Simulation experiments for paper on Hybrid point processes

Adrian Baddeley and Jorge Mateu

Revision: 1.5 Date: 2013/03/04 02:54:18

This `Sweave` script is part of the online supplementary material for the paper

Hybrids of Gibbs point processes and their implementation
Baddeley, Turner, Mateu and Bevan

This file runs the simulation experiment described in Section 10 of the paper.

1 Setup

Set the following switch to `TRUE` to recompute everything in this script, or `FALSE` to re-use the previously-saved results of simulations.

```
> recompute <- FALSE
```

Files

Create directory for plot files

```
> if(!file.exists("pix-sim")) dir.create("pix-sim")
```

Packages

Load the required packages and data.

```
> library(spatstat)
> if(versionstring.spatstat() < "1.34-1") {
+   stop("spatstat version 1.34-1 or later is required")
+ }
```

```
> sessionLibs()
```

```
Libraries loaded:
         spatstat 1.34-1
         polyclip 1.1-0
         tensor 1.5
         abind 1.4-0
         deldir 0.0-22
         mgcv 1.7-26
         nlme 3.1-111
```

Parameters

Fix the simulation seed (using Adrian's office phone number), so that the plots are repeatable!

```
> set.seed(893336177)
```

Fix the resolution for interaction radius (for grid search in profile maximum pseudolikelihood).

```
> rstep <- 0.001
```

2 Simulation experiment

The purpose of this simulation experiment is to assess the performance of maximum profile pseudolikelihood estimation of a hybrid model.

We have elected to use *exact* simulation to generate the simulated realisations, so that there are no issues regarding the convergence of the MCMC simulation algorithm (as there might be if we had used Metropolis-Hastings).

The model is the hybrid Strauss-hard core process, which is one of the few hybrid models for which we already have code for exact simulation.

The model has first order parameter $\beta = 300$, Strauss interaction parameter $\gamma = 0.5$, Strauss interaction range $r = 0.07$, and hard core radius $c = 0.04$ in the unit square. Simulated patterns contain about 100 points.

```
> beta.true <- 300
> gamma.true <- 0.5
> r.true <- 0.07
> c.true <- 0.04
> param <- list(beta=beta.true, gamma=gamma.true, R=r.true, H=c.true)
> truth <- c("(Intercept)"=log(beta.true), "Interaction"=log(gamma.true),
+            "r"=r.true, "hc"=c.true)
```

Now run the experiment.

Note that rStraussHard generates simulations inside the unit square (by default) without any kind of adjustment for edge effects. Correspondingly we don't do edge correction when fitting the model (so that the model being fitted is exactly correct). Therefore we specify `rbord=0` when fitting the model.

The hard core radius c will be estimated from each realisation x by

$$\hat{c} = \frac{n(x)}{n(x)+1} m(x)$$

where $m(x)$ is the minimum nearest neighbour distance and $n(x)$ is the number of points. We write a function that does this calculation:

```
> hard.est <- function(x) {
+   n <- npoints(x)
+   m <- min(nndist(x))
+   return(m * n/(n+1))
+ }
```

The command `profilepl` performs profile maximum pseudolikelihood. It requires a data frame specifying the grid of parameter values to be searched. This data frame will be constructed separately for each simulated pattern, because the allowable values for r are those larger than $c = \hat{c}(x)$.

This experiment works well with the default settings for quadrature schemes. However the default settings are designed so that the package checker runs quickly, rather than to achieve optimal performance. To demonstrate really good performance we increase the density of dummy points to a 128×128 grid by setting `nd=128`.

```
> if(recompute || !file.exists("simSH.rda")) {
+   rseq <- seq(0.04,0.1,by=rstep)
+   results <- NULL
+   for(i in 1:100) {
+     # generate simulated pattern
+     Xsim <- rStraussHard(beta=beta.true, gamma=gamma.true, R=r.true, H=c.true)
+     # estimate hard core distance
+     c.hat <- hard.est(Xsim)
+     # construct search grid
+     df <- data.frame(r=rseq[rseq > c.hat], hc=c.hat)
+     # apply profile maximum pseudolikelihood
+     Xsimfit <- profilepl(df, StraussHard, Xsim, ~1,
+                          rbord=0, nd=128, verbose=FALSE)$fit
+     # store results
+     results.i <- c(coef(Xsimfit), unlist(as.interact(Xsimfit)$par))
+     results <- rbind(results, results.i)
+   }
+   rownames(results) <- paste(1:100)
+   attr(results, "param") <- param
+   attr(results, "truth") <- truth
+   save(results, file="simSH.rda", compress=TRUE)
+ } else load("simSH.rda")
```

Calculate the bias:

```
> colMeans(results)

(Intercept) Interaction           r          hc
 5.64435734 -0.70384964  0.07184000  0.04062555

> colMeans(results) - truth
```

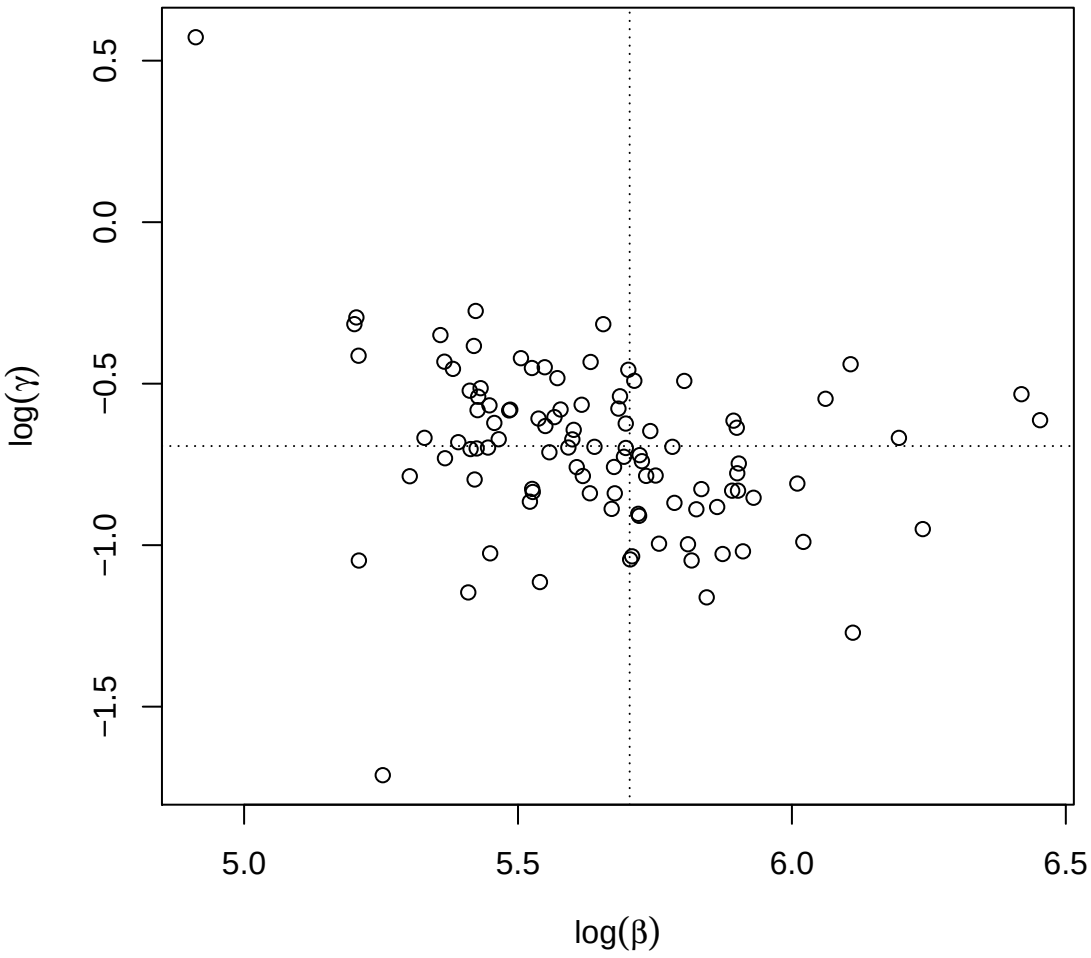
```
  (Intercept)   Interaction             r            hc
-0.0594251313 -0.0107024625  0.0018400000  0.0006255468
```

Calculate the variances and covariances:

```
> var(results)

              (Intercept)   Interaction            r            hc
(Intercept)  6.754605e-02 -2.150112e-02 1.312057e-03  1.677365e-05
Interaction -2.150112e-02  7.146800e-02 5.503722e-04 -3.722076e-06
r            1.312057e-03  5.503722e-04 7.508525e-05  3.006531e-07
hc           1.677365e-05 -3.722076e-06 3.006531e-07  1.059387e-06

> sqrt(diag(var(results)))

(Intercept) Interaction           r          hc
0.259896235 0.267334996 0.008665175 0.001029265
```

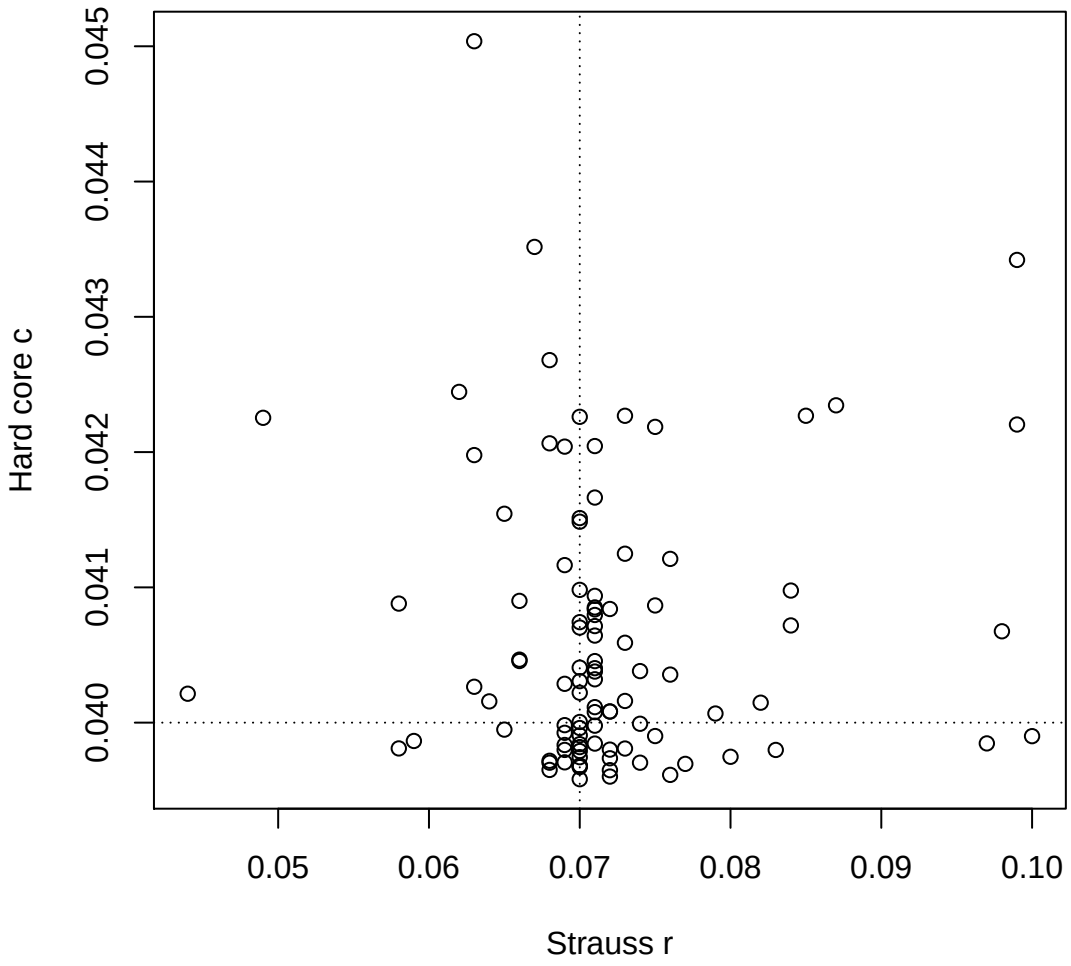
Plot the estimates of the canonical parameters

```
> plot(range(results[,1], log(beta.true)),
+      range(results[,2], log(gamma.true)),
+      type = "n",
+      xlab=expression(log(beta)), ylab=expression(log(gamma)))
> points(results[,1], results[,2])
> abline(v=log(beta.true), lty=3)
> abline(h=log(gamma.true), lty=3)
```



Plot the estimates of the irregular parameters

```
> plot(range(results[,3], r.true),
+      range(results[,4], c.true, c.true+0.005),
+      type = "n",
+      xlab="Strauss r", ylab="Hard core c")
> points(results[,3], results[,4])
> abline(v=r.true, lty=3)
> abline(h=c.true, lty=3)
```


Hard core c
0.040
0.041
0.042
0.043
0.044
0.045
0.05
0.06
0.07
0.08
0.09
0.10
Strauss r