Fin pines dataset

```
library(ppjsdm)
library(spatstat)
#> Loading required package: spatstat.data
#> Loading required package: spatstat.geom
#> spatstat.geom 2.4-0
#>
#> Attaching package: 'spatstat.geom'
#> The following object is masked from 'package:ppjsdm':
#>
#>     marks
#> Loading required package: spatstat.core
#> Loading required package: nlme
#> Loading required package: rpart
#> spatstat.core 2.3-2
#> Loading required package: spatstat.linnet
#> spatstat.linnet 2.3-1
#>
#> spatstat 2.3-0       (nickname: 'That's not important right now')
#> For an introduction to spatstat, type 'beginner'
remove(list = ls())

set.seed(1)
```

This vignette explains how to use the `ppjsdm` package with the `finpines` dataset from `spatstat`. The data record the locations of 126 pine saplings in a Finnish forest, their heights and their diameters.

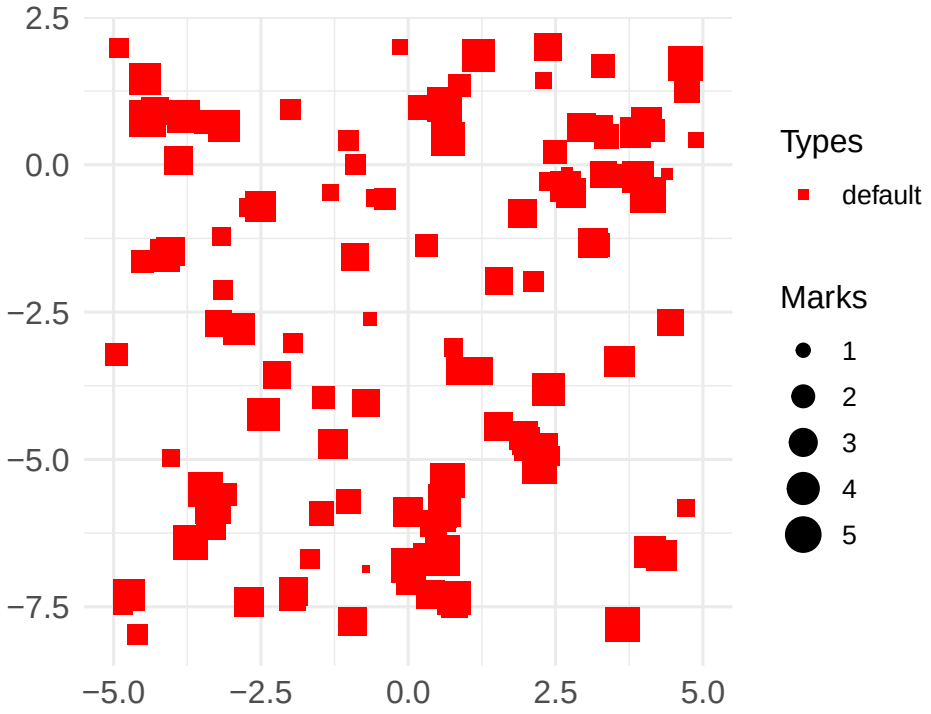
Taking marks into account

If marks are provided, the interaction radii are proportional to the marks. We begin with that setting, choosing the height (rather than the diameter) as the relevant mark.

```
configuration <- Configuration(finpines$x, finpines$y, marks = finpines$marks$height)
window <- Rectangle_window(c(-5, 5), c(-8, 2))
```

The point configuration is plotted below.

```
print(configuration)
#> An S3 object representing a configuration.
#>
#> Number of points: 126.
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"
max_points <- 150
saturation <- 2
steps <- 100000
```

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,
                      saturation = saturation,
                      fitting_package = "glm")
print(summary(fit))
#>           coefficients         se      CI95_lo    CI95_hi Ztest         Pval
#> beta0_1    -32.9445335 16.6397556 -65.55785527 -0.3312117     * 4.771827e-02
#> alpha_1_1    0.8491603  0.1571538   0.54114444  1.1571762   *** 6.540046e-08
#> gamma_1_1   16.3788077  8.3214085   0.06914682 32.6884686     * 4.903658e-02
#>                Zval se_numerical_proportion
#> beta0_1   -1.979869                0.3534096
#> alpha_1_1  5.403370                0.5426954
```

```
#> gamma_1_1  1.968273               0.3535238
print(fit$coefficients)
#> $beta0
#>   default
#> -32.94453
#>
#> $alpha
#>           default
#> default 0.8491603
#>
#> $gamma
#>          default
#> default 16.37881
#>
#> $beta
#>
#> default
#>
#> $short_range
#>            default
#> default 0.06220457
#>
#> $medium_range
#>           default
#> default 0.9203159
#>
#> $long_range
#>          default
#> default 5.416841
print(fit$aic)
#> [1] 580.2048
print(fit$bic)
#> [1] 593.5419
```

We may then plot the corresponding Papangelou conditional intensity.

```
parameters <- fit$coefficients
# plot_papangelou(window = window,
#                 configuration = configuration,
#                 type = 1,
#                 mark = mean(get_marks(configuration)),
#                 model = model,
#                 medium_range_model = medium_range_model,
#                 alpha = parameters$alpha,
#                 beta0 = parameters$beta0,
#                 beta = matrix(0, 1, 0),
#                 gamma = parameters$gamma,
#                 covariates = list(),
#                 short_range = parameters$short_range,
#                 medium_range = parameters$medium_range,
#                 long_range = parameters$long_range,
#                 saturation = saturation,
#                 max_points = max_points)
```

It is also possible to draw from the model.

```
draw <- ppjsdm::rgibbs(window = window,
                       alpha = parameters$alpha,
                       beta0 = parameters$beta0,
                       gamma = parameters$gamma,
                       model = model,
                       medium_range_model = medium_range_model,
                       short_range = parameters$short_range,
                       medium_range = parameters$medium_range,
                       long_range = parameters$long_range,
                       types = levels(types(configuration)),
                       mark_range = c(min(ppjsdm::marks(configuration)), max(ppjsdm::marks(configuratio
                       saturation = saturation,
                       steps = steps)
print(draw)
#> An S3 object representing a configuration.
#>
#> Number of points: 0.

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

Interaction radii not proportional to marks

In this section, we disregard the height of the saplings, and consider interaction radii in metres.

```
configuration <- Configuration(finpines$x, finpines$y)
```

We call the fitting function on this unmarked point process.

```
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,
                      saturation = saturation,
                      fitting_package = "glm")
print(summary(fit))
#>           coefficients       se   CI95_lo  CI95_hi Ztest         Pval
#> beta0_1     -1.7690507 1.853418 -5.401683 1.863582       3.398406e-01
#> alpha_1_1    0.7819803 0.148456  0.491012 1.072949   *** 1.383526e-07
#> gamma_1_1    0.7886131 1.012294 -1.195446 2.772672       4.359586e-01
#>                 Zval se_numerical_proportion
#> beta0_1   -0.9544802                0.2450993
#> alpha_1_1  5.2674224                0.4939217
#> gamma_1_1  0.7790359                0.2452108
print(fit$coefficients)
#> $beta0
#>   default
#> -1.769051
#>
```

```
#> $alpha
#>           default
#> default 0.7819803
#>
#> $gamma
#>           default
#> default 0.7886131
#>
#> $beta
#>
#> default
#>
#> $short_range
#>           default
#> default 0.2519842
#>
#> $medium_range
#>           default
#> default 4.430754
#>
#> $long_range
#>           default
#> default 22.34514
print(fit$aic)
#> [1] 589.7884
print(fit$bic)
#> [1] 603.1255
```

We may then plot the corresponding Papangelou conditional intensity.

```
parameters <- fit$coefficients
# plot_papangelou(window = window,
#                 configuration = configuration,
#                 type = 1,
#                 mark = mean(get_marks(configuration)),
#                 model = model,
#                 medium_range_model = medium_range_model,
#                 alpha = parameters$alpha,
#                 lambda = beta0$beta0,
#                 beta = matrix(0, 1, 0),
#                 gamma = parameters$gamma,
#                 covariates = list(),
#                 short_range = parameters$short_range,
#                 medium_range = parameters$medium_range,
#                 long_range = parameters$long_range,
#                 saturation = saturation,
#                 max_points = max_points)
```

And as previously, we draw from the model.

```
draw <- ppjsdm::rgibbs(window = window,
                       alpha = parameters$alpha,
                       beta0 = parameters$beta0,
                       gamma = parameters$gamma,
                       model = model,
```

```
                        medium_range_model = medium_range_model,
                        short_range = parameters$short_range,
                        medium_range = parameters$medium_range,
                        long_range = parameters$long_range,
                        types = levels(types(configuration)),
                        saturation = saturation,
                        mark_range = c(min(ppjsdm::marks(configuration)), max(ppjsdm::marks(configuratio
                        steps = steps)
print(draw)
#> An S3 object representing a configuration.
#>
#> Number of points: 117.

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