by 'spatstat':
#>   method     from
#>   print.boxx cli
library(spatstat)
#> Loading required package: spatstat.data
#> Loading required package: nlme
#> Loading required package: rpart
#>
#> spatstat 1.63-0       (nickname: 'Space camouflage')
#> For an introduction to spatstat, type 'beginner'
remove(list = ls())

set.seed(1)
```

This vignette explains how to use the `ppjsdm` package with the kimboto dataset from `spatstat`. We begin by loading the data with all species.

```r
print(paracou)
#> Marked planar point pattern: 884 points
#> Multitype, with levels = adult, juvenile
#> window: rectangle = [0, 400.8568] x [0, 524.4037] metres
configuration <- Configuration(paracou)
#> Warning in Configuration(paracou): There are duplicate points in the
#> configuration.
window <- Rectangle_window(c(0, 400.8568), c(0, 524.4037))
```

The point configuration is plotted below.

```r
par(mar = c(5, 4, 4, 13) + 0.1)
plot(configuration, window = window)
```

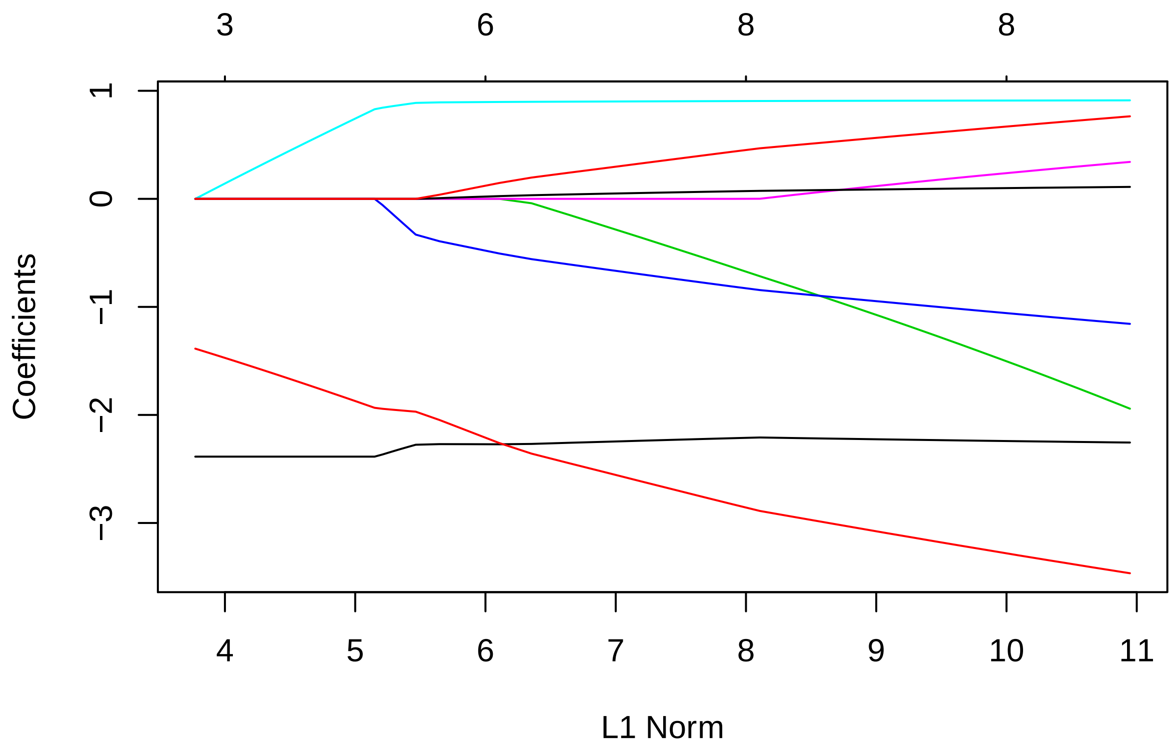



We provide a series of ranges for the interaction radii, and let the fitting function calibrate the model.

```
short_range <- c(0, 20)
medium_range <- c(0, 20)
long_range <- c(0, 20)
model <- "square_exponential"
medium_range_model <- "square_exponential"
```

We can now call the fitting function.

```
fit <- ppjsdm::gibbsm(configuration,
                      window = window,
                      model = model,
                      medium_range_model = medium_range_model,
                      short_range = short_range,
                      medium_range = medium_range,
                      long_range = long_range,
                      use_glmnet = TRUE)
#> (Intercept) log_lambda1 log_lambda2   alpha_1_1   alpha_1_2   alpha_2_2
#>   0.0000000  -8.2967095  -7.6036883  -1.9422285  -1.1569850   0.9116492
#>   gamma_1_1   gamma_1_2   gamma_2_2
#>   0.3419673   0.1106310   0.7637698
plot(fit$complete)
```

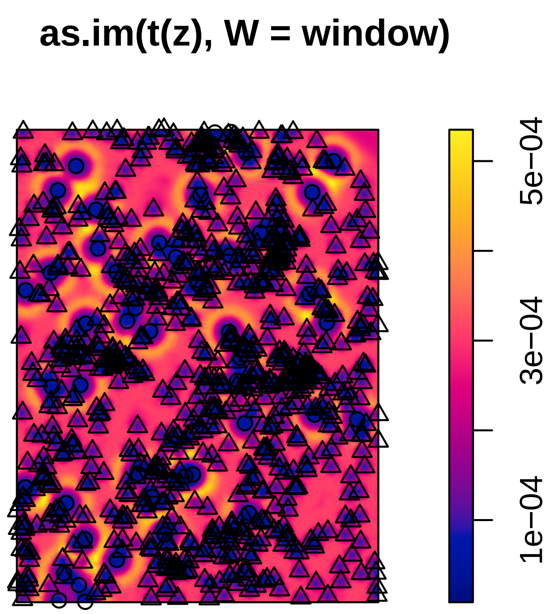



```r
print(fit$coefficients)
#> (Intercept) log_lambda1 log_lambda2   alpha_1_1   alpha_1_2   alpha_2_2
#>   0.0000000  -8.2967095  -7.6036883  -1.9422285  -1.1569850   0.9116492
#>   gamma_1_1   gamma_1_2   gamma_2_2
#>   0.3419673   0.1106310   0.7637698
print(fit$best_short)
#>           [,1]     [,2]
#> [1,] 10.760761 5.440503
#> [2,]  5.440503 6.316831
print(fit$best_medium)
#>          [,1]     [,2]
#> [1,] 18.86555 19.70384
#> [2,] 19.70384 11.22424
print(fit$best_long)
#>          [,1]     [,2]
#> [1,] 26.14602 28.37403
#> [2,] 28.37403 21.09029
print(fit$aic)
#> [1] -2284.478
print(fit$bic)
#> [1] -2232.804
```

We may then plot the corresponding Papangelou conditional intensity.

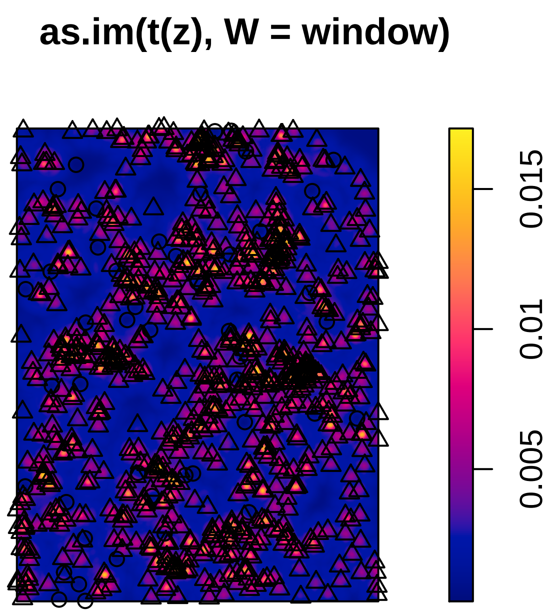
```r
parameters <- get_parameters_from_fit(fit)
lambda <- parameters$lambda
alpha <- parameters$alpha
```

```r
gamma <- parameters$gamma
plot_papangelou(window = window,
                configuration = configuration,
                type = 1,
                model = model,
                medium_range_model = medium_range_model,
                alpha = alpha,
                lambda = lambda,
                beta = matrix(0, 2, 0),
                gamma = gamma,
                covariates = list(),
                short_range = fit$best_short,
                medium_range = fit$best_medium,
                long_range = fit$best_long,
                saturation = 2)
#> Warning: data contain duplicated points
```




```r
plot_papangelou(window = window,
                configuration = configuration,
                type = 2,
                model = model,
                medium_range_model = medium_range_model,
                alpha = alpha,
                lambda = lambda,
                beta = matrix(0, 2, 0),
```

```
                 gamma = gamma,
                 covariates = list(),
                 short_range = fit$best_short,
                 medium_range = fit$best_medium,
                 long_range = fit$best_long,
                 saturation = 2)
#> Warning: data contain duplicated points
```




It is also possible to draw from the model.

```
parameters <- get_parameters_from_fit(fit)
lambda <- parameters$lambda
alpha <- parameters$alpha
gamma <- parameters$gamma
draw <- ppjsdm::rgibbs(window = window,
                       alpha = alpha,
                       lambda = lambda,
                       gamma = gamma,
                       model = model,
                       medium_range_model = medium_range_model,
                       short_range = fit$best_short,
                       medium_range = fit$best_medium,
                       long_range = fit$best_long,
                       types = levels(types(configuration)),
                       mark_range = c(min(get_marks(configuration)), max(get_marks(configuration))),
                       steps = 10000000)
print(draw)
```

```
#> An S3 object representing a configuration.
#>
#> Number of points: 858.

par(mar = c(5, 4, 4, 13) + 0.1)
plot(draw, window = window)
```