Statistical inference for Gibbs marked point processes with interactions through exploratory analysis and posterior sampling. Application on astronomical data

D. Gemmerlé, N. Gillot and R. S. Stoica

Contents

If the ancient gods of computing are with us, then this practical class will present:

DRlib is a C++ library for statistical modelling, simulation and inference using Gibbs marked point processes with interactions. One of the key concepts underlying the development of this library is the consideration of the point process model as comprising components. These components take into account the possible available covariates and point interactions.

- In a first example related to the homogeneous Poisson point process, we will present the overall structure of the library and its constituent parts. This allows us to consider purely random homogeneous point patterns.
- A modification of one component enables us to build inhomogeneous Poisson processes in order to consider purely random inhomogeneous point patterns.
- The flexibility of the library will be demonstrated by making the transition to the Strauss process in order to introduce repulsion between points.
- Finally, a three-component model is built. These components enable us to consider random point patterns (Poisson process) that exhibit attraction over short distances (area-interaction process) and repulsion over longer distances (Strauss process).

Regarding the application to real data, the data will first be presented and classical summary statistics will be used to obtain information related to the model to be implemented.

Don't worry — things may turn out well!

1 Generalities

DRlib is a C++ library for modelling, simulating and performing statistical inference using marked Gibbs point processes. Its purpose is to complement existing tools, such as the **spatstat** library in R Baddeley, Rubak, and Turner 2016, with reliable and efficient C++ code that enables intensive Bayesian Markov chain Monte Carlo (MCMC) inference. This work builds on the **MPPLIB** library, which was mainly developed by Lieshout and Stoica 2006. The current version of the library is freely available at:

`https://gitlab.univ-lorraine.fr/labos/iecl/drlib`

DRlib analyses point patterns using the following tools:

- Modelling: implements the probability density that models the characteristics of an observed pattern. These characteristics are associated to the model components which are controlled by the model parameters.
- Simulation: implements a Metropolis-Hastings algorithm, a Markov chain Monte Carlo (MCMC) algorithm that samples the proposed probability density model. In other words, it produces point patterns exhibiting the desired considered characteristics.
- Statistical inference: samples the posterior distribution, i.e. the distribution of the parameters conditioned on the observed pattern. This step uses the ABC Shadow algorithm. Sampling from the posterior distribution enables parameter-based inference (e.g. parameter estimation and statistical testing).

The probability density of a marked Gibbs point process controlling the position of points in the pattern $\mathbf{x}$ writes as :

$$f(\mathbf{x}|\theta) = \frac{\exp\left[-U(\mathbf{x}|\theta)\right]}{c(\theta)}$$

with $U(\mathbf{x}|\theta)$ the energy function and $c(\theta)$ the partition function. Here we consider energy functions of the form

$$U(\mathbf{x}|\theta) = -\langle t(\mathbf{x}), \theta \rangle = -\sum_{i=1}^{m} t_i(\mathbf{x})\theta_i$$

where $t_i(\mathbf{x})$ represents the characteristic of the pattern and θ_i its associated parameter or weight. The number of the characteristics m is pre-fixed. These characteristics are the sufficient statistics of the model, that is the all the needed knowledge in order to determine the parameters θ. From a programming perspective each statistic will be considered a particular component of the model. The sum of $t_i(\mathbf{x})\theta_i$ will be calculated with specialized `C++` objects of type **Component**.

2 Get the practical class files and project organization

2.1 Get the practical class files

Create your own working directory. In a terminal window, first make sure that no directory is named ***Excosm-Tartu-Course*** in the working directory. Then create a clone of the files' course:

```
git clone https://gitlab.univ-lorraine.fr/labos/iecl/Excosm-Tartu-Course.git
```

This command creates a directory ***Excosm-Tartu-Course*** that contains several subdirectories.

The Unix rights on .bat programs with git could be wrong. To solve that :

```
cd Excosm-Tartu-Course
find . -name \*.bat -exec chmod +x {} \;
```

2.2 Typical organisation of the files for a project built with DRlib

The files related to a typical **DRlib** project are grouped into a directory and this directory contains 2 subdirectories ***C++*** and ***R*** that have the following structure:

C++	contains all needed files for *C++* computation		
	->	*DATA*	Initial data or important intermediary results
	->	*EXEC*	executables (.exe issued from g++ and shell .bat)
	->	*PARAMS*	parameters files
	->	*RESULTS*	results files (pattern, statistics)
	->	*SRC*	sources files and Makefile
R	contains some visualization *Rstudio* programs		

A typical development phase is:

$$\begin{tikzcd}
 & DATA & \searrow & & & \\
SRC \arrow[r, "\text{make}"] & EXEC & {} \arrow[r, "\text{execution}"] & RESULTS & & \\
 & PARAMS & \nearrow & & \searrow & \\
 & & & R & {} \arrow[r, "\text{Rscript}"] & \begin{array}{l}\text{plot simulation results}\\ \quad\text{plotPattern-Stat\_rstudio.R}\\ \text{display inference results}\\ \quad\text{displaySampling\_rstudio.R}\end{array}
\end{tikzcd}$$

3 Modelling, simulation and inference tools. Work on synthetic data.

3.1 Homogeneous Poisson process

The first model presented is the homogeneous Poisson process. The probability density of the process is expressed as:

$$p(\mathbf{x}) \propto \beta^{n(\mathbf{x})} \propto \exp(n(\mathbf{x}) \log \beta)$$

with $n(\mathbf{x})$ the number of points in the configuration $\mathbf{x}$ and $\beta > 0$ the constant intensity parameter.

All the necessary files can be found in the directory ***01-Poisson-Homogen*** which is organised as follows:

C++	->	*EXEC*		executables (.exe issued from g++ and shell .bat)
			->	abc_simul.bat
			->	model_simul.bat
			->	sim_abc.exe
			->	sim_model.exe
	->	*PARAMS*		parameters files
			->	p_abc_*.txt
			->	p_sim_model_*.txt
	->	*RESULTS*		results files (pattern, statistics)
	->	*SRC*		sources files and Makefile

the file *.bat contain a line with the corresponding executable with the parameter file.

3.1.1 Compilation

The compilation is performed using make in the ***SRC*** directory. The *.exe executables are then generated in the ***EXEC*** directory. To compile the files :

```
make
```

To erase the files produced by compilation (*.obj and *.exe) do the following :

```
make clean
```

This allows to re-start the compilation from the very beginning.

3.1.2 Execution

The executable programs are located in the ***EXEC*** directory. Example execution commands are provided in the files model_simul.bat, abc_simul.bat depending on the case.

The program model_simul.bat uses the MCMC sampling method to generate point configurations following a given point process model. For each of the generated point configurations sufficient statistics are computed and saved in a file.

The program abc_simul.bat approximateley samples the posterior distribution of the model parameters using the ABC Shadow method. At each step, the program saves the current parameters value.

These programs use as entries the parameter files available in the ***PARAMS*** directory.

Important remark : the Unix rights on .bat programs with git could be wrong.
To solve this: chmod +x *.bat

3.1.3 Simulation

In the following, some key elements of the code are presented in order to understand the use of the library.

First selected code lines are from *sim_model.cpp*

```
std::vector<GenericComponent*> list_component_model;

// ====================================================
// POISSON
// For inhomonegenousity see
//    - PoissonComponent :: beta_parameter_model_func( Event* e )
//    - PoissonComponent :: beta_exp_parameter_model_func( Event* e )
PoissonComponent poissonComponent(logBetaP,K_x,K_y,0);}
list_component_model.push_back(&poissonComponent);

// Model
//   Don't forget Event* Model :: newEvent() const redefinition
//   Don't forget Event* Model :: changeEvent() const redefinition
Model the_model(&list_component_model,K_x,K_y);

// Sampler
MH algo_sampler(&the_model,time_mh,pbirth,pdeath);

// Pattern
Pattern xpattern;
MarkedEvent::numberOfMark = 0;

int nbi;
for(nbi=0;nbi<nbiter;nbi++)
{
  cout << "Iteration : "<< nbi << "\n";
  // in the method sim execution x times of the sampler transition
  algo_sampler.sim(xpattern);
```

```
145     // at each iteration a pattern is taken
146     xpattern.printInFile(name_sample_pattern);
147     the_model.computeStatistics(xpattern);
148     the_model.printStatistics(xpattern,name_statistics);
149   }
```

Lines 124 and 125 define the Poisson **Component** and add it to the list of **Component**. This list with the size of domain defines the **Model** line 130. The **MH** sampler and the **Pattern** together with the number of marks attached to a point are also defined. The loop lines 140-149 call iteratively the *MH.sim* to generate realisations from the model.

Work to be done :
Recall that the homogeneous Poisson process density's is $p(\mathbf{x}) \propto \beta^{n(\mathbf{x})} \propto \exp(n(\mathbf{x}) \log \beta)$. This model generates random configurations of points exhibiting no interactions. Its sufficient statistic is the pattern's number of points and the corresponding parameter is β. This parameter thus controls the pattern's amount of points, more precisely, it controls the average number of points of the patterns.

1. **Compilation and execution of the program.**
   Go to the directory ***01-Poisson-Homogen/C++/SRC***
   Compilation of the program:

   ```
   make clean
   make
   ```

   Go to the directory ***01-Poisson-Homogen/C++/EXEC***
   2 executable programs are generating sim_model.exe and sim_abc.exe (will explain later)
   run the following program :

   ```
   ./model_simul.bat
   ```

   this program run sim_model.exe with an example parameter file.

If you take a look at the parameter file in
01-Poisson-Homogen/C++/PARAMS/p_sim_model_poisson_inhomogen.txt, the process will be simulated in the window $[0, 1] \times [0, 1]$ (domain's dimension are given by K_x and K_y). The number of MH steps is set to 1000, then we iterate this 1000 steps 3000 times in order to generate 3000 realisations of the homogeneous Poisson point process with parameter value $\log(\beta) = 4.6$. So the expected number of points is approximately 100. Two files are created in ***01-Poisson-Homogen/C++/RESULTS*** : *Y_model_poisson_homogen.txt* and *stat_model_poisson_homogen.txt*, the last simulated pattern 2D coordinates and the statistics after each realisation respectively.

2. **Display the results** Open in *Rstudio* the following file :
   01-Poisson-Homogen/R/plotPattern-Stat_rstudio.R
   This .R file reads the .txt files *Y_model_poisson_homogen.txt* and *stat_model_poisson_homogen.txt* to plot the last simulated pattern together with the cumulative mean of the statistics, here the number of points. We

can see that the last simulated pattern countains 98 points (first line of the file), which is not exactly the parameter value. However, the mean of the number of points is very close to the parameter value.

In order to simulate other realisations with different parameter values, we can now change the parameter values inside the parameter file.
01-Poisson-Homogen/C++/PARAMS/p_sim_model_poisson_homogen.txt.

3. **Run different models and interpret the obtained results.** Use the editor of your choice, you can modifiy, in the directory ***01-Poisson-Homogen/C++/PARAMS*** the file *p_sim_model_poisson_homogen.txt*. For instance, you can modify the $\log\beta$ parameter from 4.6 to 5.0 and rerun the previous program using ./model_simul.bat

It can be seen that the average number of points in a realisation has increased from $e^{4.6} = 99.5$ to $e^{5.0} = 148.4$.

Important: it is more convenient from a numerical perspective to slightly transform the formulas of the probability densities describing the considered models in order to be able to work with $\log\beta$ instead of β.

3.1.4 Inference: posterior sampling

The previous compilation also generated an executable named sim_abc.exe. Its purpose is to sample from the posterior distribution. The key assumption is that the observed pattern is generated by the realization of a parametric model. Within this context, posterior sampling means to obtain a set of parameter values *i.e.* models, that able to produce patterns exhibiting statistics close to the observed ones. The values of the most probable parameters represent the models that have the most chances to reproduce patterns with the observed characteristics. For the homogeneous Poisson point process, the sufficient statistic allowing parameter estimation is the number of points in a configuration. The program sim_abc.exe takes a parameter file. As previously, the batch file abc_simul.bat runs the executable using a pre-defined parameter file.

Work to be done:
The parameter file contains all the information required to run the posterior sampling algorithm. Firstly, it contains the seeds of the random number generator, enabling fine-tuning during program development and reproducibility of results. This is followed by the size of the point process simulation domain, the value(s) of the sufficient statistics calculated for the observed data and the needed information related to the search domain for the model parameters values (search interval, search step and initial position). At each step of the algorithm, the program collects the values of the parameters and saves them in a file (here ../RESULTS/theta_abc_poisson_homogen.txt). The last simulated auxiliary pattern is also saved in a file (here../RESULTS/Y_abc_poisson_homogen.txt).

As explained previously, for reasons of numerical stability of the perfomed computations, the logarithm of the different paremeters values are used in the corresponding parameter files, respectively.

Here, the purpose is to run the inference based on the observations. The observed statistic is the mean number of points 99.58 (this value may be different weither you use Linux or Mac-OS due to the difference in seeding), the number of iteration for the ABC Shadow procedure is set to 50 000 and the parameter is changed 50 times at each iteration, with pertubation parameter $\delta = 0.01$. The initial $\log \beta$ value is 4.5, the minimum and maximum value are 1 and 9 respectively. The number of steps to generate an auxiliary pattern is 250. Here, we expect to find back the parameter we used to sample the different realisation because the observed statistics are based on numerous observation, making the inference more precise. Then, the results (the last auxiliary pattern and the sample of $\log \beta$) will be displayed thanks to the .R file.

1. **Run the ABC Shadow Algorithm.** Go to the directory ***01-Poisson-Homogen/C++/EXEC***
   run the following program :

   ./abc_simul.bat

2. Print and interpret the results. Open in *Rstudio* the following file :

   01-Poisson-Homogen/R/plotPattern-Stat_rstudio.R

   run it (possible if necessary to go to solutions)

The histogram of the obtained values of β represents an approximation of the desired posterior density. It can be seen that the histogram is unimodal, as expected. Its mode is close to the theoretical value 4.6. The value of the estimated maximum likelihood is therefore close to the true parameter values which here is 4.6.

In conclusion, Homogeneous Poisson is good but Inhomogeneous Poisson is better...

3.2 Inhomogeneous Poisson process

In a homogeneous Poisson process, the points are spread uniformly and independently across the space domain. There are many situations in which, mathematically speaking, we may not want to abandon the hypothesis of independence. However, information and evidence relating to the non-uniform distribution of points in space is available. For instance, galaxies tend to be found close to the spines of galactic filaments. In this case, the intensity parameter is no longer constant over the space domain. The process becomes inhomogeneous.

In this context, the probability density of an inhomogeneous Poisson point process writes as

$$p(\mathbf{x}) \propto \prod_{i=1}^{n(\mathbf{x})} \beta(x_i)$$

with the intensity function $\beta(x_i) \geq 0$ not necessary constant.

3.2.1 Inhomogeneity

From a programming perspective, one needs to transform the constant intensity of an homogeneous Poisson process into a location dependent intensity function, in order to obtain an inhomogeneous Poisson process.

Have a look at the following `DRlib` piece of code (lines 96-110) extracted from the file :

01-Poisson-Homogen/C++/SRC/poissoncomponent.cpp

```
96   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
97   double PoissonComponent :: beta_parameter_model_func( Event* e )
98   {
99     return parameter_model;
100
101  }
102
103  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
104  double PoissonComponent :: beta_exp_parameter_model_func( Event* e )
105  {
106    return exp(beta_parameter_model_func(e));
107
108  }
109
110  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

In order to change it and to obtain what we want, it's time to go in directory ***02-Poisson-Inhomogen***

```
cd {GoodPlace}/02-Poisson-Inhomogen
```

A new subdirectory ***DATA*** is present in the directory ***C++***. In this new directory, the file *lambda1_matrix_from_dist_map_xy.dat* contains the representation of discretised intensity function under the form of a matrix (see Fig. 1b). The matrix is computed using the shortest distance from the points of a given grid to an already existing filamentary pattern. Before getting the final results, a smooting kernel and a normalisation to 1 are applied (see Fig. 1a). The sum of all the grid values approximates the integral of the intensity function over the considered domain W, hence we have

$$\int_W \beta(x)dx \approx \sum_{j=1}^{m} \hat{\beta}(w_i)$$

with $\hat{\beta}$ the smoothed intensity funcion, w_i the grid points and m the size of the grid.

The practical implication is that using this intensity function, a proper multiplicative constant should be added in front of the intensity function in order to control the average number of points in a configuration. Furthermore, discretisation effects should be taken into account as well. Within this context, the probability density of the considered inhomogeneous Poisson process has the following form:

$$p(\mathbf{x}) \propto \prod_{i=1}^{n(\mathbf{x})} \tilde{\beta}\beta(x_i)$$

with $\beta(x_i)$ a known intensity function that integrates on W to 1 and $\tilde{\beta}$ the unknown parameter controlling the number of points in the configuration, that should be estimated.

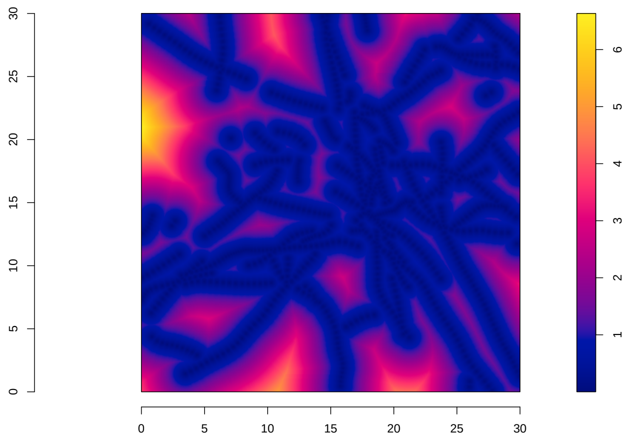
3.2.2 A modified PoissonComponent : PoissonInhomogenComponent

To take the inhomogeneity into account, in addition to the fairly simple modifications of the files *sim_model.cpp* and *sim_abc.cpp*, the **PoissonComponent** must be adapted into **PoissonInhomogenComponent**.

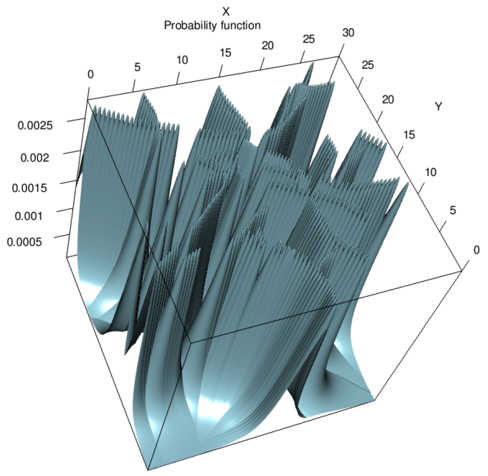
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
double PoissonInhomogenComponent :: beta_parameter_model_func( Event* e )
{
  return parameter_model;

}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
double PoissonInhomogenComponent :: beta_exp_parameter_model_func( Event* e )
{
  double x_event,y_event;
  x_event=e->getX();
  y_event=e->getY();
  double area_elt;
  area_elt=(width_window/nb_column_lambda1)*(height_window/nb_line_lambda1);
  int x_grid_matrix,y_grid_matrix;
  x_grid_matrix=static_cast<int>(x_event/(width_window/nb_column_lambda1));
  y_grid_matrix=static_cast<int>(y_event/(height_window/nb_line_lambda1));
```



(a) Distance Map from filaments on domain [0,30]x[0,30]



(b) From Distance Map, intensity matrix 512x512 on domain [0,30]x[0,30]. The sum of 512x512=262144 values is 1.

```

    return ( lambda1matrix[y_grid_matrix][x_grid_matrix]
                    /area_elt*exp(beta_parameter_model_func(e)) );

}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

In this case, the intensity depends on the location of the point in the domain, as described in the figures above.

Work do be done :
Reminder : the model parameter in the parameter file is not β but $\log \beta$.

In addition to the parameters presented in the homogeneous case, the parameter file *p_sim_model_poisson_inhomogen.txt* includes a line that is the name of the file containing the inhomogeneity unit matrix (in our case :
../DATA/lambda1_matrix_from_dist_map_xy.dat)
This matrix describes the inhomogeneity we want to take into account for the simulations.

1. Go to the directory ***02-Poisson-Inhomogen/C++/SRC***
   compile and run the samples programs ./model_simul.bat and ./abc_simul.bat

If you take a look at the parameter file in
02-Poisson-Inhomogen/C++/PARAMS/p_sim_model_poisson_homogen.txt, the process will be simulated in the window $[0,1] \times [0,1]$ (domain's dimension are given by K_x and K_y). The number of MH steps is set to 1000, then we iterates this 1000 steps 10000 times to generate 10000 realisations of the inhomogeneous Poisson point process with parameter value $\log(\beta) = 4.60$. This time, the points should be spread differently than the homogeneous case.

For the inference, we used the same parameters as previously, except that we took a different number of points as inputs and ran only 20000 iterations of the ABC Shadow algorithm. The program creates 2 .txt file named *theta_abc_poisson_inhomogen.txt* and *Y_abc_poisson_inhomogen.txt* (containing the last simulation).

Now, it's time to display the results.

2. As previsouly, using *Rstudio* , you can check that :
   - the points organize along certain regions
   - the inference indeed provides a good approximation of the correct number of points. (possible if necessary to go to solutions)

3.3 Inhomogeneous Marked Strauss process

Inhomogeneity may be seen as an effect due to the nature of the space to which the points belong. However, it is also possible that points interact with each other. For example, two points may exhibit repulsion if they are too close to each other. This is the case in the repulsive Strauss process.

Let us further assume that the points are marked. A marked point pattern is a sequence $\mathbf{x} = \{x_1, \ldots, x_n\}$ with $x_i = (w_i, m_i)$ a point with position $w_i \in W$ and mark $m_i \in M$, where W and M are the location and mark spaces, respectively. If the marks are positive real values, the marked point x_i can be seen as a disk centred in w_i with radius m_i.

There are several ways to introduce interactions between points using the Strauss process philosophy, while taking into account the positions of points and their marks. For more detailed considerations and related bibiliography on the subject the reader may refer to Baddeley, Rubak, and Turner 2016; Møller and Waagepetersen 2004; Lieshout 2000; Stoica 2025.

Let us consider the Strauss process given by the following probability density:

$$p(\mathbf{x}) \propto \left[\prod_{i=1}^{n(\mathbf{x})} \beta(x_i) \right] \gamma^{s(\mathbf{x})}$$

with the statistic $s(\mathbf{x})$ given by

$$s(\mathbf{x}) = \sum_{i,j=1,i<j}^{n(\mathbf{x})} \mathbf{1}\{d(w_i, w_j) \leq m_i + m_j\}.$$

This statistic counts the pairs of points that are too close, or more easy to see it, the pairs of disks that tend to overlap. The model is well defined for $\beta > 0$ and γ in $]0, 1]$.

The DRlib uses the previous construction of the intensity of the Poisson process and adds interactions in order to construct and manipulate the Strauss model.

It's time to go in directory ***03-Poisson-Inhomogen-MarkedStrauss***

```
cd {GoodPlace}/03-Poisson-Inhomogen-MarkedStrauss
```

In the subdirectory ***C++/SRC*** we find the same file names *sim_model.cpp* and *sim_abc.cpp*

Let's take a closer look at an excerpt from *sim_model.cpp*

```
std::vector<GenericComponent*> list_component_model;
// ==================================================
// POISSON
// For inhomonegenousity see
//    - PoissonComponent :: beta_parameter_model_func( Event* e )
//    - PoissonComponent :: beta_exp_parameter_model_func( Event* e )
PoissonInhomogenComponent poissonInhomogenComponent(logBetaP,K_x,K_y,0);
poissonInhomogenComponent.loadFromFileLambda1matrix(name_lambda1_matrix);
list_component_model.push_back(&poissonInhomogenComponent);
// Model
```

```
147 //   Don't forget Event* Model :: newEvent() const redefinition
148 //   Don't forget Event* Model :: changeEvent() const redefinition
149 // ===================================================
150 // STRAUSS
151 MarkedStraussComponent straussrandomComponent(logGammaS,K_x,K_y,
152             radius_random_min_Strauss,radius_random_max_Strauss,
153             alea_type_Strauss,alea_parameter_Strauss,1,0);
154 list_component_model.push_back(&straussrandomComponent);
155 // ===================================================
156 Model the_model(&list_component_model,K_x,K_y);
157
158 // Sampler
159 MH algo_sampler(&the_model,time_mh,pbirth,pdeath);
160
161 // Pattern
162 Pattern xpattern;
163 MarkedEvent::numberOfMark = 1;
164
165 int nbi;
166 for(nbi=0;nbi<nbiter;nbi++)
167   {
168     cout << "Iteration : "<< nbi << "\n";
169     // in the method sim execution x times of the sampler transition
170     algo_sampler.sim(xpattern);
171     // at each iteration a pattern is taken
172     xpattern.printInFile(name_sample_pattern);
173     the_model.computeStatistics(xpattern);
174     the_model.printStatistics(xpattern,name_statistics);
175   }
```

Some explanations:

- the model is composed of 2 **Component** that are added to the list of **list_component_model** (lines 145 and 154).

- the mark number of each point is 1. This mark represents the radius of the disk of influence of each point in the Strauss process. The number of marks of each point is defined line 163 (in red)
  MarkedEvent::numberOfMark = 1;

- in the definitions of the **Component poissonInhomogenComponent** (line 143) there is a parameter 0 in red that corresponds to the rank (its position) of the component in the list of **list_component_model**. **straussrandomComponent** is in second position so of rank 1. The index (or position) of the mark concerning the **straussrandomComponent** is in first position (since there is only one brand) and is therefore equal to 0 (line 153).

  This mechanism is needed for models exhibiting two or more components.

This time, in addition to the inhomogeneity, we're working with a new model, the Strauss model. We're looking to control the amount of connected $r-$pairs, where r is a fixed radius. The expected behaviour is the following : β will control the amount of points (the greater it is, the larger will be the amount of points) and γ will control the strength of repulsion. If $\gamma = 1$, the model boils down to an inhomogeneous Poisson point process and the closer γ is to 0, the stronger the repulsion will be.

1. Compile and run the programs ./model_simul.bat and ./abc_simul.bat
2. Note the repulsive behaviour on the last realisation and how the number of points has evolved. (possible if necessary to go to solutions)
3. Modify the γ parameter and re-run the programs. Observe the repulsion phenomenon through the obtained point configurations and the behaviour of the statistics.

3.4 Inhomogeneous Marked Area-Interaction process

The interactions in the Strauss model are distance based. Mathematical requirements impose that this model can be used only for modelling repulsive point patterns. It is perfectly possible to introduce different type of interactions. For instance the area-interaction point processes controls the area occupied by the underlined point pattern. Furthermore, the model is able to simulate both type of patterns, clustered or regular (repulsive). Its probability density is as follows

$$p(\mathbf{x}) \propto \left[\prod_{i=1}^{n(\mathbf{x})} \beta(x_i)\right] \gamma^{-a_r(\mathbf{x})}$$

with $n(\mathbf{x})$ the number of points in the configuration. The statistic $a_r(\mathbf{x})$ is given by:

$$a_r(\mathbf{x}) = \frac{\nu\left[\bigcup_i^{n(\mathbf{x})} b(w_i, m_i)\right]}{\pi r^2}$$

where $b(x_i, m_i)$ represents the balls of center x_i and radius m_i, ν denotes the area of a set and r is a pre-defined range parameter covering to the average value of the radius of a disk in the marked points' configuration. The numerator in the statistic $a_r(\mathbf{x})$ represent the area of the disks configuration. The denominator is a normalisation factor that allows to interpret $a_r(\mathbf{x})$ as the number of disks of average radius r that occupy the area given by a random configuration of disks. The model is well-defined for $\gamma > 0$. If $\gamma > 1$ the process tends to produce clustered patterns, otherwise the patterns produced tend to exhibit a repulsive character.

It's time to teleport to the directory ***04-Poisson-Inhomogen-MarkedAreaInt*** cd {GoodPlace}/04-Poisson-Inhomogen-MarkedAreaInt

Work To Be Done: Now we're going to simulate and run the inference for this new model.

1. Compile and run the samples programs ./model_simul.bat and ./abc_simul.bat
2. In our case $\log(\gamma_A) = -0.3$. The process tends to spread points.

   Observe the repulsive phenomenon through the obtained point configurations and the behaviour of the sufficient statistics, that is the number of points and the normalised area of the pattern. If necessary go directly to solutions)

3. Modify the model parameters. Replace the value $\log(\gamma_A) = -0.3$ by $\log(\gamma_A) = 1$.

   For doing this, in file *p_sim_model_poisson_inhomogen_markedareaint.txt*, modify the line after //AreaInt_log_GammaA by replacing −0.3 by 1 (line 24) and re-run the programs. The process is attractive, and the AreaInt component tends to gather the points together.

3.5 Inhomogeneous Marked Strauss Area-Interaction process

Galaxies in our Universe tend to form clusters that align along filaments like pearls on necklace Stoica 2025; Tempel, Kipper, et al. 2014. In this case, we may consider that the galaxy distribution exhibits short range attraction in order to form clusters, a long range repulsion in order to separate the cluster and also an inhomogeneous aspect induced by the filamentary pattern. It is perfectly possible to use **DRlib** in order to build marked point processes showing these types of characteristics. Please consider the following point process model:

$$p(\mathbf{x}) \propto \left[\prod_{i=1}^{n(\mathbf{x})} \beta(x_i)\right] \gamma_S^{s(\mathbf{x})} \gamma_A^{-a_r(\mathbf{x})}$$

with

$\beta(\cdot)$	the inhomogeneity function
$n(\mathbf{x})$	the number of points in $\mathbf{x}$
γ_S	interaction parameter of the Strauss component
$s(\mathbf{x})$	number of the interacting pairs of the Strauss component
γ_A	interaction parameter of the Area -Interaction component
$-a_r(\mathbf{x})$	normalised surface of the pattern $\mathbf{x}$ of the Area-Interaction component

From a programming perspective, the range parameters in the Strauss and Area-Interaction components may be considered as marks.
It's time to pass through the stargate to the directory ***05-Poisson-Homogen-MarkedStrauss-MarkedAreaInt***

```
cd {GoodPlace}/05-Poisson-Homogen-MarkedStrauss-MarkedAreaInt
```

In the subdirectory ***C++/SRC*** we find the same file names *sim_model.cpp* and *sim_abc.cpp*

Let's have a closer look at an except from *sim_model.cpp*

```
std::vector<GenericComponent*> list_component_model;

// ==================================================
// POISSON
// For inhomonegenousity see
//   - PoissonComponent :: beta_parameter_model_func( Event* e )
//   - PoissonComponent :: beta_exp_parameter_model_func( Event* e )
PoissonComponent poissonComponent(logBetaP,K_x,K_y,0);
list_component_model.push_back(&poissonComponent);

// ==================================================
// STRAUSS
MarkedStraussComponent straussrandomComponent(logGammaS,K_x,K_y,
                     alea_type_Strauss,alea_parameter_Strauss,1,0);
list_component_model.push_back(&straussrandomComponent);

// ==================================================
// AREAINT
MarkedAreaIntComponent areaintrandomComponent(logGammaA,K_x,K_y,
                     radius_random_min_AreaInt,radius_random_max_AreaInt,
                     alea_type_AreaInt,alea_parameter_AreaInt,
                     resolution,2,1);
list_component_model.push_back(&areaintrandomComponent);

// ==================================================
// Model
//   Don't forget Event* Model :: newEvent() const redefinition
//   Don't forget Event* Model :: changeEvent() const redefinition
Model the_model(&list_component_model,K_x,K_y);

MH algo_sampler(&the_model,time_mh,pbirth,pdeath);

Pattern xpattern;
```

```
MarkedEvent::numberOfMark = 2;
```

Some explanations: the model is made up of three **Component** numbered 0, 1 and 2. Each point is associated with a mark which is a pair (*radius for Strauss*, *radius for AreaInt*). In our case, the *radius for Strauss* will be referenced as component 0 of the mark, and the *radius for AreaInt* will be referenced component 1 of the mark.

Work To Be Done:

1. Compile and run the programs ./model_simul.bat and ./abc_simul.bat
2. Note the phenomenon of aggregation on the last realization and how the number of points has evolved (possible if necessary to go to solutions)
3. Modify the model parameters and re-run the programs. Observe the repulsion or the attraction phenomenon through the obtained point configurations and the behaviour of the statistics.

3.6 Some extra explanations

3.6.1 newEvent: not so trivial problem

Intuitively, extending programs from un-marked point patterns to the marked ones may appear as a simple task. This is not always the case. For the presented models, we detail the current choice done for the DRib construction. Our aim is to be able to propose general mathematical and computing construction allowing to manage, as desired - separetely or jointly - the position of points, their marks, the interactions component and their associated parameters.

In the following, we detail the solutions actually adopted in DRlib for the construction of the presented model. We believe that after thorough lecture, the reader will be able to extend the type of solutions proposed. All comments and suggestions are the most welcome.
To illustrate the problem, let's take an extract from the file *model.cpp*

```
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Event* Model :: newEvent() const
{
  Point p(xrange()*whran.uniform(),yrange()*whran.uniform());
  MarkedEvent* a_marked_event = new MarkedEvent(p);

  int i;
  int numberOfComponent=list_component_model->size();

  for (i=0;i<numberOfComponent;i++)
    {
      (list_component_model->at(i))->markNewEvent(a_marked_event);
    }

  return a_marked_event;

}

//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```
This implementation associate the same mark to all the components involved in the model. It is clear that for different models this solution has to be changed.

3.6.2 Component specialization

In some specific cases, an adapted version of an already existing component can be created. Our team worked on developing a type Strauss model: the StraussCrown (Gillot et al. 2024). This model uses the same statistics as the Strauss model but this time the number of connected pairs is calculated by counting the number of pairs between two radii r_1 and r_2 ($r_1 < r_2$).
The profile of constructor object is in the form :

```
StraussCrownFixedComponent (double parameter_model_parameter,
       double width_parameter, double height_parameter,
       double radius_parameter_1, double radius_parameter_2,
       int rank_parameter);
```

Note that since in this case the radius of the Strauss process is fixed, it has been implemented as a model parameter and does not appear as a point mark. The adaptation of the computeStatistic method for the StraussCrownFixedComponent is as follows:

```
void StraussCrownFixedComponent::computeStatistic( StandardPattern& p )
{
  int sz = p.getSize();
  double pairs = 0.0;

  for( int i=0; i<sz-1; i++ )
    {
      for( int j= i+1; j<sz; j++ ) {
        if(( r1 <= p.getEvent(i)->getDistance(p.getEvent(j))) &&
           ( p.getEvent(i)->getDistance(p.getEvent(j)) <= r2) )
          pairs=pairs+1.0;
      }
    }

  current_statistic=pairs;

}
```

The modular organization of the library means that components can be reused in a single or duplicated way (with different parameters), and existing components can be adapted.

4 Real data application

4.1 Data presentation

The data used in this practical work is given by the positions of galaxies in a region of our Universe Hurtado-Gil et al. 2021; Tempel, Stoica, et al. 2014. The data contains also the spine of the cosmic filaments detected using the Bisous model Stoica 2025; Tempel, Stoica, et al. 2014.
The aim of this practical lesson is to fit a point process model with interactions to the galaxies positions conditioned by the observation of the cosmic filaments spines.

cd {GoodPlace}/10-DATA-3D

In this directory there are three **R** scrispts that can be used in Rstudio to display the galaxies, the cosmic filaments and their superposition , respectively:

display3D_gal_rstudio.R
display3D_fil_rstudio.R
display3D_gal_fil_rstudio.R

If it isn't working, you can go directly to solutions.

For pedagogical reasons, during this practical work we will use the two-dimensional projections of the data. The data is available through the directory ***{GoodPlace}/Results*** directory. The *gal_sample_xy.dat* and *fil_sample_xy.dat* files contain respectively the 2D coordinates of the projected 3D coordinates of the galaxies and filaments on the plane 0xy.

To visualize the superimposition of 2D galaxies and filaments coordinates, you can run the superimpose_fil_gal.R program in the ***{GoodPlace}/11-Display-Gal-Fil-2D*** directory.

If it isn't working, you can go directly to solutions.

4.2 Distance map

From the filaments, a distance map estimator is computed (cf Fig. 2). The estimated quantity represents the shortest distance from any point of the domain to the filamentary pattern. The details of the computation of this estimator are not given here.

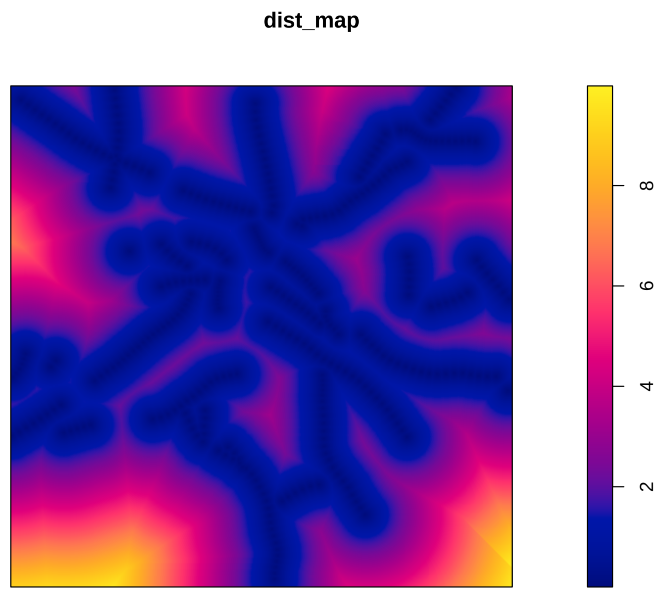


Figure 2: Distance map from filaments

To visualize the superimposition of galaxies on distance map and filaments on distance map, you can run the superimpose_gal_distmap_rstudio.R and superimpose_fil_distmap_rstudio.R program in the ***{GoodPlace}/12-Surperimpose-Gal-Fil-Distmap*** directory.

If it isn't working, you can go directly to solutions.

4.3 Exploratory analysis using summary statistics

4.3.1 Choosing the null model

One of the first objectives of the exploratory analysis of a point pattern using summary statistics is to grasp insight regarding the possible model to be fitted to the observed data.

This task is not so trivial as it may appear. The first advantage of it is that it requires neither model assumptions nor model simulations in order to estimate the chosen summary statistics. Still, as it can be seen later, the use of the exploratory analysis may require at least good estimates if not real knowledge of the intensity function of the process. It may also require the simulation from a null model, if one desires more robust answers to the formulated questions.

It is important, when fitting models to an observed point pattern, to understand its general characteristics: purely random, clustered or repulsive. The purely random pattern is considered to be the outcome of a Poisson point process. The clustering or the repulsion behaviour are seen as deviations from a Poisson process.

Therefore the inhomogeneous Poisson proces is the very first model to be considered as the null model.

Knowing that the number of galaxies in the data file is 460. Go to the ***{GoodPlace}/13-Poisson-Inhomogen-Astro***. After compilation in ***C++/SRC*** whith make, execute in directory ***C++/EXEC*** the model_simul.bat program.

To visualize the result pattern, go to ***{GoodPlace}/13-Poisson-Inhomogen-Astro/R*** and execute the plotPattern-Stat_rstudio.R.

If it isn't working, you can go directly to solutions.

4.3.2 Choosing the summary statistic

The choice of the summary statistics to be used is important. The empty space function, the nearest neighbour distribution, the $K-$ function estimates are cumulative quantities. In this case, it is difficult to assess at what level the deviation from the Poisson process occurs.

For these reasons, the pair-correlation function can be preferred. The estimator of the pair-correlation function does not exhibit the previously mentioned cummulative effect.

Go to the directory ***14-Inhomogeneous-Pair-Correlation-Function***

In this directory there is a **R** script that can be used in Rstudio to compute the inhomogeneous PCF of Poisson with the previous distmap :

compute_2D_PCF_inhomogen_rstudio.R

If it isn't working, you can go directly to solutions.

For values r lower than 2 the pair correlation function is superior to 1 hence indicating clustering with respect to the considered inhomogeneous Poisson process. For values r greater than 2 the inhomogeneous pair correlation function

tends to behave similarly as the one of the considered inhomogeneous Poisson process.
In the light of this analysis, in the following, we propose to fit to the data an inhomogeneous marked area-interaction process. It is also important to take into account that the implemented envelope test is local, that is a multiple statistical test implemented for each considered r value. The authors Myllymäki et al. 2017 developed a global test allowing to assess a $p-$value for the whole range of r values.

4.3.3 PCF based envelope test

The estimates of the summary statistics are random variables. Hence, their values computed from a data set are a realisation of these random variables. When comparing the realisation of a random variable to the possible outcome from the model considered under H_0, it is more robust to simulate envelopes of the statistics value under the null hypothesis. This is the role of the envelope tests Baddeley, Rubak, and Turner 2016; Lieshout 2000; Møller and Waagepetersen 2004; Stoica 2025ere.

Go to the directory ***15-PCF-Envelope-Poisson***

cd {GoodPlace}/15-PCF-Envelope-Poisson

In this directory there is a **R** scripts that can be used in Rstudio to compute envelope test:

compute_2D_PCF_inhomogen_rstudio.R

In this envelope test, 50 simulations are used to compute the extremal value of the pcf for different values of r.

If it isn't working, you can go directly to solutions.

Clearly, the hypothesis that the data may be the outcome of an inhomogeneous Poisson point process is rejected for values r smaller than 2. For the chosen intensity function, for values r smaller than 2, the data exhibit clustering with respect to the considered inhomogeneous Poisson process.

4.4 Choosing an alternative model to be fitted

The *Inhomogeneous Area-Interaction process* model was presented in section 3.4.

We now develop the step-by-step procedure to compute the envelope in this case.

- computation of sufficient statistics for the proposed model

- estimation of the parameters values of the model by ABC Shadow
- simulation of n realisations with the estimated parameters
- using the envelopes of the simulations, control the quality of the inference

To sample the posterior density of this model using the ABCShadow method, we need to compute the studied galaxies pattern's sufficient statistics.

Following the results of the exploratory analysis, the range parameter was fixed to $r = 0.5$.

The programs needed to do this and the inference can be found in ***16-Poisson-Inhomogen-MarkedAreaInt-Astro*** :

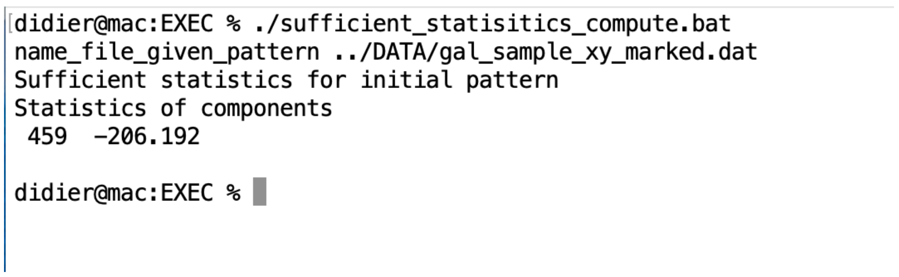
cd {GoodPlace}/16-Poisson-Inhomogen-MarkedAreaInt-Astro/C++/EXEC

In this directory after compilation, there is executable program to compute the sufficient statistics and estimate the parameters from them.

sufficient_statisitics_compute.bat

After compilation and execution, we obtain the following sufficient statistics:

$$n(x) = 459, \quad a(x) = -206.192$$



```
didier@mac:EXEC % ./sufficient_statisitics_compute.bat
name_file_given_pattern ../DATA/gal_sample_xy_marked.dat
Sufficient statistics for initial pattern
Statistics of components
 459  -206.192

didier@mac:EXEC %
```

Figure 3: 16-Poisson-Inhomogen-MarkedAreaInt-Astro sufficient statistics

These parameters and $r = 0.1$ are written in the parameter file *p_abc_poisson_inhomogen_markedareaint.txt* of the *ABC shadow* located in the ***PARAMS*** directory.

Next, we run the estimation program using the *ABCShadow* method in ***EXEC*** directory

abc_simul.bat

Calculation times are very long. In ***R*** directory, displaySampling_rstudio.R displays the result.

If it isn't working, you can go directly to solutions.

With the results file *theta_abc_poisson_inhomogen_markedareaint-5-10e5.txt* in ***RESULTS*** directory corresponding to the 5.10^5 iterations, we estimate the parameters to the following values:

$$\beta = 7.7436, \quad \gamma = 3.8767$$

To get a rough idea of the quality of the result, you can use the R program plotPattern_rstudio.R, which visualizes the last pattern generated using ABC. Figure 4 shows the last pattern generated after 5.10^5 iterations of ABC.

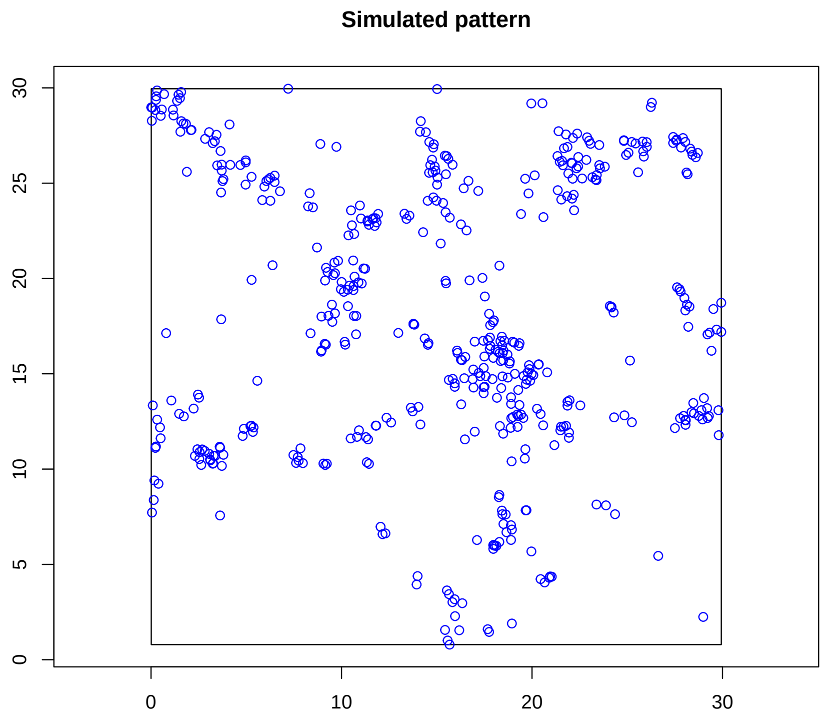


Figure 4: Last pattern generated after 5.10^5 iterations of ABC

We can see that compared with inhomogeneous Poisson process (see above) there is clustering of points compared to the inhomogeneous Poisson case.

4.4.1 Model verification

Once a model is fitted to the observed data, there are two more steps to go. First, the estimated values of the parameters should be verified. One wishes to see how far these values are from the values of the model parameters that would really produce the data. Since our estimation is Monte Carlo based, one wishes to check after how many simulations or iterations of the algorithm the theoretical precision is reached.

Second, the previous verification is a non-default test. A more insightful information is to really check and validate the chosen model. Residual theory can be adapted to rigorously validate point process choices Baddeley, Rubak, and Turner 2016. A complementary strategy is to perform envelope tests using the estimated parameters in order to see if the simulated patterns using these parameters fit the characteristics of the observations outlined by the envelope test statistic. This test requires the simulation of the null hypothesis model. In our case, the null hypothesis is the inhomogeneous are-interaction process with the estimated parameters. The chosen summary statistics for the test was the pair correlation function.

The necessary material is available in the directory ***17-PCF-Envelope-AreaInt***

```
cd {GoodPlace}/17-PCF-Envelope-AreaInt
```

After compilation, running the program model_simul.bat generates in the ***{GoodPlace}/17-PCF-Envelope-AreaInt/Simulated*** directory 50 patterns generated with the *Poisson inhomogen / marked areaint* model.

From these simulations, the compute-envelope_rstudio.R program in the ***R*** directory calculates and displays the envelope test.

Let's look at the envelope test result for inhomogeneous Area-Interaction process.

If it isn't working, you can go directly to solutions.

We can see from this envelope that the pcf function of the given galaxy pattern is included in the envelope. The hypothesis of a marked areaint process is therefore not invalidated.

References

Baddeley, A., E. Rubak, and R. Turner (2016). *Spatial Point Patterns. Methodology and Applications with R.* Chapman and Hall/CRC.

Gillot, Nathan et al. (Sept. 2024). "Spatial point process modelling and Bayesian inference for large data sets". In: *RING Meeting*. École nationale supérieure de géologie (ENSG) Nancy. Nancy, France. URL: https://hal.science/hal-04645186.

Hurtado-Gil, L. et al. (2021). "Morpho-statistical characterisation of the spatial galaxy distribution through Gibbs point processes". In: *Monthly Notices of the Royal Astronomical Society* 507.2, pp. 1710–1722.

Lieshout, M. N. M. van (2000). *Markov Point Processes and their Applications.* Imperial College Press, London.

Lieshout, M. N. M. van and R. S. Stoica (2006). "Perfect simulation for marked point processes". In: *Computational Statistics and Data Analysis* 51, pp. 679–698.

Møller, J. and R. P. Waagepetersen (2004). *Statistical inference and simulation for spatial point processes.* Chapman and Hall/CRC, Boca Raton.

Myllymäki, M. et al. (2017). "Global envelope tests for spatial processes". In: *Journal of the Royal Statistical Society, Series B* 79.2, pp. 381–404.

Stoica, R. S. (2025). *Random patterns and structures in spatial data.* Chapman and Hall.

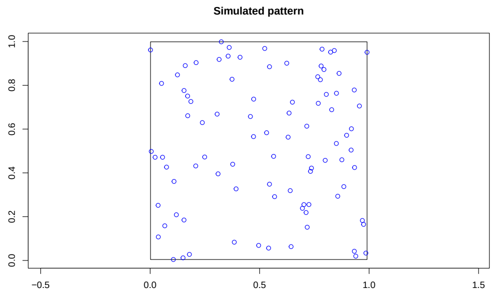
Tempel, E., R. Kipper, et al. (2014). "Galaxy filaments as pearl necklaces ?" In: *Astronomy and Astrophysics* 572.A8, pp. 1–8.

Tempel, E., R. S. Stoica, et al. (2014). "Detecting filamentary pattern in the cosmic web: a catalogue of filaments for the SDSS". In: *Monthly Notices of the Royal Astronomical Society* 438.4, pp. 3465–3482.

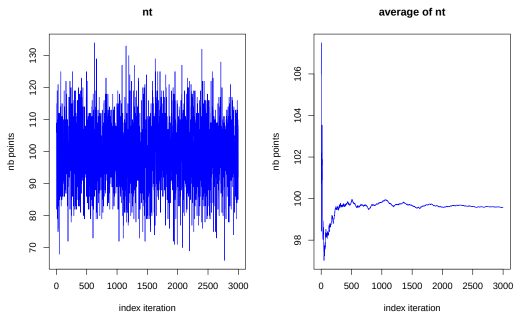
5 Appendix: solutions and results

5.1 Proposed questions simulation of Homogeneous Poisson process

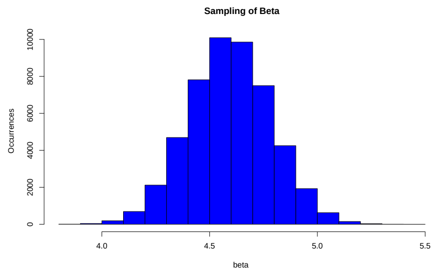
$\beta = 4.6$ nt.average $= 99.58$ xd mean $= 4.59$



(a) Sample pattern



(b) Statistics

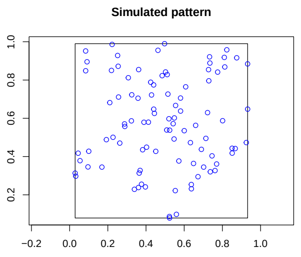


(c) Parameters sampling

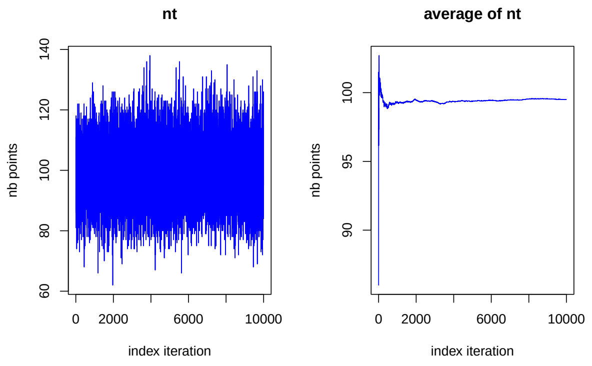
back to questions.

5.2 Proposed questions Inhomogeneous Poisson process

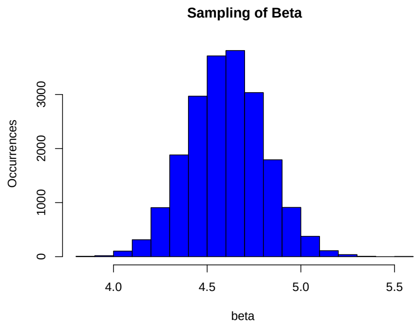
$log(\beta) = 4.60$ nt.average = 99.5 xd mean = 4.60



(a) Sample pattern



(b) Statistics

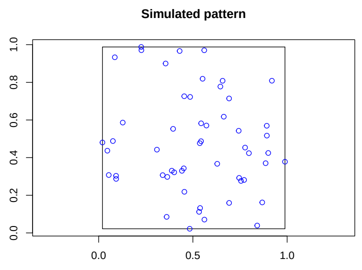


(c) Parameters sampling

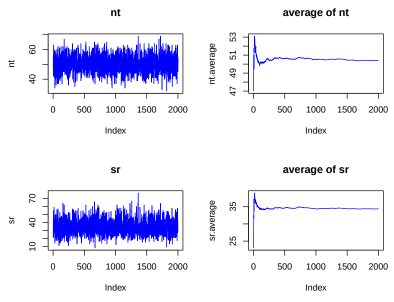
back to questions.

5.3 Proposed questions Inhomogeneous Poisson / Marked Strauss process

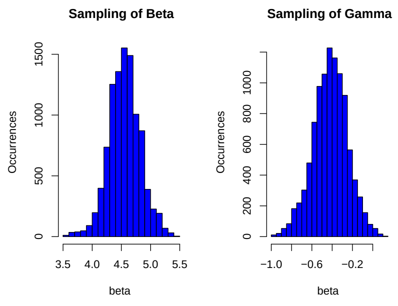
$$log(\beta) = 4.60 \qquad log(\gamma) = -0.4 \qquad r = 0.05$$
$$nt = 50.41 \qquad sr = 34.3755$$
$$xd \text{ mean} = 4.55 \qquad yd \text{ mean} = -0.41$$



(a) Sample pattern



(b) Statistics



(c) Parameters sampling

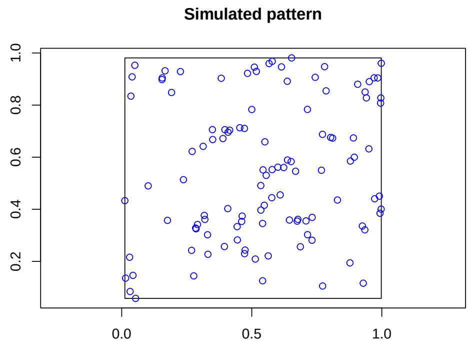
back to questions.

5.4 Proposed questions Inhomogeneous Poisson / Marked AreaInt process

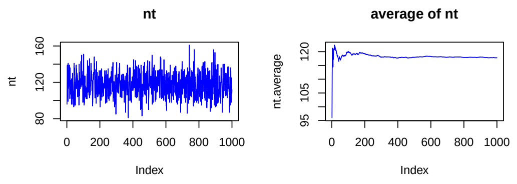
$log(\beta) = 5.4$ $\quad log(\gamma) = 1.0$ $\quad r = 0.03$

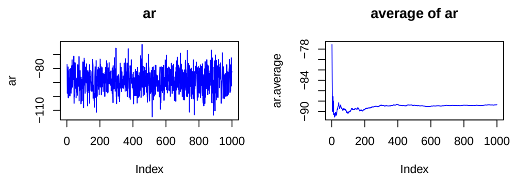
$nt = 117.76$ $\quad ar = -88.69$

$log(\beta_{ABC})$ mean $= 5.34$ $\quad log(\gamma_{ABC})$ mean $= 0.93$

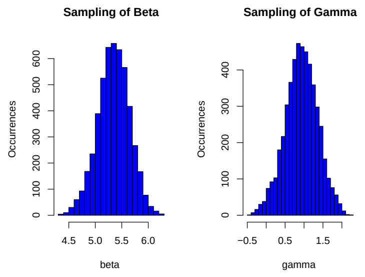


(a) Sample pattern



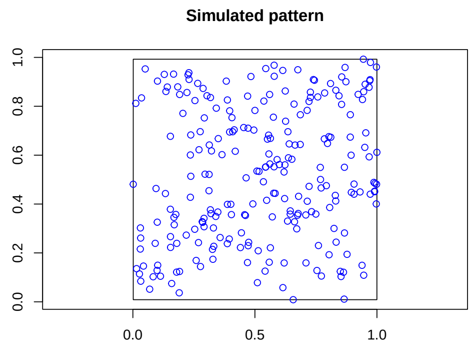


(b) Statistics

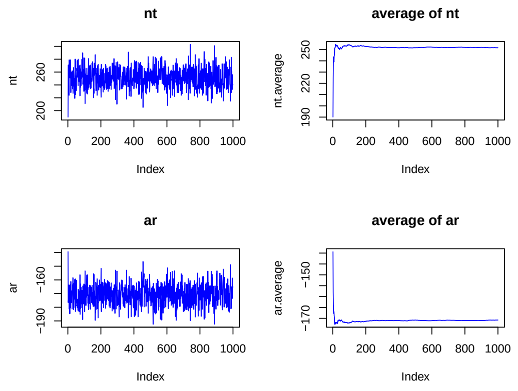


(c) Parameters sampling

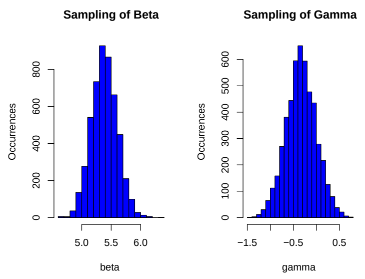
$$log(\beta) = 5.4 \qquad log(\gamma) = -0.3 \qquad r = 0.03$$
$$nt = 251.62 \qquad ar = -170.78$$
$$log(\beta_{ABC}) \text{ mean} = 5.38 \qquad log(\gamma_{ABC}) \text{ mean} = -0.33$$



(a) Sample pattern



(b) Statistics



(c) Parameters sampling

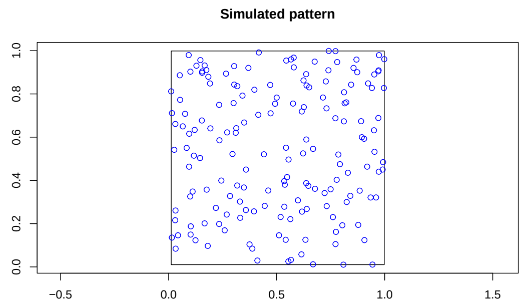
back to questions.

5.5 Proposed questions Inhomogeneous Poisson / Marked Strauss / Marked AreaInt process

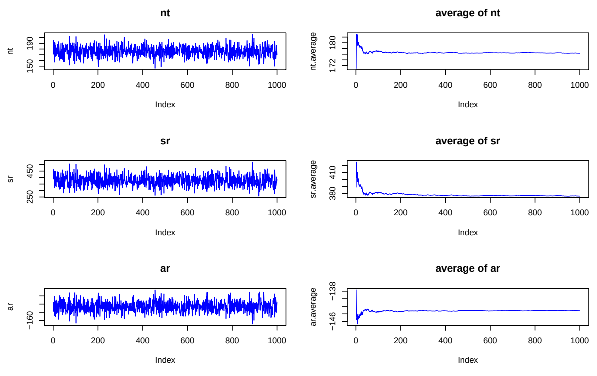
$\beta = 5.6$ $\gamma = -0.20$ $r_s = 0.05$ $\delta = -0.8$ $r_a = 0.03$

$nt = 176.329$ $sr = 376.277$ $ar = -143.04$

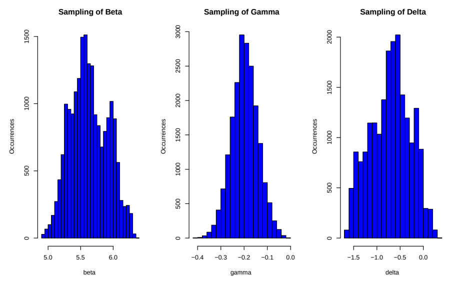
xd mean $= 5.63$ yd mean $= -0.19$ zd mean $= -0.70$



(a) Sample pattern



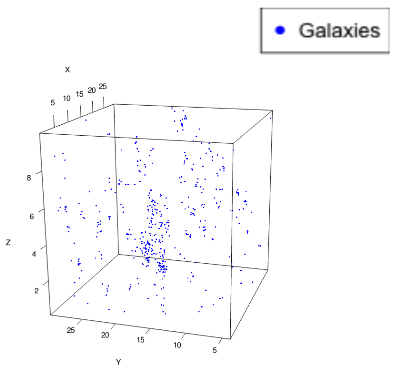
(b) Statistics



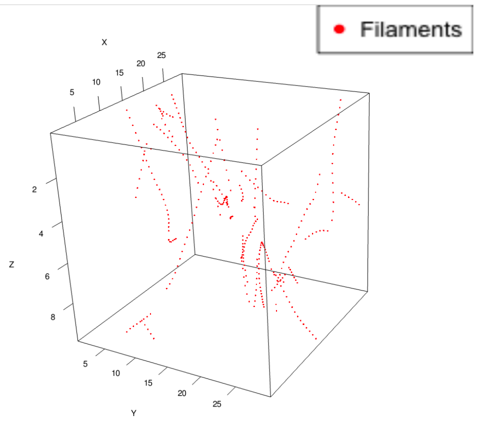
(c) Parameters sampling

back to questions.

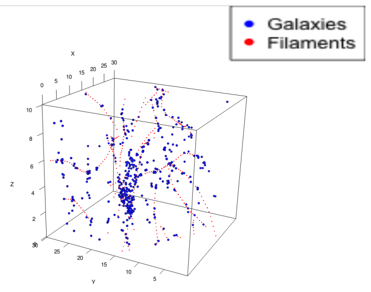
5.6 Proposed questions 3D data



(a) Initiales 3D positions of galaxies



(b) Initiales 3D positions of filaments



(c) Surperimposition of galaxies and filaments

back to questions.

5.7 Proposed questions superimposition 2D galaxies / filaments

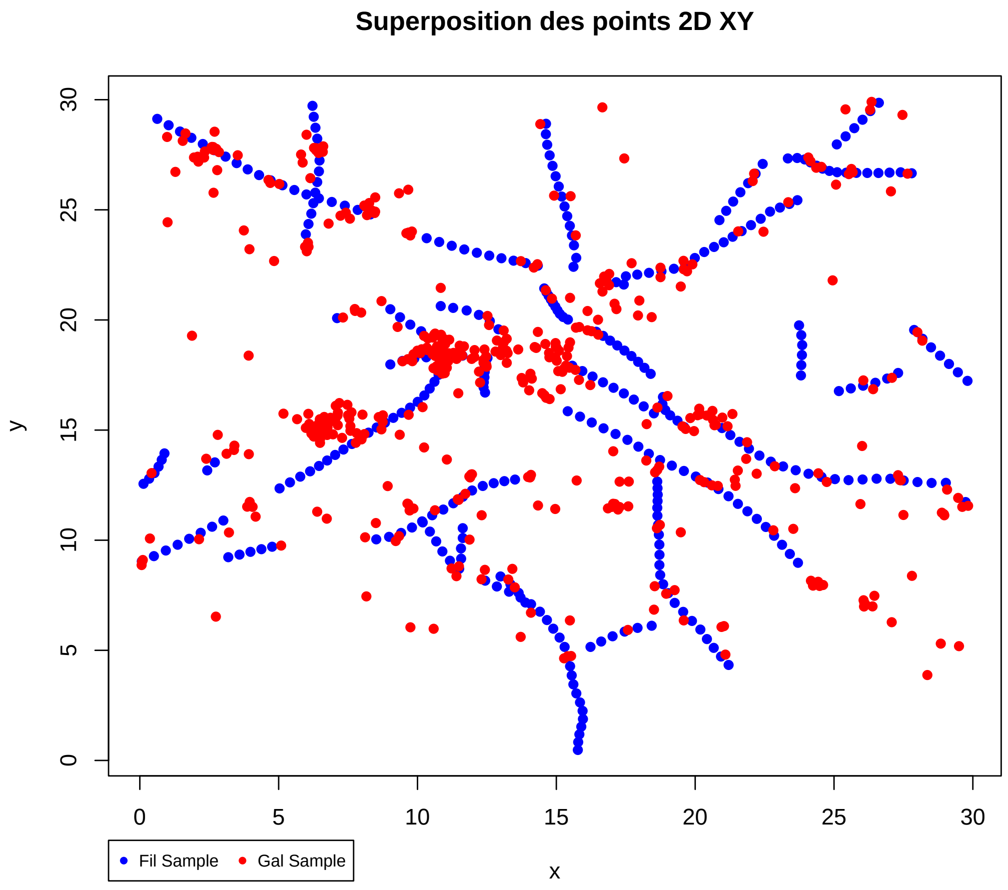
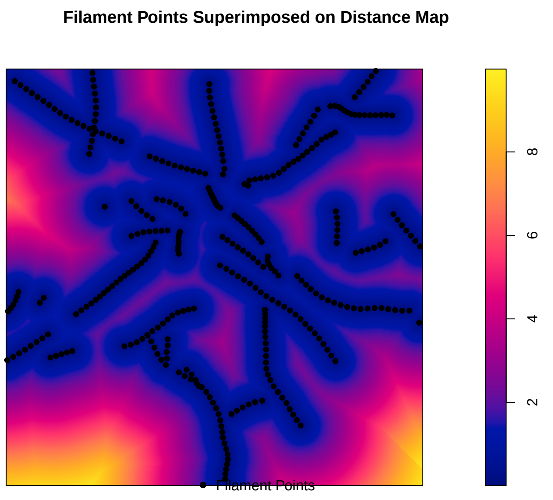


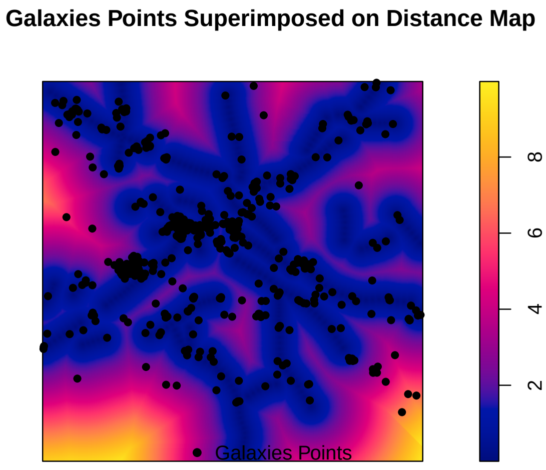
Figure 12: Superimposition of 2D galaxies and filaments coordinates

back to questions.

5.8 Proposed questions distance map



(a) Superimposition of filaments and distance map



(b) Superimposition of galaxies and distance map

back to questions.

5.9 Proposed questions realization of inhomogeneous Poisson process

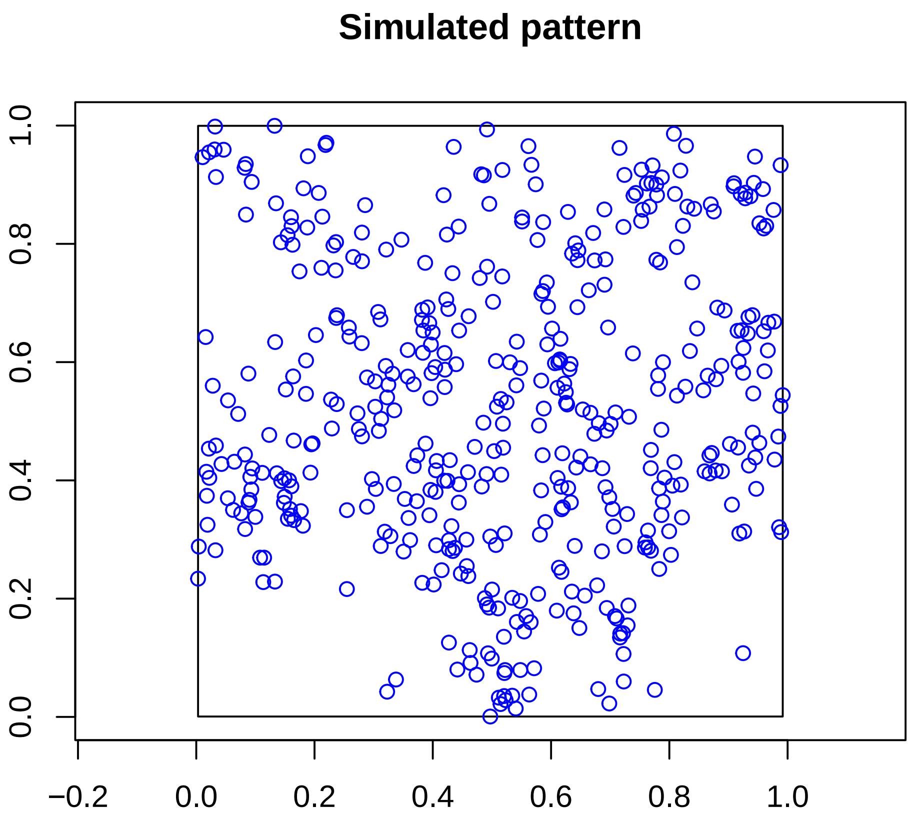


Figure 14: Realization of inhomogeneous Poisson process

back to questions.

5.10 Proposed questions inhomogeneous Pair Correlation Function

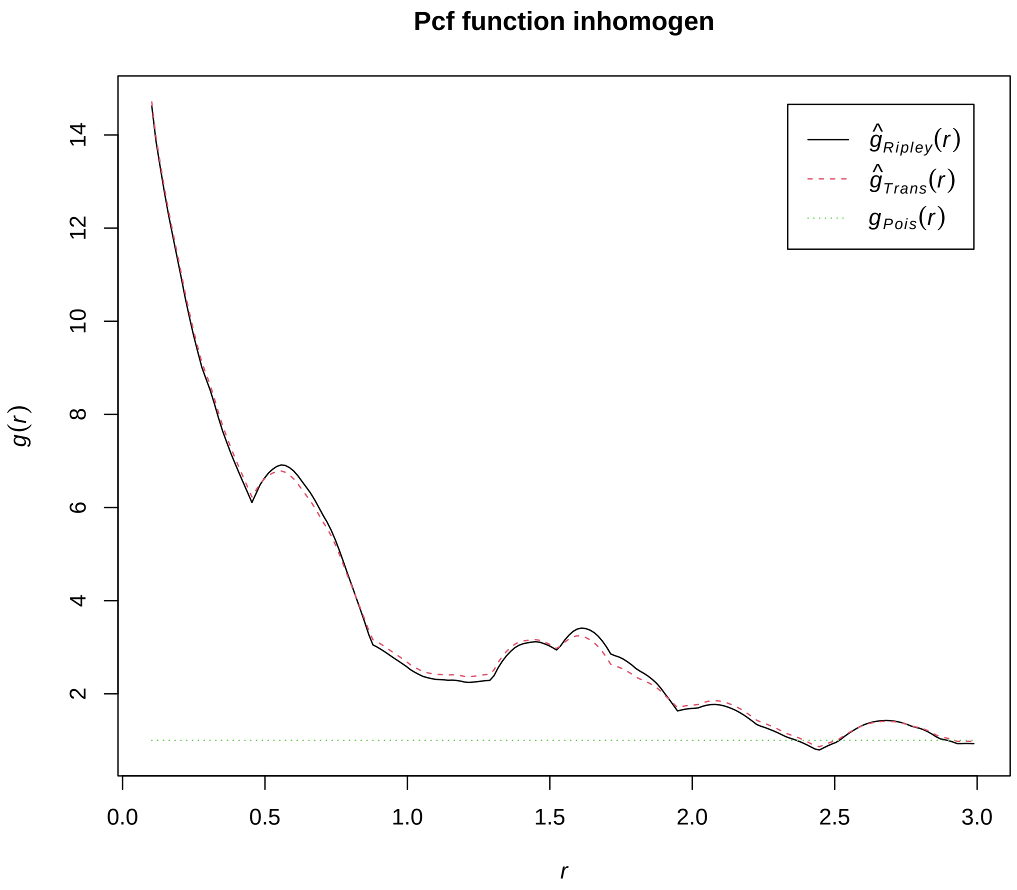


Figure 15: Inhomogeneous Pair Correlation Function

back to questions.

5.11 Proposed questions PCF envelope inhomogeneous Poisson

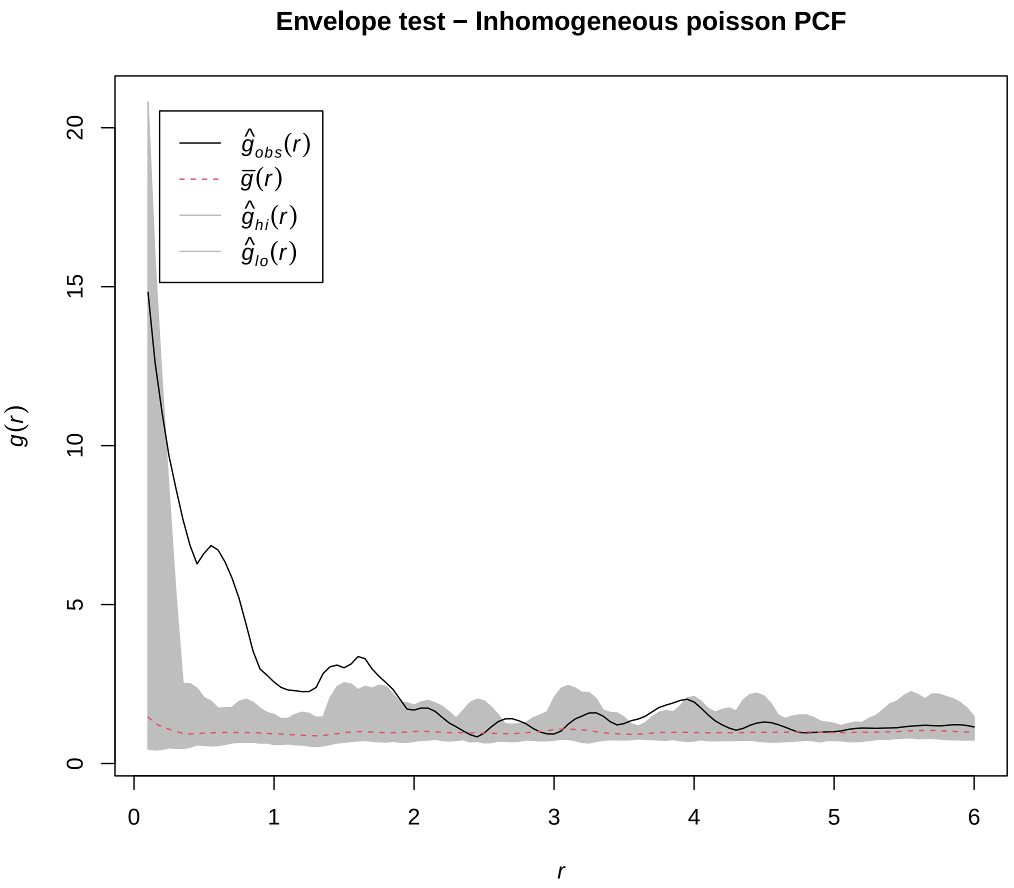


Figure 16: PCF envelope inhomogeneous Poisson

back to questions.

5.12 Proposed questions Astronomical data sampling-theta

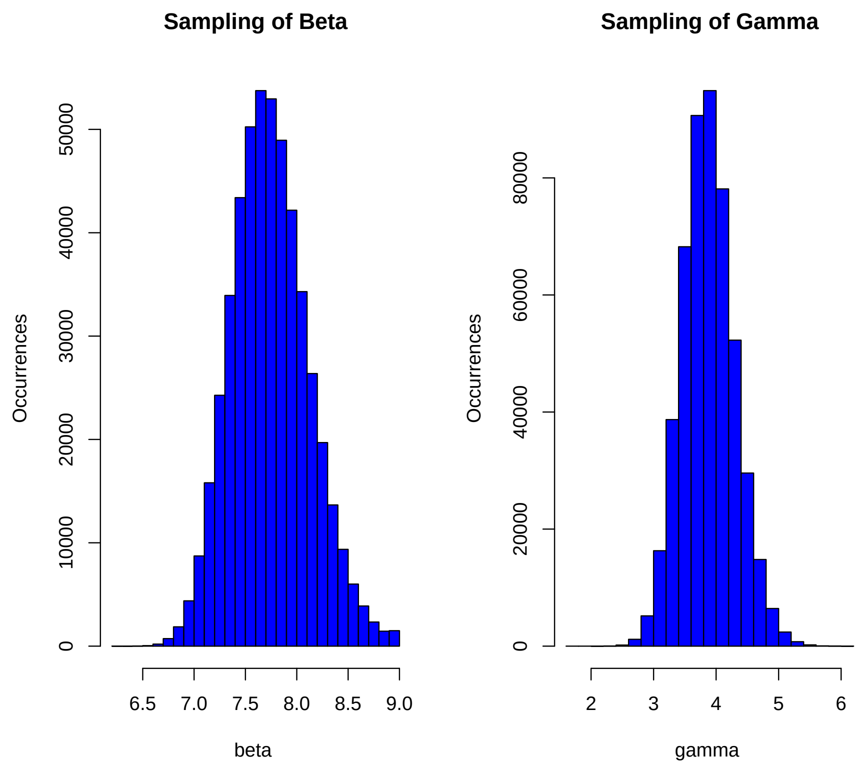


Figure 17: Astronomical data sampling-theta by ABC shadow

back to questions.

5.13 Proposed questions PCF envelope AreaInt for given pattern galaxies

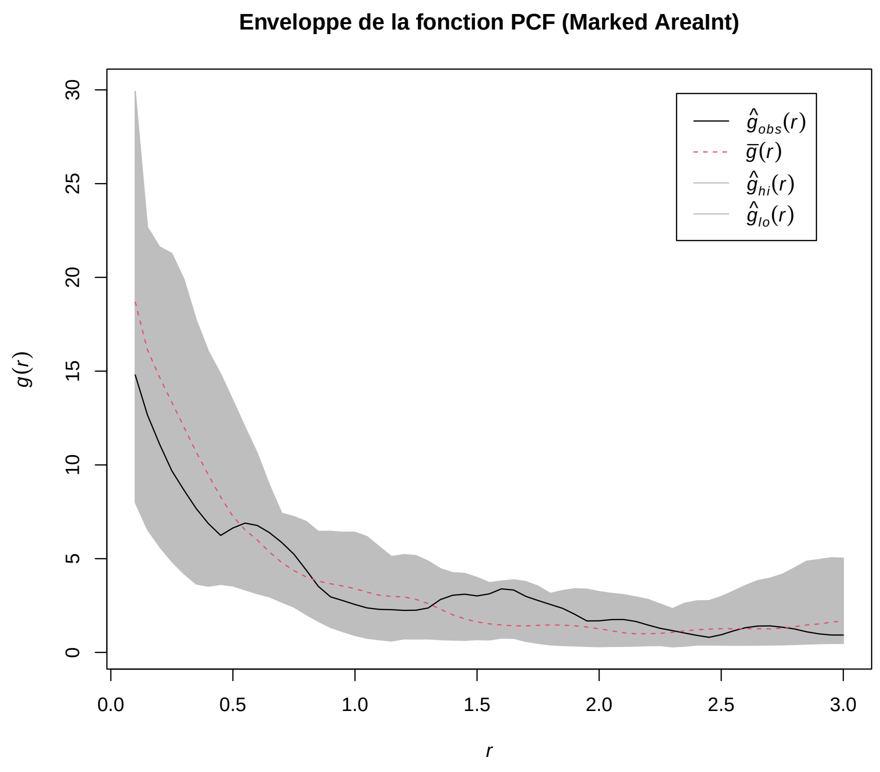


Figure 18: PCF envelope AreaInt for given pattern galaxies

back to questions.