Norwegian spruces dataset

```
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 1.64-0       (nickname: 'Susana Distancia')
#> For an introduction to spatstat, type 'beginner'
remove(list = ls())

set.seed(1)
```

This vignette explains how to use the `ppjsdm` package with the `spruces` dataset from `spatstat`. The data give the locations of Norwegian spruce trees in a natural forest stand in Saxonia, Germany. Each tree is marked with its diameter at breast height.

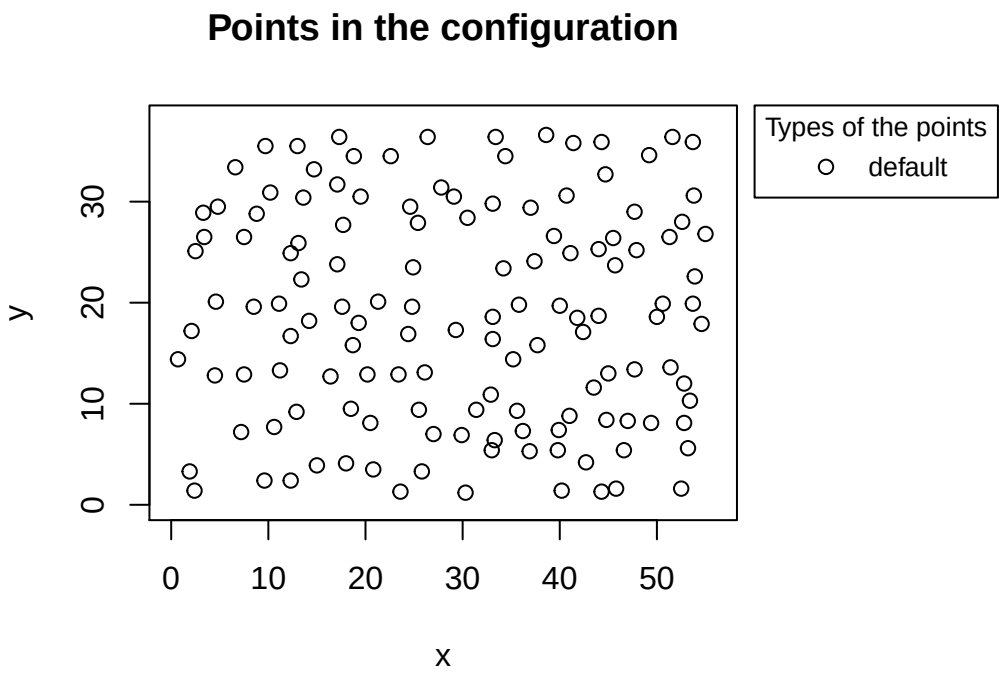
Taking marks into account

If marks are provided, the interaction radii are proportional to the marks, i.e. their diameter at breat height. We begin with that setting.

```
configuration <- Configuration(spruces$x, spruces$y, marks = spruces$marks)
window <- Rectangle_window(c(0, 56), c(0, 38))
spatstat_window <- owin(c(0, 56), c(0, 38))
```

The point configuration is plotted below.

```
print(configuration)
#> An S3 object representing a configuration.
#>
#> Number of points: 134.
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"
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,
                      use_glmnet = FALSE,
                      use_aic = TRUE,
                      saturation = saturation)
#> $beta0
#> [1] -1.536804
#>
#> $alpha
#>           [,1]
#> [1,] -1.957606
#>
#> $gamma
```

```
#>           [,1]
#> [1,] 0.1668877
#>
#> $beta
#>
#> [1,]
print(summary(fit))
#>             coefficients        se    CI95_lo    CI95_hi Ztest         Pval
#> log_lambda1   -1.5368038 0.7143363 -2.9368773 -0.1367303     * 3.144678e-02
#> alpha_1_1     -1.9576064 0.3250987 -2.5947882 -1.3204246   *** 1.727281e-09
#> gamma_1_1      0.1668877 0.3279599 -0.4759018  0.8096772       6.108459e-01
#>                  Zval
#> log_lambda1 -2.1513729
#> alpha_1_1   -6.0215749
#> gamma_1_1    0.5088663
print(fit$coefficients)
#> $beta0
#> [1] -1.536804
#>
#> $alpha
#>           [,1]
#> [1,] -1.957606
#>
#> $gamma
#>           [,1]
#> [1,] 0.1668877
#>
#> $beta
#>
#> [1,]
#>
#> $short_range
#>          [,1]
#> [1,] 6.254278
#>
#> $medium_range
#>          [,1]
#> [1,] 15.72074
#>
#> $long_range
#>          [,1]
#> [1,] 27.77754
print(fit$aic)
#> [1] 546.9585
print(fit$bic)
#> [1] 560.4804
```

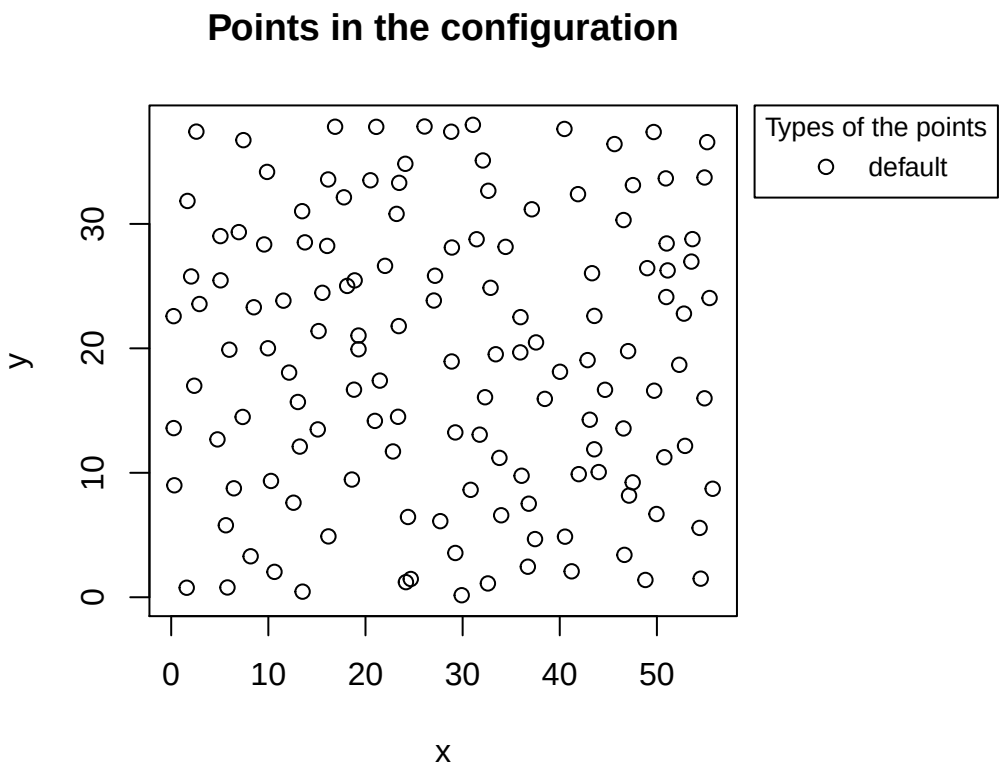
We may then plot the corresponding Papangelou conditional intensity.

```
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.

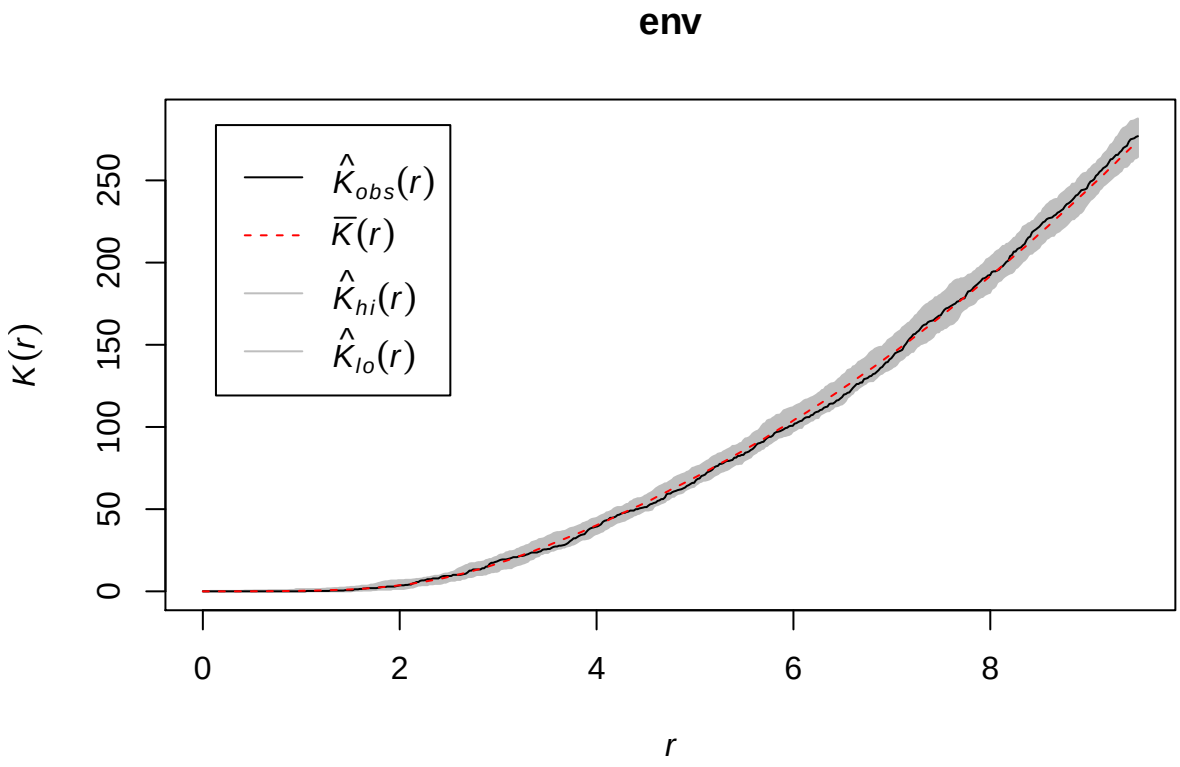
```
draw <- ppjsdm::rgibbs(window = window,
                       alpha = parameters$alpha,
                       beta0 = parameters$beta0,
                       gamma = parameters$gamma,
                       model = model,
                       medium_range_model = medium_range_model,
                       short_range = parameters$short_range,
                       medium_range = parameters$medium_range,
                       long_range = parameters$long_range,
                       types = levels(types(configuration)),
                       mark_range = c(min(get_marks(configuration)), max(get_marks(configuration))),
                       saturation = saturation,
                       steps = steps)
print(draw)
#> An S3 object representing a configuration.
#>
#> Number of points: 137.

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



A more standard approach to check goodness of fit is a comparison of K functions.

```
env <- envelope.ppp(ppp(configuration$x, configuration$y, window = spatstat_window), simulate = functio
  z <- 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(get_marks(configuration)), max(get_marks(configuration))),
                      saturation = saturation,
                      steps = steps)
  ppp(z$x, z$y, window = spatstat_window)
  })
#> Generating 99 simulations by evaluating function  ...
#> 1, 2,  [etd 1:52:38] 3,  [etd 1:51:11] 4,
#>  [etd 1:49:55] 5,  [etd 1:49:15] 6,  [etd 1:48:57] 7,  [etd 1:52:08] 8,
#>  [etd 1:59:30] 9,  [etd 2:04:19] 10,  [etd 2:06:59] 11,  [etd 2:04:14] 12,
#>  [etd 2:01:46] 13,  [etd 1:59:30] 14,  [etd 1:57:21] 15,  [etd 1:55:19] 16,
#>  [etd 1:53:19] 17,  [etd 1:51:26] 18,  [etd 1:49:46] 19,  [etd 1:48:02] 20,
#>  [etd 1:46:27] 21,  [etd 1:44:52] 22,  [etd 1:43:21] 23,  [etd 1:41:49] 24,
#>  [etd 1:40:15] 25,  [etd 1:38:46] 26,  [etd 1:37:17] 27,  [etd 1:35:52] 28,
#>  [etd 1:34:22] 29,  [etd 1:32:56] 30,  [etd 1:31:31] 31,  [etd 1:30:05] 32,
```

```
#>  [etd 1:28:39] 33,  [etd 1:27:15] 34,  [etd 1:25:50] 35,  [etd 1:24:27] 36,
#>  [etd 1:23:06] 37,  [etd 1:21:41] 38,  [etd 1:20:19] 39,  [etd 1:18:57] 40,
#>  [etd 1:17:36] 41,  [etd 1:16:14] 42,  [etd 1:14:53] 43,  [etd 1:13:31] 44,
#>  [etd 1:12:09] 45,  [etd 1:10:48] 46,  [etd 1:09:28] 47,  [etd 1:08:08] 48,
#>  [etd 1:06:48] 49,  [etd 1:05:27] 50,  [etd 1:04:08] 51,  [etd 1:02:49] 52,
#>  [etd 1:01:29] 53,  [etd 1:00:09] 54,  [etd 58:50] 55,  [etd 57:29] 56,
#>  [etd 56:10] 57,  [etd 54:50] 58,  [etd 53:47] 59,  [etd 52:46] 60,
#>  [etd 51:37] 61,  [etd 50:13] 62,  [etd 49:07] 63,  [etd 48:00] 64,
#>  [etd 46:49] 65,  [etd 45:25] 66,  [etd 44:02] 67,  [etd 42:41] 68,
#>  [etd 41:19] 69,  [etd 39:58] 70,  [etd 38:38] 71,  [etd 37:17] 72,
#>  [etd 35:58] 73,  [etd 34:45] 74,  [etd 33:29] 75,  [etd 32:15] 76,
#>  [etd 31:02] 77,  [etd 29:57] 78,  [etd 28:49] 79,  [etd 27:42] 80,
#>  [etd 26:42] 81,  [etd 25:37] 82,  [etd 24:29] 83,  [etd 23:20] 84,
#>  [etd 22:09] 85,  [etd 20:55] 86,  [etd 19:39] 87,  [etd 18:19] 88,
#>  [etd 16:59] 89,  [etd 15:34] 90,  [etd 14:09] 91,  [etd 12:41] 92,
#>  [etd 11:12] 93,  [etd 9:40] 94,  [etd 8:07] 95,  [etd 6:31] 96,
#>  [etd 4:54] 97,  [etd 3:18] 98,  [etd 1:39]  99.
#>
#> Done.
plot(env)
```



Interaction radii not proportional to marks

In this section, we disregard the diameter at breast height, and consider interaction radii in metres.

```
configuration <- Configuration(spruces$x, spruces$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,
                        use_glmnet = FALSE,
                      use_aic = TRUE,
                      saturation = saturation)
#> $beta0
#> [1] -12.12663
#>
#> $alpha
#>           [,1]
#> [1,] -1.827938
#>
#> $gamma
#>          [,1]
#> [1,] 5.585367
#>
#> $beta
#>
#> [1,]
print(summary(fit))
#>              coefficients         se    CI95_lo   CI95_hi Ztest         Pval
#> log_lambda1   -12.126628 13.8715569 -39.314380 15.061125       3.820049e-01
#> alpha_1_1      -1.827938  0.2964326  -2.408935 -1.246941   *** 6.983841e-10
#> gamma_1_1       5.585367  6.9314534  -8.000032 19.170766       4.203580e-01
#>                    Zval
#> log_lambda1 -0.8742081
#> alpha_1_1   -6.1664539
#> gamma_1_1    0.8058003
print(fit$coefficients)
#> $beta0
#> [1] -12.12663
#>
#> $alpha
#>           [,1]
#> [1,] -1.827938
#>
#> $gamma
#>          [,1]
#> [1,] 5.585367
#>
#> $beta
#>
#> [1,]
#>
#> $short_range
```

```
#>           [,1]
#> [1,] 1.772047
#>
#> $medium_range
#>           [,1]
#> [1,] 10.05164
#>
#> $long_range
#>           [,1]
#> [1,] 17.51518
print(fit$aic)
#> [1] 548.342
print(fit$bic)
#> [1] 561.8638
```

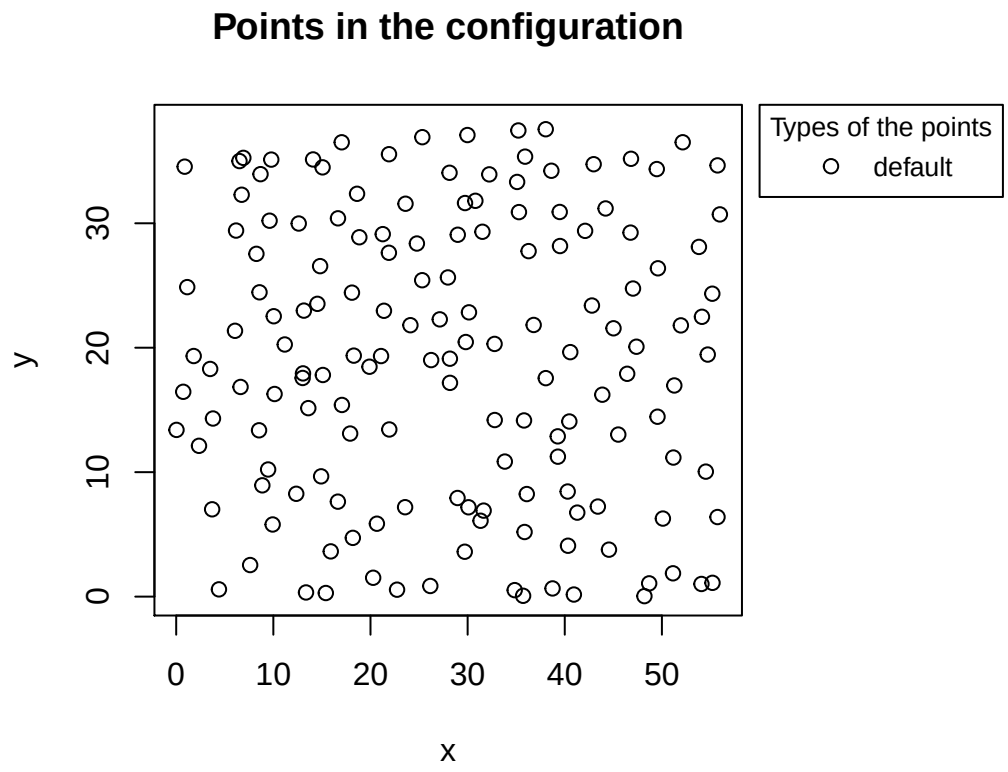
We may then plot the corresponding Papangelou conditional intensity.

```
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)
```

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)),
                       mark_range = c(min(get_marks(configuration)), max(get_marks(configuration))),
                       saturation = saturation,
                       steps = steps)
print(draw)
#> An S3 object representing a configuration.
#>
#> Number of points: 153.

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

```
plot(draw, window = window)
```



And let us also plot the K function.

```
env <- envelope.ppp(ppp(configuration$x, configuration$y, window = spatstat_window), simulate = functio
  z <- 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(get_marks(configuration)), max(get_marks(configuration))),
                      saturation = saturation,
                      steps = steps)
  ppp(z$x, z$y, window = spatstat_window)
  })
#> Generating 99 simulations by evaluating function  ...
#> 1, 2,  [etd 4:04:52] 3,  [etd 3:27:41] 4,
#>  [etd 3:38:01] 5,  [etd 3:38:21] 6,  [etd 3:33:34] 7,  [etd 3:33:51] 8,
#>  [etd 3:34:31] 9,  [etd 3:37:37] 10,  [etd 3:38:16] 11,  [etd 3:38:29] 12,
#>  [etd 3:37:04] 13,  [etd 3:35:42] 14,  [etd 3:34:14] 15,  [etd 3:32:21] 16,
#>  [etd 3:30:50] 17,  [etd 3:27:55] 18,  [etd 3:22:01] 19,  [etd 3:16:27] 20,
```

```
#>  [etd 3:11:08] 21,  [etd 3:05:58] 22,  [etd 3:04:57] 23,  [etd 3:02:48] 24,
#>  [etd 2:59:20] 25,  [etd 2:54:57] 26,  [etd 2:50:53] 27,  [etd 2:46:56] 28,
#>  [etd 2:43:15] 29,  [etd 2:39:43] 30,  [etd 2:36:14] 31,  [etd 2:32:44] 32,
#>  [etd 2:29:19] 33,  [etd 2:26:13] 34,  [etd 2:23:28] 35,  [etd 2:20:37] 36,
#>  [etd 2:18:00] 37,  [etd 2:15:18] 38,  [etd 2:12:49] 39,  [etd 2:10:06] 40,
#>  [etd 2:07:27] 41,  [etd 2:04:46] 42,  [etd 2:02:07] 43,  [etd 1:59:26] 44,
#>  [etd 1:56:47] 45,  [etd 1:54:06] 46,  [etd 1:51:31] 47,  [etd 1:48:53] 48,
#>  [etd 1:47:08] 49,  [etd 1:45:32] 50,  [etd 1:43:58] 51,  [etd 1:42:18] 52,
#>  [etd 1:39:53] 53,  [etd 1:37:26] 54,  [etd 1:34:59] 55,  [etd 1:32:33] 56,
#>  [etd 1:30:09] 57,  [etd 1:27:43] 58,  [etd 1:25:22] 59,  [etd 1:23:04] 60,
#>  [etd 1:20:47] 61,  [etd 1:19:00] 62,  [etd 1:16:53] 63,  [etd 1:14:35] 64,
#>  [etd 1:12:14] 65,  [etd 1:09:54] 66,  [etd 1:07:34] 67,  [etd 1:05:15] 68,
#>  [etd 1:02:58] 69,  [etd 1:00:38] 70,  [etd 58:18] 71,  [etd 56:01] 72,
#>  [etd 53:46] 73,  [etd 51:32] 74,  [etd 49:20] 75,  [etd 47:10] 76,
#>  [etd 45:00] 77,  [etd 42:52] 78,  [etd 40:46] 79,  [etd 38:41] 80,
#>  [etd 36:37] 81,  [etd 34:33] 82,  [etd 32:31] 83,  [etd 30:31] 84,
#>  [etd 28:30] 85,  [etd 26:31] 86,  [etd 24:34] 87,  [etd 22:37] 88,
#>  [etd 20:40] 89,  [etd 18:44] 90,  [etd 16:49] 91,  [etd 14:54] 92,
#>  [etd 13:00] 93,  [etd 11:07] 94,  [etd 9:14] 95,  [etd 7:22] 96,
#>  [etd 5:30] 97,  [etd 3:40] 98,  [etd 1:50]  99.
#>
#> Done.
plot(env)
```