

Supplementary material 4: Simulation Study

T. Leon, J. Domingo, A.L. Rizzo-Campos and G. Ayala

2026-05-22

Table of contents

1 Packages	1
2 Setup	1
3 Simulating point patterns	2
4 Linked pairs	2
5 Neyman-Scott process	3
6 Multitype hard core point process	4

1 Packages

```
pacman::p_load(spatstat,combinat,parallel)
source("functions_srt.R")
```

2 Setup

```
set.seed(123)
```

3 Simulating point patterns

4 Linked pairs

We generate pairs of dependent point processes. For each pair, the first component is a Poisson point process with intensity `lambda` meanwhile the second component is a random shift of the first one where the random translations are chosen within a disc. It is implemented in the file `funtions_srt.R` in the function `rModel1`.

```
winbase = owin(c(0,1),c(0,1))
num.genes.values = seq(50,100,10)
num.b.values = seq(20,40,5)
lambda.values = seq(10,20,2)
radius.values = seq(0.005,.05,length.out=5)
nsim = 100
errors = matrix(NA,ncol=7,nrow=length(num.genes.values)*
               length(num.b.values)*
               length(lambda.values)*
               length(radius.values)*nsim)

rvalues = seq(0,.15,length.out=20)
row = 0
for(num.genes in num.genes.values){
  cat("num.genes ",num.genes,"\n")
  pairs_to_eval = t(combn(num.genes,2))
  for(num.b in num.b.values){
    cat("num.b ",num.b,"\n")
    pairs_sig = cbind( 2*(1:(num.b/2))-1, 2*(1:(num.b/2)))
    ptr = pairs_to_rows(pairs_sig,pairs_to_eval)
    for(lambda in lambda.values){
      cat("lambda ",lambda,"\n")
      for(radius in radius.values){
        cat("radius ",radius,"\n")
        for(sim in 1:nsim){
          row = row +1
          X = rModel1(num.genes=num.genes,num.b=num.b,
                     winbase=winbase,lambda=lambda,
                     radius=radius)
          ## Evaluation of errors
          tryCatch(
            {
              y = do_desc(X=X,pairs_sel=pairs_to_eval,descs="Kcross",
```

```

                                rvalues=rvalues)
    sig = oneRow(y$kc,method="all")
    errors[row,] = c(num.genes,num.b,lambda,radius,sim,
                    unlist(do_errors(sig,ptr$rows_sig,
                                    ptr$rows_to_eval)))
  }, error = function(msg){
    errors[row,] = c(num.genes,num.b,lambda,radius,sim,
                    NA,NA)
  })
  }
}
}
  save(errors,file="results/simulation/Model1.rda")
}
save(errors,file="results/simulation/Model1.rda")

```

5 Neyman-Scott process

We generate pairs of dependent point processes. For each pair, the first and second component of each pair correspond with the parents and offsprings in a Neyman-Scott process. The intensity of parents (Poisson point pattern) is κ and the mean number of offsprings per parent is μ . It is implemented in the file `funtions_srt.R` in the function `rModel2`.

```

winbase = owin(c(0,1),c(0,1))
num.genes.values = seq(50,100,10)
num.b.values = seq(20,40,5)
kappa.values = seq(10,20,5)
mu.values = seq(2,6,2)
scale = .01
nsim = 100
errors = matrix(NA,ncol=7,nrow=length(num.genes.values)*
               length(num.b.values)*
               length(kappa.values)*
               length(mu.values)*nsim)

rvalues = seq(0,.15,length.out=20)
row = 0
for(num.genes in num.genes.values){
  cat("num.genes ",num.genes,"\n")

```

```

pairs_to_eval = t(combn(num.genes,2))
for(num.b in num.b.values){
  cat("num.b ",num.b,"\n")
  pairs_sig = cbind( 2*(1:(num.b/2))-1, 2*(1:(num.b/2)))
  ptr = pairs_to_rows(pairs_sig,pairs_to_eval)
  for(kappa in kappa.values){
    cat("kappa ",kappa,"\n")
    for(mu in mu.values){
      cat("mu ",mu,"\n")
      for(sim in 1:nsim){
        row = row +1
        X = rModel2(num.genes=num.genes,num.b=num.b,
                    winbase=winbase,kappa=kappa,mu=mu,
                    scale=.01)
        ## Evaluation of errors
        tryCatch(
          {
            y = do_desc(X=X,pairs_sel=pairs_to_eval,descs="Kcross",
                        rvalues=rvalues)
            sig = oneRow(y$kc,method="all")
            errors[row,] = c(num.genes,num.b,kappa,mu,sim,
                            unlist(do_errors(sig,ptr$rows_sig,
                                              ptr$rows_to_eval)))
          }, error = function(msg){
            errors[row,] = c(num.genes,num.b,kappa,mu,sim,NA,NA)
          })
      }
    }
  }
  save(errors,file="results/simulation/Model2.rda")
}
save(errors,file="results/simulation/Model2.rda")

```

6 Multitype hard core point process

We generate a given number of multitype hard core point process. It is implemented in the file `functions_srt.R` in the function `rModel3`.

```

winbase = owin(c(0,1),c(0,1))
num.genes.values = seq(50,100,10)
num.b.values = seq(20,40,5)
beta.values = seq(10,20,5)
hmin.values = seq(.01,.05,.01)
nsim = 100
errors = matrix(NA,ncol=7,nrow=length(num.genes.values)*
                length(num.b.values)*
                length(beta.values)*
                length(hmin.values)*nsim)

rvalues = seq(0,.15,length.out=20)
row = 0
for(num.genes in num.genes.values){
  cat("num.genes ",num.genes,"\n")
  pairs_to_eval = t(combn(num.genes,2))
  for(num.b in num.b.values){
    cat("num.b ",num.b,"\n")
    pairs_sig = cbind( 2*(1:(num.b/2))-1, 2*(1:(num.b/2)))
    ptr = pairs_to_rows(pairs_sig,pairs_to_eval)
    for(beta in beta.values){
      cat("beta ",beta,"\n")
      for(hmin in hmin.values){
        cat("hmin ",hmin,"\n")
        for(sim in 1:nsim){
          row = row +1
          X = rModel3(num.genes=num.genes,num.b=num.b,
                      winbase=winbase,beta=beta,hmin=hmin)
          ## Evaluation of errors
          tryCatch(
            {
              y = do_desc(X=X,pairs_sel=pairs_to_eval,descs="Kcross",
                          rvalues=rvalues)
              sig = oneRow(y$kc,method="all")
              errors[row,] = c(num.genes,num.b,beta,hmin,sim,
                              unlist(do_errors(sig,ptr$rows_sig,
                                                  ptr$rows_to_eval)))
            }, error = function(msg){
              errors[row,] = c(num.genes,num.b,beta,hmin,sim,NA,NA)
            })
        }
      }
    }
  }
}

```

```
    }  
  }  
  save(errors,file="results/simulation/Model3.rda")  
}  
save(errors,file="results/simulation/Model3.rda")
```