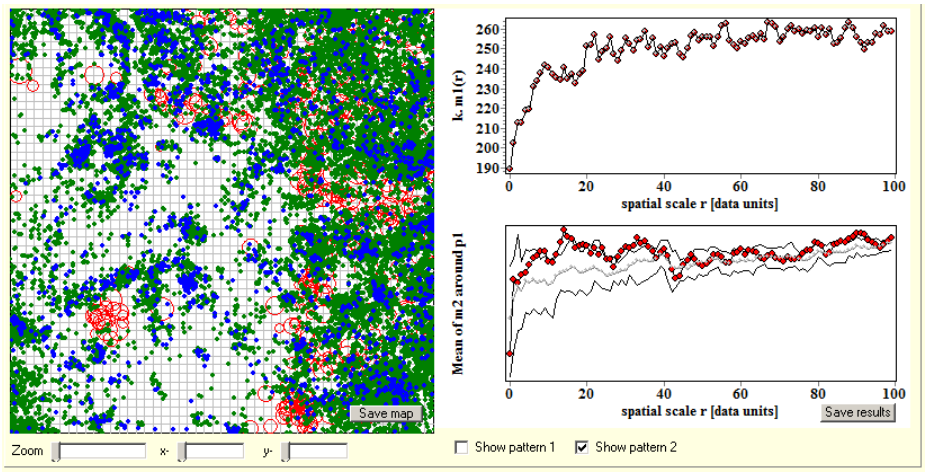
User Manual for the *Programita* software



Version
January 2014

written by
Thorsten Wiegand
Department of Ecological Modelling,
Helmholtz Centre for Environmental Research - UFZ,
Permoserstr. 15,
04318 Leipzig, Germany,

Tel.: (**49) 341 235 1714
thorsten.wiegand@ufz.de

Contents

1 General information

Programita is a comprehensive software package for conducting spatial point pattern analysis in ecology. I tailored *Programita* to accommodate the needs of "real world" applications in ecology and developed the different modules in response to my own research questions and to requests of colleagues and students who have approached me with their specific research problems in mind. Ten years after its launch in the 2004 Oikos Mini review (Wiegand and Moloney 2004), *Programita* has considerably grown and now contains a variety of statistical methods for most of the **point pattern data types** which are relevant in ecological applications, including

- **univariate patterns** (i.e., one type of points)
- **bivariate patterns** (i.e., two types of points such as two species of trees)
- **multivariate patterns** (i.e., several types of points such as a forest tree community)
- **multivariate patterns** with a matrix of pairwise (e.g., functional or phylogenetic) dissimilarities between types of points
- **qualitatively marked patterns** (i.e., one type of point with a qualitative mark such as surviving vs. dead)
- **quantitatively marked patterns** (a pattern augmented with quantitative marks such as size)
- **objects with finite size and real shape** (cases for which the point approximation does not hold)

Programita offers for each of these data types the most appropriate **summary statistics**:

- univariate patterns [pair correlation function $g(r)$, O-ring statistic $O(r)$, L-function $L(r)$, the $K2$ function $K2(r)$, the distribution functions $D^k(r)$ of the distances to the kth nearest neighbor, the spherical contact distribution $H_s(r)$, the mean distance to the kth neighbor $nn(k)$, and inhomogeneous g- and L-functions]
- bivariate patterns [pair correlation function $g_{12}(r)$, O-ring statistic $O_{12}(r)$, L-function $L_{12}(r)$, the $K2$ function $K2_{12}(r)$, the distribution functions of the distances to the kth nearest neighbor $D^k{}_{12}(r)$, the mean distance to the kth neighbor $nn_{12}(k)$]
- multivariate patterns [spatially explicit Simpson index $\beta(r)$, community mean ISAR function $\overline{ISAR}(r)$, species-area relationship $SAR(r)$]
- multivariate patterns with dissimilarity matrix [phylogenetic Simpson index $\beta_{phy}(r)$, phylogenetic mark correlation function $k_d(r)$, mean phylogenetic ISAR function $\overline{PISAR}(r)$, mean rISAR function]
- qualitatively marked patterns [mark connection functions $p_{ij}(r)$ and various test functions based on pair correlation or K-functions]
- quantitatively marked patterns [various normalized and non-normalized mark correlation functions $k_t(r)$]
- objects with finite size and real shape [$g(r)$, $L(r)$, $g_{12}(r)$, $L_{12}(r)$]

and the most important **null models** for ecological applications, which often allow for consideration of a spatially varying intensity function $\lambda(\mathbf{x})$, and a variety of **point process models** describing clustering.

Programita allows you to conduct Monte Carlos simulations of the null models, fit cluster point processes to the data, and to generate stochastic realizations of the fitted point processes to determine **simulation envelopes** and to conduct a **Goodness-of-Fit (GoF) test**.

1.1.1 Terms of use

I am not in the commercial software business, but recognize the need for a tool like *Programita* for scientists to assist in research on spatial point pattern analysis. I produced the *Programita* software in my spare time to foster the analysis of point patterns in ecology and to provide ecologists with a tool that contains null models and procedures not supported by most statistical packages, but which are essential for a thorough analysis of point-patterns. I have done my best to provide in the documentation complete, step-by-step instructions for the variety of analysis you can conduct with *Programita*, but the user is responsible for acquiring the necessary background knowledge to appropriately use the software.

The *Programita* software may be downloaded and used free of charge for purposes of scientific research and teaching. However, please do not distribute the link, *Programita* or the manual. Any commercial application of the program requires the previous permission by the author. *Programita* is provided "as is" without warranty of any kind. In no event will the authors be liable for any damages, including lost profits, lost savings, or other incidental or consequential damages arising from the use of or the inability to use this software.

Publications using *Programita* must acknowledge use of *Programita* and include the following citations:

Wiegand T., and K. A. Moloney 2004. Rings, circles and null-models for point pattern analysis in ecology. *Oikos* 104: 209-229.

Wiegand T., and K. A. Moloney 2014. A handbook of spatial point pattern analysis in ecology. Chapman and Hall/CRC press, Boca Raton, FL.

if analysis of objects with finite size and real shape is used add:

Wiegand, T., Kissling, W.D., Cipriotti, P.A., and Aguiar, M.R. 2006. Extending point pattern analysis to objects of finite size and irregular shape. *Journal of Ecology* 94: 825-837)

if cluster processes are used add:

Wiegand, T, A. Huth., and I. Martínez. 2009. Recruitment in tropical tree species: revealing complex spatial patterns. *The American Naturalist* 174: E106 - E140)

if random labeling analysis is used add:

Jacquemyn, H., P. Endels, O. Honnay, and T. Wiegand. 2010. Evaluating management interventions in small populations of a perennial herb Primula vulgaris using spatio-temporal analyses of point patterns. *Journal of Applied Ecology* 47: 431–440

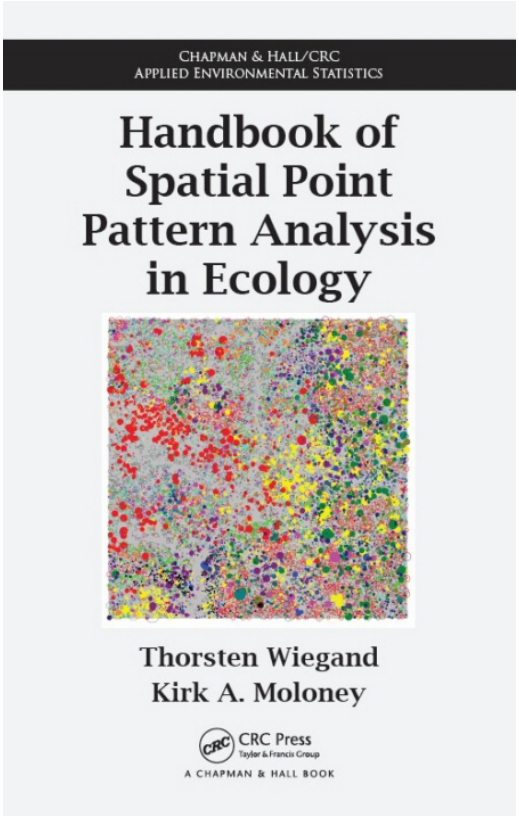
if phylogenetic mark correlation functions are used

Shen, G., T. Wiegand, X. Mi, and F. He in 2014. Quantifying spatial phylogenetic structures of fully mapped plant communities. *Methods in Ecology and Evolution* 4: 1132-1141.

1.1.2 Handbook of Spatial Point Pattern Analysis in Ecology

Understand How to Analyze and Interpret Information in Ecological Point Patterns

The methods underlying *Programita* and many examples executed with *Programita* can be found in our recent book published by Chapman and Hall. This manual will therefore not contain detailed explanations of the methods, but I will refer instead to the respective sections in the book. **I warmly recommend you to buy the book to fully benefit from the possibilities offered by *Programita*.**



Although a broad array of statistical methods for analyzing spatial point patterns have been available for several decades, they haven't been extensively applied in an ecological context. Addressing this gap, Handbook of Spatial Point Pattern Analysis in Ecology shows how the techniques of point pattern analysis are useful for tackling ecological problems. Within an ecological framework, the book guides readers through a variety of methods for different data types and aids in the interpretation of the results obtained by point pattern analysis.

Ideal for empirical ecologists who want to avoid advanced theoretical literature, the book covers statistical techniques for analyzing and interpreting the information contained in ecological patterns. It presents methods used to extract information hidden in spatial point pattern data that may point to the underlying processes. The authors focus on point processes and null models that have proven their immediate utility for broad ecological applications, such as cluster processes.

Along with the techniques, the handbook provides a comprehensive selection of real-world examples. Most of the examples are analyzed using *Programita*, a continuously updated software package based on the authors' many years of teaching and collaborative research in ecological point pattern analysis. *Programita* is tailored to meet the needs of real-world applications in ecology. The software and a manual are available online.

Features

- Focuses on the application of spatial point pattern analysis in an ecological context
- Helps ecologists unfamiliar with advanced statistics select the proper analysis method
- Emphasizes the formulation of appropriate null models and point processes for describing the features of point patterns and testing ecological hypotheses of spatial dependence
- Provides the *Programita* software package on the first author's website, enabling readers to perform analyses with their own point pattern data
- Includes a collection of real-world examples
- Offers suggestions on how to use the book for teaching graduate students

1.2 Before starting *Programita*

1.2.1 Hardware requirements

Programita is a free unsupported software, developed in Borland Delphi under a Windows 7 environment. *Programita* is also executable under older Windows versions such as Windows XP and can be used under a MacOs with help of programs like WinneBotter (http://winebottler.kronenberg.org) and Xquartz (http://xquartz.macosforge.org/landing/).

1.2.2 Installation

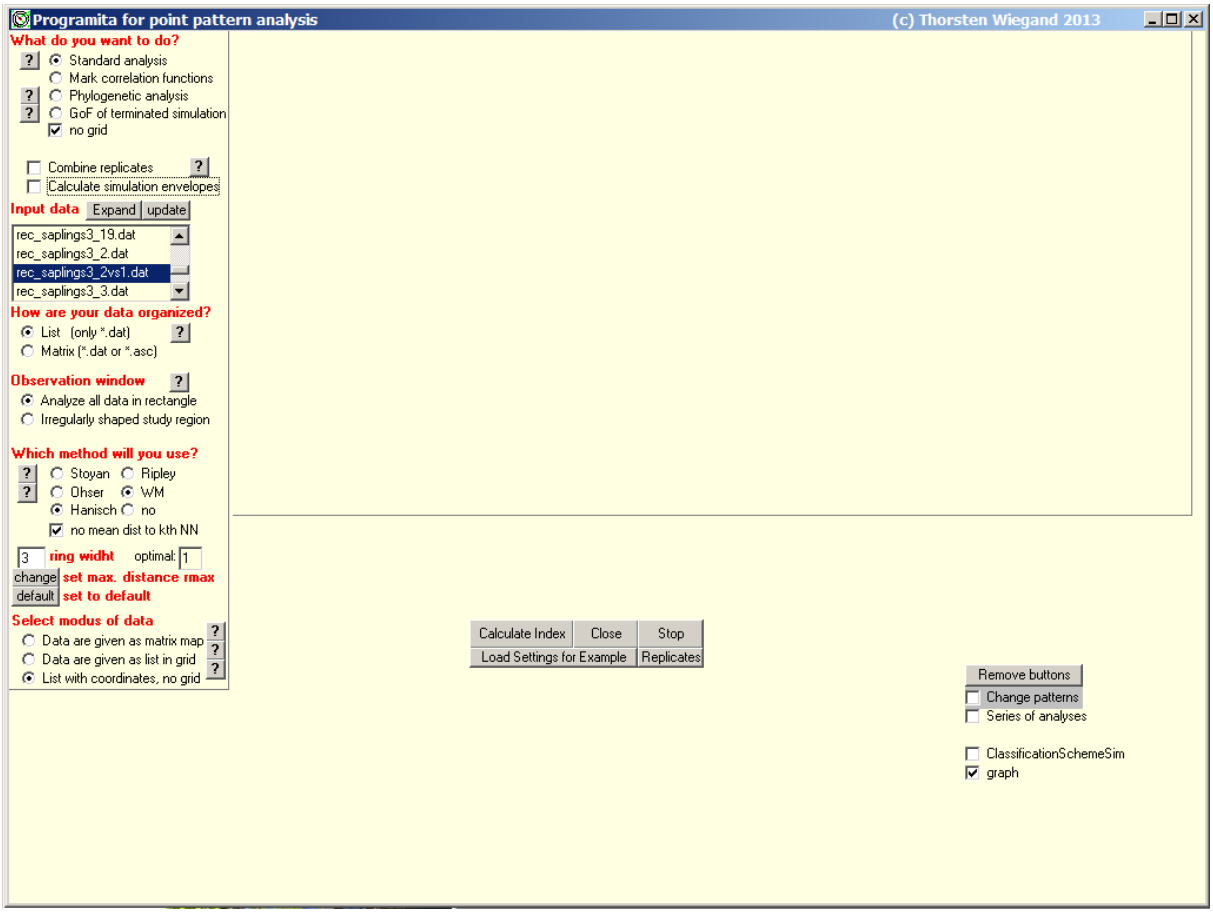
There is no setup procedure; installation of *Programita* requires only the extraction of all files from the zip file ProgamitaEnero2014.zip. Place the files into a directory of your choice; extracting the zip file will place all files into the sub-directory *Programita*. Note that you must place *Programita* into the same directory as the data input files; for simplicity *Programita* does not use a path variable. However, since *Programita* occupies little space you can place into each folder with data files for specific analysis a copy of *Programita*.

The zip-file contains the following files and file types:

*Programita*Enero2014.exe	• the executable of *Programita,* version of January 2014
*.dat files	• data files for uni- and bivariate analyses and temporary files
*.mcf files	• data files for mark correlation analysis
*.phy files	• data files for multivariate analysis using a dissimilarity matrix, e.g., analysis using phylogenetic mark correlation functions
*.asc files	• data file in ArcView raster format for grid based analyses of objects with finite size and real shape
	•
*.shp files	• text file with coordinates to encircle an irregularly shaped observation window
*.txt files	• data files with the dissimilarity matrix for multivariate analysis
*.res files	• files that contain the results of an analysis and all settings. Can be used to load the settings of an analysis and to redo an analysis
*.rep files	• data files to show results of previous analyses and for combining replicates
*.int files	• plug-in files with the intensity function $\lambda(\mathbf{x})$ for example used in the heterogeneous Poisson process
*.env files	• temporary files containing the observed summary statistics and that of the simulations of the null model; required for the GoF test

1.2.3 Screen size

Programita occupies a screen of 1024 × 768 pixels and after executing *Programita* the screen shown below should appear. However, sometimes buttons or windows within *Programita* are truncated and the text does not fit. To avoid this problem check the default letter size in the settings of your computer. Your computer may scale the letters but not the window sizes and as a consequence, the windows appear too small.



2 Features of *Programita* common to all analysis modes

2.1 Load a settings file to redo an analysis

There is a convenient way to quickly start with *Programita* and to learn the settings. You can read a file (a *.res-file) that contains all setting of a previous analysis and redo this analysis. To do this, click button "**Load settings for Example**", select a results file (in the example "test.res"), and the small "**ok**" button. *Programita* now reads all settings from the file test.res and if you click "**Calculate Index**" *Programita* repeats the analysis. However, this works only if all data files are in the same folder.

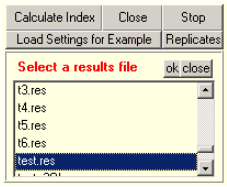


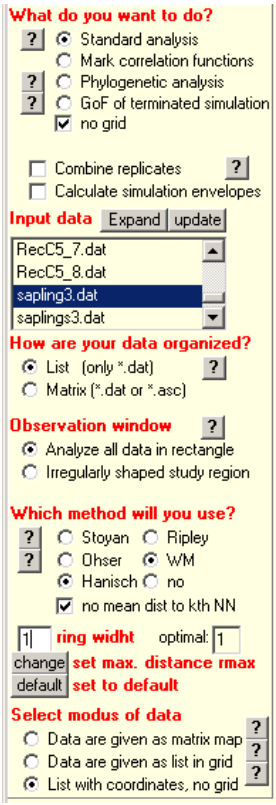
Figure 1.3.1.F1. Load an example settings file.

2.2 Overview over menus

Programita allows you to conduct analyses of a variety of data types. Basically, you can use:

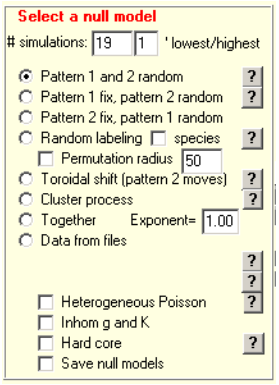
1. the standard analysis mode for uni- and bivariate analyses
2. the grid-based standard analysis mode
3. the analysis mode for mark correlation functions
4. the analysis mode for multivariate analysis using a dissimilarity matrix
5. the analysis model for object of finite size and real shape

You can select the data type in the window What do you want to do?



The settings are governed by two different sets of menus. First, one menu bar (at the left side of the screen) allows you to select the type of analysis and the estimators. For example, in the menu for the **standard analysis mode** shown on the left you can:

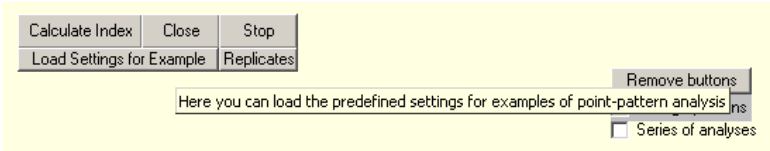
- select "**Standard analysis**" in What do you want to do? and "**no grid**" to reach the standard analysis mode. With "**no grid**" disabled you reach the grid-based standard analysis mode
- select the input data file (Input data file)
- provide information on the organization of your data (How are your data organized?) and (Select modus of data)
- analyze observation windows with irregular shape (Observation window)
- select the estimators of the summary statistics (Which method will you use?)
- define a ring width and a maximal distance *r* of analysis.



Second, if you check the checkbox "**Calculate simulation envelopes**" located above "Input data" a menu window that appears on the bottom right allows you to specify the settings of the null model. The radio buttons are the different null models available for this data type (e.g., "**Pattern 1 and 2 random**" uses the CSR null model for pattern 1 and pattern 2), and the checkboxes specify settings of the null model selected (e.g., with "**heterogeneous Poisson**" you can select an intensity file for the heterogeneous Poisson process). On the top you can select the number of simulations of the null model **(# simulations**) and the rule for the simulation envelope. Mostly you may use 199 simulations and the 5th lowest and highest values as simulation envelopes.

2.3 Hints

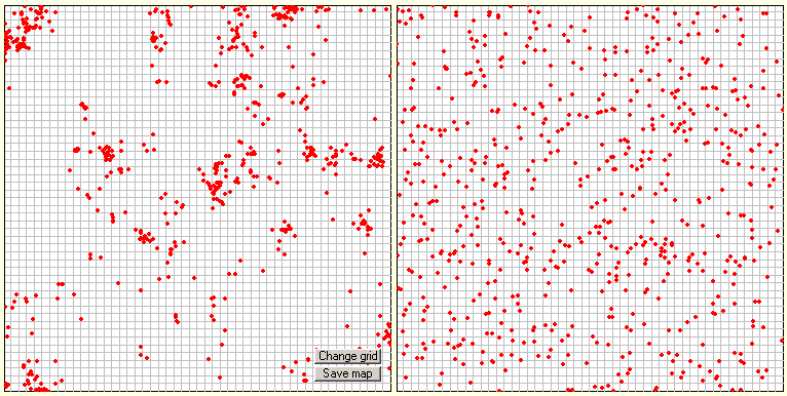
The help buttons are not yet implemented, but almost all important elements on the screen contains a hint. For example, if you move the cursor over the element "**Load settings for Example**", a small message box shows the hint that briefly explains the meaning of the element.



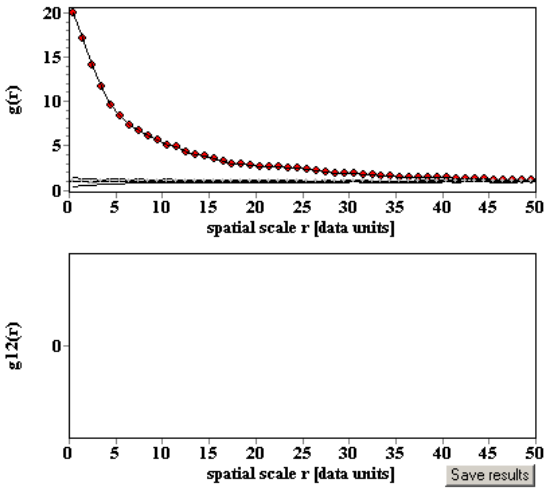
2.4 What happens on the screen?

After loading the settings file test.res as described above, *Programita* will automatically select all settings for the data and analysis mode and all settings for the null model. If you click "**Calculate Index**" *Programita* repeats the analysis. Two plots will appear additionally to the menus:

One plot shows the original point pattern being analyzed (left or top plot), and the other plot (on the right or bottom) shows the patterns of the Monte Carlo simulations of the null model used for constructing the simulation envelopes and the GoF test.

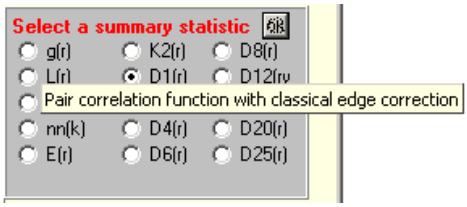


After the simulations of the null model are finished, the figure with the simulated patterns of the null model disappears, and instead a figure with the result of the analysis appears. The top (or left) figure shows generally the results of the univariate analysis and the bottom (right) figure shows the results of the bivariate analysis. (Exceptions are multivariate analysis using a dissimilarity matrix under the random labeling and trivariate random labeling mode). The data file in the example contained only one pattern (i.e., an univariate analysis), therefore no figure appears for the bivariate results



The observed summary statistics are indicated by red dots, the simulation envelopes by black lines and the expectation of the null model by a grey dotted line.

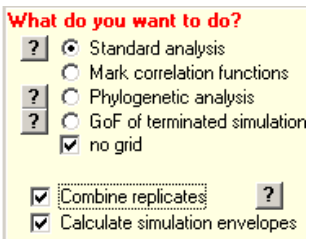
In the standard mode you can then view the results of various summary statistics, the hint explains the symbols.



2.5 Save the results of the analysis

To save the results of the analysis, press the button "**Save results**" that appears below the graph with the results of the bivariate analysis, and insert a name for the result file. The results file will be saved as ASCII-file name.res in the same directory where the *.exe file of *Programita* is located.

If the setting "**Combine replicates**" was enabled, *Programita* saves additionally the file **name.rep** that contain the information to recreate the results graph. Additionally, the *.rep files contain all information to combine the results of different replicate analysis. However, in the standard mode you need to save the results for each summary statistic separately.



The *.res results file (see below) contains the settings of this analysis and the results of the univariate and the bivariate point pattern analysis. The results file name.res can be used to load the setting and to repeat the analysis. More importantly you can copy-paste the second part of the *.res file into a graphics program to produce the figures.

```
Pointpattern analysis of file c:\Programita\saplings3.dat
Method Wiegand-Moloney (ring) with  19 replicates for simulation envelopes, ring width = 3   1 th lowest and highest values of 19 simulations
Test Model= 12random
the null model assumed homogeneous pattern(s)
Analysis modus= points        gridless    WM    NN Hanisch
several points per cell allowed
All cells within the rectangle were considered for calculating the indices
number points of pattern 1 =  626
number points of pattern 2 =    0
the rectangular area contains 500*500 = 250000   cells  (= dim1*dim2)
 x-grid-size=  500 y-grid-size=  500 cell-size =  1.0000 units. rmax=     50,  max distance for NN functions:    354
```

Scale	rr	g11(r)	E11-	E11+	Expect	g12(r)	E12-	E12+	Expect
0.50	+r	19.9600000	0.4071000	1.7326000	1.0831684	0.0000000	0.0000000	0.0000000	0.0000000
1.50	+r	17.0493000	0.5437000	1.6350000	1.0602789	0.0000000	0.0000000	0.0000000	0.0000000
2.50	+r	14.0978000	0.6263000	1.4757000	1.0582842	0.0000000	0.0000000	0.0000000	0.0000000
3.50	+r	11.6271000	0.8178000	1.3682000	1.0507368	0.0000000	0.0000000	0.0000000	0.0000000
4.50	+r	9.5810000	0.7798000	1.3273000	1.0564684	0.0000000	0.0000000	0.0000000	0.0000000
5.50	+r	8.3210000	0.8089000	1.2150000	0.9861684	0.0000000	0.0000000	0.0000000	0.0000000
6.50	+r	7.2732000	0.8022000	1.1272000	0.9614842	0.0000000	0.0000000	0.0000000	0.0000000
7.50	+r	6.6848000	0.8795000	1.1553000	0.9754263	0.0000000	0.0000000	0.0000000	0.0000000
8.50	+r	6.0997000	0.8702000	1.1641000	1.0171474	0.0000000	0.0000000	0.0000000	0.0000000
9.50	+r	5.6139000	0.8822000	1.1749000	1.0165895	0.0000000	0.0000000	0.0000000	0.0000000
10.50	+r	5.0888000	0.8604000	1.1505000	1.0044421	0.0000000	0.0000000	0.0000000	0.0000000

The *.res results file (test.res). The **first 11 lines** contain the information on the settings of the analysis; the following part contains a table with the results of the analysis for the particular summary statistic selected. The important information for this example is heighted in grey. Note that the cell-size indicates in the standard analysis mode only the bin of the distance axis (given in data units), but in the grid-based standard analysis mode it indicates the size of the cells. In the following lines:

- The **first column** gives the spatial distance *r* of the point-pattern analysis in units of bins,
- the second and third column provide a summary of the Monte Carlo significance test of the null model ("-": data at scale *r* below the simulation envelopes, "r": inside the simulation envelopes, and "+": above the simulation envelopes; second column for univariate analysis, third column for bivariate analysis),
- **columns 4, 5, 6, 7**: results of univariate analysis (column 4: summary statistics of the data, column 5: lower simulation envelope, column 6: upper simulation envelope, column 7: expectation under the null model),
- **columns 8, 9, 10, 11:** results of bivariate analysis (column 8: summary statistics of the data, column 9: lower simulation envelope, column 6: upper simulation envelope, column 7: expectation under the null model).

2.6 Goodness-of-Fit (GoF) tests

The simulation envelopes provide a good idea on the range of the summary statistics expected under the null model, but they cannot be used to assess the general fit of the model because of problems associated with Type I error inflation. This can be avoided by using a Goodness-of-Fit test (GoF) (see section 2.5.1.2 of Wiegand and Moloney 2014). A GoF is used to test if a given null model (or point process model) fits a given summary statistic of the observed data over a given distance interval (r_{min}, r_{max}). The GoF test collapses the scale-dependent information of a functional summary statistic [e.g., $g(r)$] into a single index u_i. The index u_i represents the accumulated deviation of the observed summary statistic from the expected summary statistic under the null model, summed up over an appropriate distance interval (r_{min}, r_{max}) (i.e., a Cramer-von Mises type statistic; Loosmore and Ford 2006):

$$u_i = \sum_{r=r_{min}}^{r_{max}} (\hat{H}_i(r) - H(r))^2$$

where the $\hat{H}_i(r)$ is the empirical summary statistic of the observed pattern (i = 0) and that of the simulated patterns (i = 1,...m), and $H(r)$ the expected summary statistic under the null model. If the expected test statistic $H(r)$ is not known analytically, $H(r)$ can be replaced by

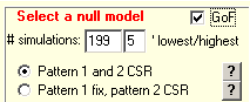
$$\overline{H}_i(r) = \frac{1}{m} \sum_{j=0, j\neq i}^{m} \hat{H}_j(r)$$

which is the average over all summary statistics, $\hat{H}_i(r)$ except the one with index i. Note that $\overline{H}_0(r)$ yields the average over the summary statistic of all m simulated patterns and provides therefore an unbiased estimate of $H(r)$ under the null model. For the GoF test the u_i are calculated for the observed data (i = 0) and for the simulated data (i = 1...m) and the rank of u_0 among all u_i is determined. The observed P value of this test is

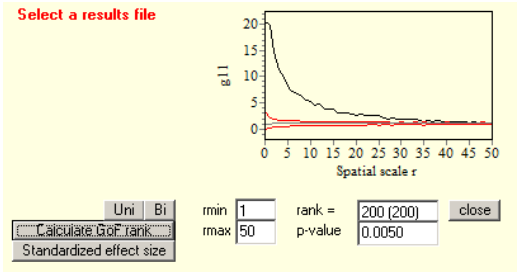
$$\hat{p} = 1 - \frac{rank[u_0] - 1}{m + 1}$$

For example, if the u_0 computed for the observed pattern was larger than the u_i computed for each of the m = 199 simulations of the null model we have rank[u_0] = 200 and $\hat{p} = 1 - (199/200) = 0.005$.

Note that the GoF test is somewhat sensitive to the distance interval selected. For example, if the departure from the null model occurs only at small scales of say 5m, but the test is conducted over an interval of 0-100m a true departure may be overpowered and not detected. Therefore use only an interval where departures from the null model are (a priori) expected. The P-value alone does not always convey the nature of discrepancy between the data and the null model. It should always be used in conjunction with visual inspection of the simulation envelopes.

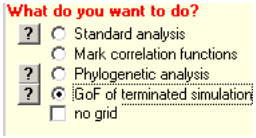


Programita allows you to access the GoF test in two ways. First, after termination of a simulation of the null model a small checkbox "**GoF**" appears top right on the window "Select a null model". After enabling the check box a window appears where you need to click "**Uni**" or "**Bi**", depending if the analysis of interest is uni- or bivariate, respectively.



Then a small graph with the observed summary statistic and the lowest and highest values of the null model appear. Provide now the distance interval (rmin, rmax) to be tested and click "**Calculate GoF rank**". The rank and the associated P-value are then provided.

Programita saves for the test statistic selected a temporary file Uni_confidence.env (or Bi_confidence.env). This file contains the observed univariate summary statistic of the data (first line) and of all simulations of the null model (following lines).



The second way to access the GoF test is to click in the menu "What do you want to do?" the option "**GoF of terminated simulation**". With this option *Programita* reads a *.env file that you saved after running a simulation with "**Save results**".

The Cramer-von Mises type test statistic u_i presented above is a one-sided test statistic. A similar two-sided test statistics can be constructed based on the standard T-test (Diggle et al. 2007; equation 2.10 in Wiegand and Moloney 2014). This test uses the variables $S_i(r)$, the summary statistic for the observed data (for i = 0) and the summary statistics of the m simulations of the null model (for i = 1 to m), and the variables $E(r)$ and $V(r)$ which are the mean and variance of the summary statistic of the m simulations of the null model at distance r. The test is conducted over an appropriate distance interval (r_{min}, r_{max}) where departures from the null model are expected (before conducting the test). For the data (i = 0) and the m simulations of the null model (for i = 1 to m) the test statistic

$$T_i = \sum_{r=r_{min}}^{r_{max}} \left(\frac{S_i(r) - E(r)}{\sqrt{V(r)}}\right) = \sum_{r=r_{min}}^{r_{max}} T_i(r)$$

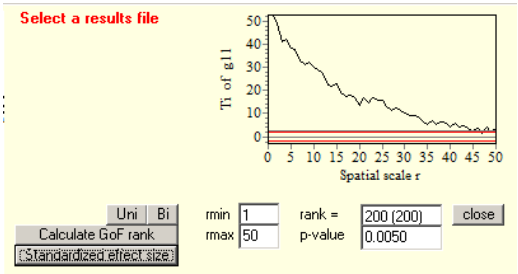
is estimated which sums up the T_i-values over the selected distance interval. Basically, the test statistic $T_0(r)$ is a transformation of the observed test statistic $S_0(r)$ (subtracting of expectation and dividing by standard deviation) with 5% simulation envelopes having constant values of -1.96 and 1.96 This transformation holds if the values of $S_i(r)$ (i= 1,..m) follow a normal distribution with mean $E(r)$ and variance $V(r)$.

The p-value of the test is obtained by estimating T_0 from the data and comparing it to each of the m estimated T_i statistics from the null model. The significance level of the test is given by

$P = (k + 1)/(m + 1)$,

where k is the number of simulated T_i greater (or smaller) than T_0 if the departure was positive (or negative).

Access the GoF test as described above. A small graph with the observed summary statistic and the lowest and highest values of the null model appear. Provide now the distance interval (rmin, rmax) to be tested and click "**Standardized effect size**".



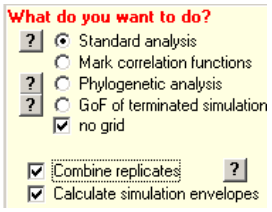
Programita now estimates the $T_0(r)$ values

$$T_0(r) = \frac{S_0(r) - E(r)}{\sqrt{V(r)}}$$

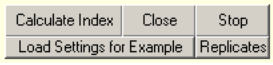
for the selected distance interval and plots the values of $T_0(r)$ together with the critical bands for a P-value of 0.05 (red) and 0.01 (grey). The rank and the associated P-value of the T_i test over distance interval (r_{min}, r_{max}) are then provided.

Note that the T_i test statistic can have negative and positive values and if the observed summary statistics shows at some distances positive and at others negative departures from the null model, they may cancel.

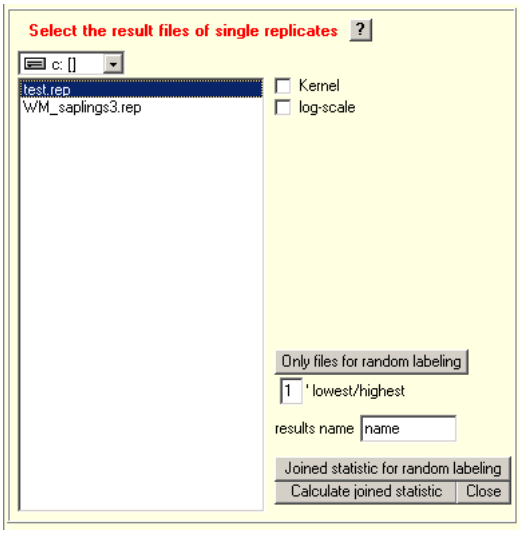
2.7 Show results of previous analyses



Programita offers a convenient possibility to show the results of previous analyses. In the standard mode, this works only if the option "**Combine replicates**" was enabled when doing the original analysis.



To show the results of a previous analysis, apply the button "**Replicates**" and a window with a list of results files appears. Highlight test.rep, and click "**Calculate joined statistic**". The result of the analysis will then appear.



See also the description of combine replicates. Clearly, if you "combine" only the results of one "replicate", you view the analysis of one analysis.

2.8 Run series of analyses

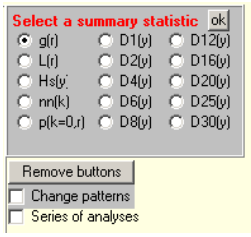
Sometimes you have to conduct many times the same analysis. *Programita* provides you means for doing this automatically for the standard modes and the mark correlation function mode. There are three different possibilities for this:

- using numbered files
- select the files to be analyzed from a list (only mode for mark correlation functions)
- for bivariate standard analysis you can analyze all pairs of univariate patterns defined in two file lists. This is practical if you have many pairs to analyze because you need not to store all data files as in the first two cases.

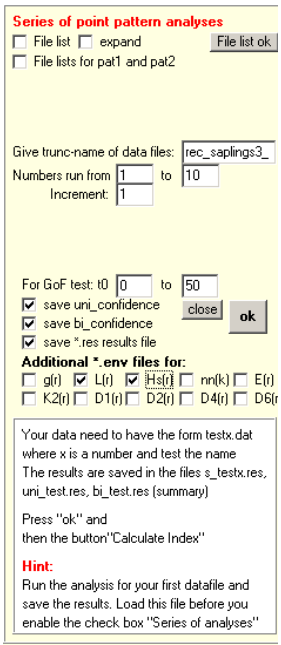
Programita conducts many individual analyses, outputs results files for each analysis, and one summary file that provides an overview over all analyses.

2.8.1 Multiple analyses with numbered files under the standard mode

The first step is to conduct the analysis with one of the data files. Once this is done, select the check box "**Series of analysis**".



A window opens where you need to provide the specifications of your series of analysis. Your data files must all follow the name convention "name_n.dat" where name is a name you can provide in the "**Give trunk name of data files**" and n is the number of the data file. You can specify the first number and the last number and an increment.



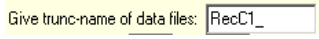
Programita conducts a Goodness-of-fit (GoF) test for each analysis. You can specify the distance interval over which to conduct the GoF test.

Programita can output for the analysis of each data file several results files:

- uni_confidence: the observed univariate summary statistic of the data (first line) and of all simulations of the null model (following lines)
- bi_confidence: same as uni_confidence, but for bivariate functions.
- *.res: this is the results file which contains all your settings. This file should be outputted.
- a *.txt summary file with name Summary_name.txt
- if you check additional summary statistics at "**additional *.env files for**" *Programita* outputs the corresponding *.env files.

Once all settings are specified, click the fat **ok** bottom 

, and then "**Calculate Index**" to start the series of analyses.

After termination of the simulation series, you can load the comma delimited summary file into EXCEL. The summary file has the name Summary_truncname.txt where truncname is the shared name of all data files you provided in 

This is an example of the summary file for an univariate analysis:

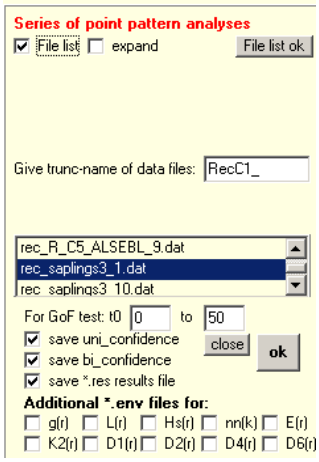
Dataname	nr	r0	r1	rank11	rank12	anzp1	anzp2	rank11_g	rank11_L	rank11_D	rank11_Hs	rank11_K2	g(r)11 0	g(r)11 1	g(r)11 2	g(r)11 3	g(r)11 4	g(r)11 5	*	g(r)-11 0	g(r)-11 1	g(r)-11 2	g(r)-11 3
rec_saplings3_1.dat	1	0	50	20	0	626	0	20	20	20	20	20	15.05	18.19	15.42	12.26	9.19	7.64	*	0.00	0.14	0.33	0.47
rec_saplings3_2.dat	2	0	50	20	0	626	0	20	20	20	20	20	16.68	19.96	15.09	12.08	9.32	7.60	*	0.00	0.27	0.57	0.47
rec_saplings3_3.dat	3	0	50	20	0	626	0	20	20	20	20	20	17.90	18.86	15.31	10.95	9.93	8.04	*	0.00	0.41	0.49	0.35
rec_saplings3_4.dat	4	0	50	20	0	626	0	20	20	20	20	20	19.93	16.82	16.21	11.96	8.95	7.78	*	0.00	0.41	0.65	0.47
rec_saplings3_5.dat	5	0	50	20	0	626	0	20	20	20	20	20	13.83	19.54	16.55	11.67	8.63	7.82	*	0.41	0.41	0.49	0.41
rec_saplings3_6.dat	6	0	50	20	0	626	0	20	20	20	20	20	15.46	18.31	15.00	11.09	9.51	7.02	*	0.00	0.27	0.57	0.65
rec_saplings3_7.dat	7	0	50	20	0	626	0	20	20	20	20	20	15.05	17.91	16.04	11.53	9.80	7.52	*	0.00	0.41	0.57	0.64
rec_saplings3_8.dat	8	0	50	20	0	626	0	20	20	20	20	20	14.64	18.63	14.46	11.40	8.29	8.21	*	0.00	0.41	0.49	0.64
rec_saplings3_9.dat	9	0	50	20	0	626	0	20	20	20	20	20	13.83	17.66	16.26	11.93	9.38	7.53	*	0.00	0.14	0.49	0.76
rec_saplings3_10.dat	10	0	50	20	0	626	0	20	20	20	20	20	16.68	19.56	15.83	10.46	8.51	8.69	*	0.00	0.41	0.49	0.29

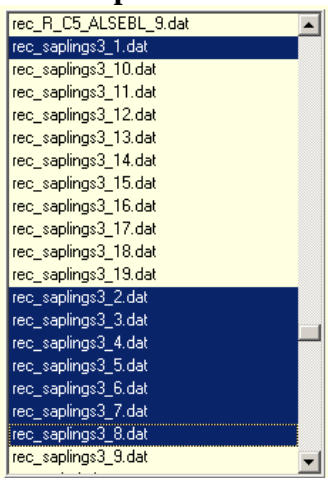
- (r0, r1) is interval of the GoF test,
- "rank 11" is the rank of the GoF test for the univariate analysis,
- "rank12" is the rank of the GoF test for the bivariate analysis,
- "anzp1" and "anzp2" give the number of points of pattern 1 and 2, respectively.
- The "rank11_g", "rank11_L", "rank11_D", "rank11_Hs", "rank11_K2" show the rank of the GoF test for the summary statistics $g(r)$, the $L(r)$, $D(r)$, $H_s(r)$, and $K2(r)$, respectively.
- The following lines are observed summary statistics and simulation envelopes, g11 is $g(r)$, g-11 and g+11 are the lower and upper simulation envelopes of $g(r)$, etc.
- The rank0, rank1, .. , rankr give the rank of the GoF test of the selected summary statistic at distance r.

In the **standard grid based mode**, you can only output the $O(r)$ or the $L(r)$ in the summary file, and additionally the $D(r)$ as *.env file.

2.8.2 Multiple analyses with files selected from a list

As above, the first step is to conduct the analysis with one of the data files. Once this is done, select the check box "**Series of analysis**".
A window opens where you need to provide the specifications of your series of analysis. To select files from the working directory select "**File list**". A list with files appears. To enlarge the file list click "**expand**".





Now you can select all files you want to analyze. The trunk-name is now only used to name the summary output file.

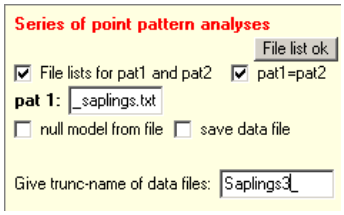
If your data are selected, click "**File list ok**" to confirm your selection.

Once all settings are specified, click the fat **ok** bottom 

, and then "**Calculate Index**" to start the series of analyses.

2.8.3 Multiple bivariate analyses with all pairs of files from two file lists

As above, the first step is to conduct one analysis with a bivariate data file. Once this is done, select the check box "**Series of analyses**".



A window opens where you need to provide the specifications of your series of analyses. To select files for pairwise analyses select "**File list for pat1 and pat2**".

If pattern 1 and 2 should be selected from the same list (e.g., bivariate analyses of recruits of different species) select "**pat1=pat2**". This is necessary to omit that the same file is selected as pattern 1 and pattern 2.

Insert the name of the file list for pat 1 (and if appropriate, for pat 2). The file list is an *.txt ASCII file with the names of the files to be analyzed (but without the *.dat extension).

The trunk-name is now only used to name the summary output file.

If your data are selected, click "File list ok" to confirm your selection.

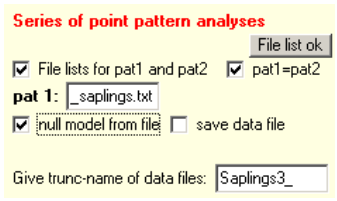
Once all settings are specified, click the fat **ok** bottom 

, and then "**Calculate Index**" to start the series of analyses.

this is an example of a file list (rec_saplings3.txt):

```
rec_saplings3_1
rec_saplings3_2
rec_saplings3_3
rec_saplings3_4
rec_saplings3_5
```

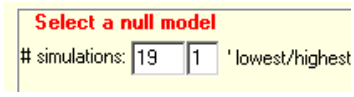
You can use this analysis series also for cases where you previously saved the null model for the different patterns listed in the file lists, for example generated with **pattern reconstruction**.



In this case you need to click "null model from file". The null model patterns corresponding to your data files must follow the name conventions:

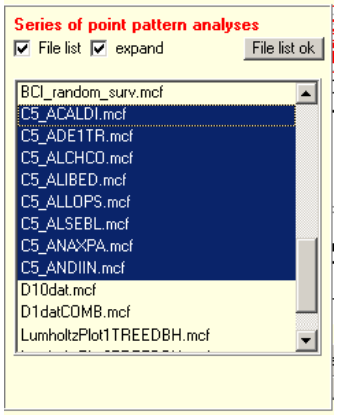
data file: `name.dat`
null model file: `rec_name_n.dat`

where `name` is the data file (e.g., Saplings3 in the example from the file list above) and `n` the number that should run from 1 to the number of # simulations of the null model specified in the window "**Select a null model**".



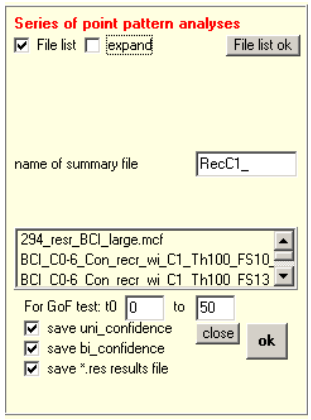
2.8.4 Multiple analyses with mark correlation functions

Running series of analyses works for mark correlation functions in the same way as for the standard analysis but only with the list option. The first step is to conduct the analysis with one of the data files. Once this is done, select the check box "**Series of analyses**". A window opens where you need to select "**File list**". A list with files appears. To enlarge the file list click "expand".



Now you can select all files you want to analyze. If your data are selected, click "**File list ok**" to confirm your selection. Provide also the name of the file that contains a summary of the results of the series of analyses.

To get the *.env files for the GoF test of individual analyses click "**save uni_confidence**" and "save **bi_confidence**" if appropriate. Once all settings are specified, click the fat **ok** bottom  , and then "**Calculate Index**" to start the series of analyses.



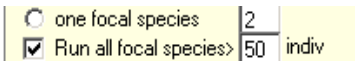
After termination of the simulation series, you can load a comma delimited summary file (name convention "**Summary_mcf_name.dat**") into EXCEL. This is an example of the summary file for an univariate analysis:

dataname	#pat 1	# pat 2	#pat 3	r0	r1	rank11_	rank12_	mcf11(0)	mcf11(1)	mcf11(2)	mcf11(3)	mcf11(4)	mcf11(5)	*	E11-(0)	E11-(1)	E11-(2)	E11-(3)	E
C5_ADE1TR.mcf	142	0	0	0	50	13	0	0.18	0.16	0.27	0.19	0.62	1.06	*	0.12	0.21	0.36	0.28	
C5_ALCHCO.mcf	227	0	0	0	50	17	0	0.24	0.24	0.61	0.77	0.34	0.53	*	0.25	0.39	0.10	0.48	
C5_ALIBED.mcf	370	0	0	0	50	3	0	0.46	0.63	0.69	1.11	1.25	0.97	*	0.57	0.57	0.70	0.59	
C5_ALLOPS.mcf	103	0	0	0	50	8	0	0.00	0.62	0.12	0.00	3.09	0.12	*	0.00	0.23	0.06	0.00	
C5_ALSEBL.mcf	7754	0	0	0	50	20	0	0.29	0.26	0.48	0.61	0.59	0.76	*	0.79	0.79	0.81	0.85	
C5_ANAXPA.mcf	794	0	0	0	50	20	0	0.95	1.04	1.08	1.07	1.09	1.08	*	0.94	0.96	0.98	0.98	

- #pat 1, #pat 2, and #pat 3 gives the number of points of type 1, 2 or 3 points, respectively,
- (r0, r1) is interval of the GoF test,
- rank 11 is the rank of the GoF test for the univariate analysis,
- rank12 is the rank of the GoF test for the bivariate analysis.
- The following lines are the observed summary statistics (using the mark correlation function you selected in the example analysis) and simulation envelopes (indicated by mvf11, E-11, E+11), and mcf11_exp is the expectation of the mark correlation function.

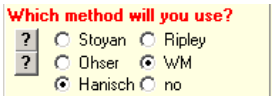
2.8.5 Multiple analyses for multivariate analysis using a dissimilarity matrix

I did not implement a Series of analyses option for this data type because it will be in most cases a community level analysis. However, there is a possibility to conduct series of "individual" analyses where the individuals of a given species are selected as focal points and the individuals of the entire community are used as second points at distance *r* away. To use this option enable the checkbox "**Run all focal species**" in the window specifying the null model:

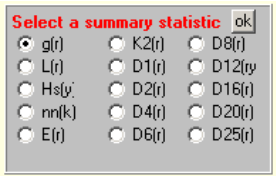


2.9 Combine results from replicate analyses

In some cases you may have maps of several replicate plots of a larger point pattern under identical conditions. In this case the resulting test statistics of the individual replicate plots can be combined into average test statistics (Diggle 2003: page 123; Illian et al. 2008: page 263; Wiegand and Moloney 2014: section 3.2). This is of particular interest if the number of points in each replicate plot is relatively low. In this case the simulation envelopes of individual analyses would become wide, but combining the data of several replicate plots into average test statistics increases the sample size and thus narrows the simulation envelopes. When considering different species as replicates, average test statistics are also an effective way of summarizing the results of an analysis on the community level. Average test statistics based on replicates can also be used to implement specific null models that would otherwise require very specific software. Section 3.2.1 of Wiegand and Moloney (2014) provides details on the aggregation formulas for different summary statistics and section 3.2.2 several examples.

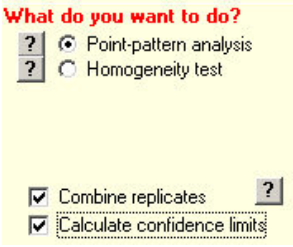


The default estimators of the *Programita* standard mode (and the grid-based mode) use the WM estimators for the pair correlation and the *K*-function based on the quantities $\lambda g(r)$ and $\lambda K(r)$. The corresponding aggregation formulas for the WM estimator are provided in equations 3.114 and 3.117 in Wiegand and Moloney (2014). For the other estimators available in the standard mode (Stoyan, Ripley, and Ohser) the aggregation formula for $g(r)$ and $K(r)$ is based on the abundance weighted mean of the $g(r)$ and $K(r)$ for the individual plots where the weight is the abundance of the focal pattern 1 [because it combines $\lambda g(r)$ and $\lambda K(r)$].



If you select other summary statistics than $g(r)$ or $L(r)$, the aggregation formula used by *Programita* is also the abundance weighted mean of the summary statistics of the individual replicates where the weight is the abundance of the focal pattern 1. Note that it makes little sense to apply the aggregation formula to the K2 function, here you should first estimate the pair correlation function and then estimate the derivative.

2.9.1 Combine replicates for standard analysis



Before running an individual standard analysis enable the checkbox "**Combine replicates**", then run all analyses with replicate plots of the same treatment with the same settings (this is important!) and the same summary statistic(!), i.e., do not change the maximal scale analyzed, or the grid size, or the summary statistic. This can be done most conveniently with the "**Series of analyses**" option described above.

When the option "**Combine replicates**" is enabled, *Programita* creates specific results files that contain all information necessary for combining the replicates. If your data file was named "name.dat", the corresponding results file will be called "WM_name.rep" if you used the pair correlation function (or any other summary statistic in the standard mode other than the *L*-function) and "R_name.rep" if you select the *L*-function.

In case of using the random labeling null model, more information is needed since e.g., g_{12} as well as g_{21} has to be calculated. In this case the corresponding results files are named "WM_name_1.rep" and "WM_name_2.rep" (or R_name_1.rep" and "R_name_2.rep for the *K*-function).

This is a typical WM_name.rep output file:

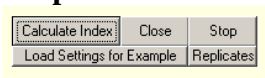
```
    3    199  626     0 250000  W-M     1  gridless  g(r)
 0  0   1966.6370   100.0000     0.0000  12732.40    0.00
 0  1   5848.9918   290.0000     0.0000  12415.13    0.00
 0  2   9639.6005   354.0000     0.0000  9195.57    0.00
 0  3  13395.3274   374.0000     0.0000  6991.21    0.00
 1  0   1966.6370     2.0000     0.0000  254.65    0.00
 1  1   5878.7354    10.0000     0.0000  425.94    0.00
 1  2   9770.7399    14.0000     0.0000  358.79    0.00
 1  3  13644.6155    30.0000     0.0000  550.55    0.00
 2  0   1966.6370     4.0000     0.0000  509.30    0.00
 2  1   5885.8916    16.0000     0.0000  680.68    0.00
 2  2   9774.6603    34.0000     0.0000  870.99    0.00
 2  3  13663.3306    26.0000     0.0000  476.49    0.00
C1 C2  C3           C4           C5      C6         C7
```

The header contains basic information on the number of distance bins used (3), the number of simulations of the null model (199), the number n_1 of points of pattern 1 (626), the number n_2 of points of pattern 2 (0), the area of the observation window in units of the bin (250000), and if the L-function (R) or any other summary statistic was selected (W-M). Additionally, in the standard mode the header contains information on the test function selected [1 and g(r)], and (gridless). The following columns contain information of the estimators:

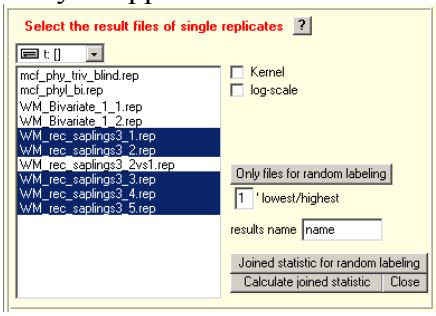
- C1: number of simulation of the null model where 0 are the observed data and 1, 2, are the simulations of the null model.
- C2: the distance bin
- C3: the denominator of equation (3.106) in Wiegand and Moloney (2014). If an estimator other than WM is selected (i.e., Ohser, Stoyan or Ripley), the denominator yields n_1 times the area of a ring with radius r and width 1 [$n_1 2\pi r$] for the pair correlation function, and n_1 times the area of a circle with radius r [πr^2] if the *L*-function is selected. (Note that the bins are 0.5, 1.5, 2.5, … for the pair correlation function and 0, 1, 2, 3,… for the *L*-function)
- C4: the numerator of equation (3.106) in Wiegand and Moloney (2014). If an estimator other than WM is selected, the numerator yields $n_1 2\pi r\lambda g(r)$ for the pair correlation function, and $n_1 \pi r^2 \lambda K(r)$ if the *L*-function is selected. Application of the aggregation formulas 3.114 and 3.117 in Wiegand and Moloney (2014) then yield the n_1 weighted mean of the summary statistics of the individual replicates.
- C5: same as C4, but for the bivariate summary statistic.
- C6: n_1 times the summary statistic selected, for example $n_1 D^k{}_{11}(r)$ if you selected the distribution function to the kth neighbour.
- C7: same as C6, but for the bivariate summary statistic.

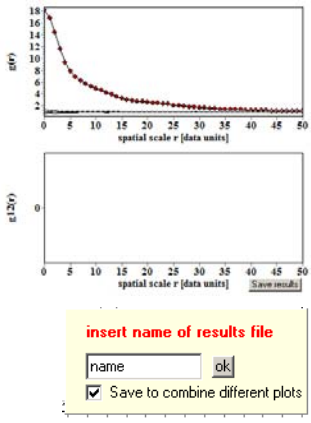
The columns C1-C5 are the same for all summary statistics selected, columns C6 and C7 contain the information on the summary statistics other than $g(r)$ and $L(r)$.

Once you completed all analyses close *Programita*, open it again and click the button "**Replicates**" below the "Stop" button:



A window opens where you can select the files you like to combine. Highlight the files you want to combine and click "**Calculate joined statistic**", and the result of the combined analysis appears.





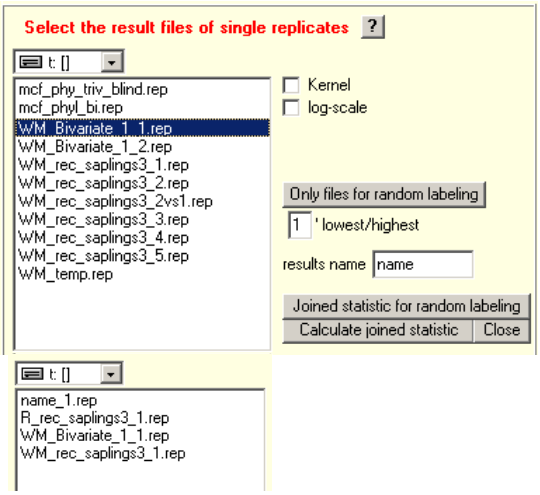
You can save the results using the "**Save results button**". Insert a name. If "name" stands for the selected name, the results file "name.res" gives you the mean weighted summary statistic, the file "name.rep" the file that allows you to view the results with "**Combine replicates**", and *.env files with the summary statistics for the observed data and the null model simulations for use in the GoF test.

Note that these *.rep and *.env files contain at the end a list of the files you combined.

```
This result file joined the results from point-pattern analysis of several single analyses in
different replicate plots, namely:
WM_saplings3_1.rep
WM_saplings3_2.rep
WM_saplings3_3.rep
WM_saplings3_4.rep
WM_saplings3_5.rep
```

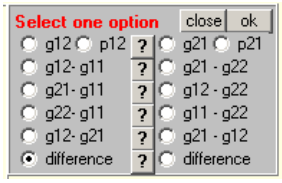
2.9.2 Combine replicates for random labeling

If the null model used in your analyses was random labeling, *Programita* saved two results files per analysis. To simplify selection of results files and to tell *Programita* that you will combine replicates that used random labeling click "**Only files for random labeling**".



In this case *Programita* hides all *.rep files which are not random labeling. Now select the replicates as before, but click "**Joined statistic for random labeling**".

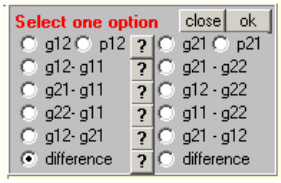
In case of random labeling you can view the results of all test statistics by using the test statistics in the window on the right.



The "WM_name_1.rep" and "WM_name_2.rep" (or "R_name_1.rep" and "R_name_2.rep") files for random labeling are basically the same as for the standard analysis mode explained in detail above:

```
   50     199  244     84 250000  W-M      1  gridless  g(r)
 0  0   3059.5516       2.0000       2.0000  164.10  472.77
 0  1   6864.6946       6.0000       3.0000  219.41  316.06
 0  2  11409.6905      10.0000       5.0000  220.01  316.94
 0  3  15918.2302      16.0000       4.0000  252.32  181.74
 0  4  20401.9507      18.0000       6.0000  221.47  212.69
 0  5  24862.8655      28.0000       7.0000  282.70  203.62
 0  6  29303.3423      26.0000      11.0000  222.73  271.49
 0  7  33725.9864      30.0000      10.0000  223.30  214.44
C1 C2  C3             C4            C5       C6      C7
```

The "WM_name_1.rep" contains the information on the $\lambda_1 g_{11}(r)$ and $\lambda_2 g_{12}(r)$ [or the $\lambda_1 K_{11}(r)$ and $\lambda_2 K_{12}(r)$] and the "WM_name_2.rep" contains the information on the $\lambda_2 g_{22}(r)$ and $\lambda_1 g_{21}(r)$ [or the $\lambda_2 K_{22}(r)$ and $\lambda_1 K_{21}(r)$] required to assemble the different test statistics to be selected:



2.9.3 Combine replicates for mark correlation functions

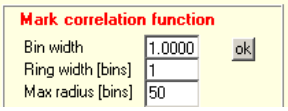
Combine replicates works for mark correlation functions in the same way as for the standard mode. However, the *.rep files are automatically created if you save the results of a mark correlation analysis or if you run a series of mark correlation analyses. In the first case they follow the convention "mcf_name.rep" and in the second case they follow the convention "s_mcf_name.rep" where the "s_" indicates series. The same applies for phylogenetic analysis, here the *.rep files follow the convention "mcf_name_phy.rep".

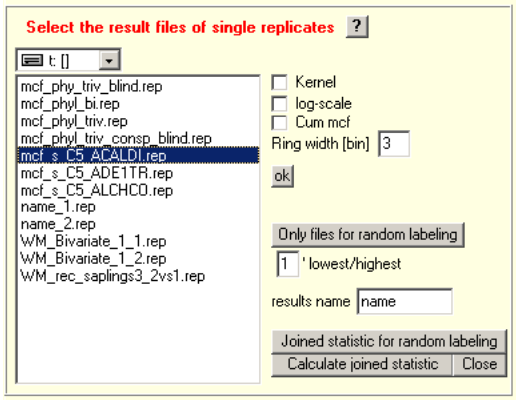
Lines 8 and 9 of the *.res results file of mark correlation functions contain also the values of the normalization constants c_t for the different mark correlation functions which are needed in the aggregation formula for normalized mark correlation functions. For example, line 8 shows for a univariate analysis the following information:

```
number points of pattern 1 =    103
mean mark p1=    77.2718
variance marks p1=  3910.3144
mean mark p1+p2=    77.2718
ct:  3910.3144   5970.9380     77.2718     77.2718   3910.3144   3910.3144
```

where "p1" refers to pattern 1 and the "ct" are the normalization constants of the six mark correlation functions.

Additionally to the standard mode, you can also change *a posteriori* the ring width, use the cumulative mark correlation function, or make the x-axis logarithmic. However, to manipulate the ring width or to make use the cumulative mark correlation function you need to run the analysis with a ring width of 1 to yield non-overlapping concentric rings.



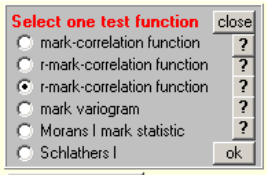


After each mark correlation analysis a temporary file MCF_test.dat is created which is then renamed into mcf_name.rep after saving results. The example below shows the first few lines of the univariate part of a mcf_name.rep output file:

```
simnr    r    MCF11_t0 MCF11_t1     MCF11_t2    MCF11_t3    MCF11_t4    MCF11_t5   Zaehler11
    0    0    1.00222  613.42544    0.99117     0.37239     0.37239     0.00155    964.00
    0    1    1.00321  613.76804    0.99173     1.25462     1.25462     0.00522    1298.00
    0    2    1.00255  612.62473    0.98988     2.45553     2.45553     0.01021    1664.00
    0    3    1.00408  615.76596    0.99496     3.85241     3.85241     0.01605    1766.00
```

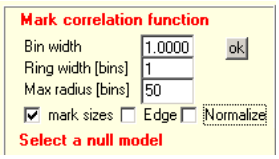
The columns of the file contain the following information:

- simnr: number of simulation of the null model where 0 are the observed data and 1, 2, are the simulations of the null model.
- r: the distance bin
- MCF11_t0, MCF11_t1, …, MCF11_t5: the values of the six univariate mark correlation function where t1 refers to the "mark-correlation function", t2 - t5 follow in descending order as shown in the selection window, and Schlather's I is t0.



- Zaehler11: the number of pairs at distance *r* (the denominator of the estimator equation (3.84) in Wiegand and Moloney (2014).
- MCF12_t0, MCF121_t1, …, MCF12_t5: the values of the six bivariate mark correlation functions.
- Zaehler12: the number of bivariate pairs at distance *r*.

Based on the information in this file, *Programita* can re-estimate the numerator and the denominator of equation (3.84) and apply the aggregation formula (3.107) in Wiegand and Moloney (2014).



If you use the default of normalized mark correlations, the information in lines 8 and 9 of the *.res file on the normalization constants c_t is used to "de-normalize" the mark correlation functions of the individual replicates by multiplying with c_t, and then the aggregation formula (3.107) is applied. The resulting non-normalized mark correlation functions are then normalized with a combined normalization constant c_t^{agg}.

The normalization constant c_t of a given replicate is the average of the test function $t(m_i, m_j)$ of all pairs of points of the pattern:

$$\hat{c}_t = \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j=1}^{n,\neq} t(m_i, m_j)$$

where m_i, m_j are the marks of the points i and j of the pattern, respectively, and n the total number of points of the pattern. Therefore, we estimate the combined normalization constant c_t^{agg} as

$$\hat{c}_t^{agg} = \frac{\sum_{m=1}^{M} c_t^m (n_m - 1) n_m}{\sum_{m=1}^{M} (n_m - 1) n_m}$$

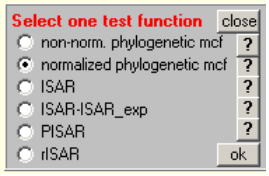
where the superscript *m* refers to replicate *m*.

If you use non-normalized mark correlation functions, the results will be the same but not be divided by $\hat{c}_t^{agg}$.

2.9.4 Combine replicates for multivariate analysis

Combine replicates for multivariate analysis works in the same way as for mark correlation functions. After each multivariate analysis with a distance matrix a temporary file MCF_test.dat is created which is then renamed into mcf_name_phy.rep after saving results (see below an example for the univariate half of the output file). The columns of the file contain the following information:

- simnr: number of simulation of the null model where 0 are the observed data and 1, 2, are the simulations of the null model.
- r: the distance bin
- MCF11_t0, MCF11_t1, …, MCF11_t5: the values of the six summary statistics where
  - t1 refers to the non-normalized phylogenetic mark correlation function,
  - t2 refers to the normalized phylogenetic mark correlation function
  - t3 refers to the average ISAR function,
  - t4 shows the average ISAR function, but subtracts the expectation of the null model,
  - t5 refers the average PISAR function, and
  - t0 refers to the average rISAR function



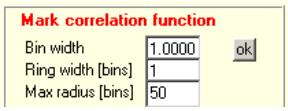
- Zaehler11: the number of pairs at distance *r* used for t1 and t2 (the denominator of the estimator equation (3.84) in Wiegand and Moloney (2014).
- MCF12_t0, MCF121_t1, …, MCF12_t5: the values of the six bivariate mark correlation functions.
- Zaehler12: the number of bivariate pairs at distance *r* used for t1 and t2.

Additionally, lines 8 and 9 of the *.res files contain the info on the normalization constants for the ISAR and PISAR function needed to estimate the rISAR. For example, line 8 shows for a univariate analysis the following information:

```
number points of foc/count of pattern 1 =   9205 /   9205
exp phl dist11=  620.82328
SD phl dist11=   137.0707
ISAR11_exp=   198.00000
PISAR11_exp=  122415.57832
```

where the "`exp phl dist11`" refers to the normalization constant c_d of the univariate phylogenetic mark correlation function $k_d(r)$ (i.e., the mean phylogenetic distance between all pairs of individuals), and "`SD phl dist11`" is the corresponding standard deviation. The "`ISAR11_exp`" and "`ISAR11_exp`" are the normalization constants of the ISAR and PISAR, respectively, which are the corresponding asymptotes.

The aggregation formula for two phylogenetic mark correlation functions is (as for mark correlation functions above) given in equation (3.107) in Wiegand and Moloney (2014). Because the ISAR and PISAR functions are based on nearest neighbor distribution functions we use an abundance weighted mean as aggregation formula where the relative abundance of pattern 1 among all replicates is used as weight. The rISAR function is estimated from the combined ISAR and PISAR functions.



You can also change *a posteriori* the ring width, use the cumulative mark correlation function, or make the x-axis logarithmic. However, to manipulate the ring width or to make use of the cumulative mark correlation function you need to run the analysis with a ring width of 1.
This is an example for the first few lines of the univariate part of a mcf_name_phy.rep output file:

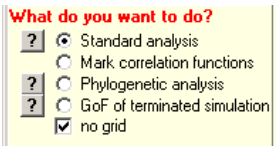
simnr	r	MCF11_t0	MCF11_t1	MCF11_t2	MCF11_t3	MCF11_t4	MCF11_t5	Zaehler11
0	0	0.994660	611.599757	0.985143	0.348398	0.348398	212.997979	3286.00
0	1	0.997803	614.284335	0.989467	1.158284	1.158284	710.371146	8286.00
0	2	0.997525	610.901445	0.984018	2.273656	2.273656	1394.035937	12460.00
0	3	0.999353	613.801751	0.988690	3.564693	3.564693	2189.610864	15758.00
0	4	1.001270	615.290318	0.991088	4.963715	4.963715	3054.806301	18902.00
0	5	1.001618	613.680947	0.988495	6.460728	6.460728	3977.493102	21960.00
0	6	1.003077	617.437710	0.994547	8.035959	8.035959	4954.475959	24598.00
0	7	1.003720	617.016484	0.993868	9.637697	9.637697	5945.819098	27250.00

3 Univariate analysis in the standard analysis mode

The standard analysis mode allows for analysis of
- univariate patterns (data type 1)
- bivariate patterns (data type 2)
- qualitatively marked patterns (i.e., random labeling analysis; data type 4).

In the following I explain the different settings, null models, and point process models for univariate analysis in detail. Most examples are taken from Wiegand and Moloney (2014).

The standard analysis mode works in most cases exactly the same way as the grid-based standard mode. The only difference is to enable or disable the checkbox "no grid" in the window What do you want to do?



I therefore report here only cases where differences occur. For example, in some cases the procedures for the standard mode are not yet implemented and you have to use instead the grid-based mode for particular analyses.

3.1 Getting started

3.1.1 Data preparation

Univariate analysis deals with a data type that comprises only the coordinates of a given point pattern. There is only one type of points and no mark considered. The univariate data type is the most analyzed data type in point pattern analysis.

The data files for univariate standard analysis must be an ASCII file with the *.dat extension and the following format (the example shows the first lines of the file saplings_3.da)t:

```
0   500 0   500 626
  3.96      55.94     1     0
277.66     230.78     1     0
273.28     235.15     1     0
296.37      99.51     1     0
273.10     217.30     1     0
 40.28       7.81     1     0
140.55     194.02     1     0
180.49     300.19     1     0
187.01     304.66     1     0
275.27     229.10     1     0
295.40      45.60     1     0
210.36      12.00     1     0
…
```

where the first line gives the size of the observation window (500 × 500 units in the example) and the number of points in the pattern (= number of lines following the header). The first two columns are the coordinates, an entry "1" in the third column indicates that the point is of pattern 1 (i.e., a type 1 focal point) and an entry "1" in the fourth column indicates that the point is of pattern 2 (i.e., a type 2 point). The value of the third and the forth columns must be for the standard analysis mode "0 1" or "1 0", no larger numbers or "1 1" are allowed.

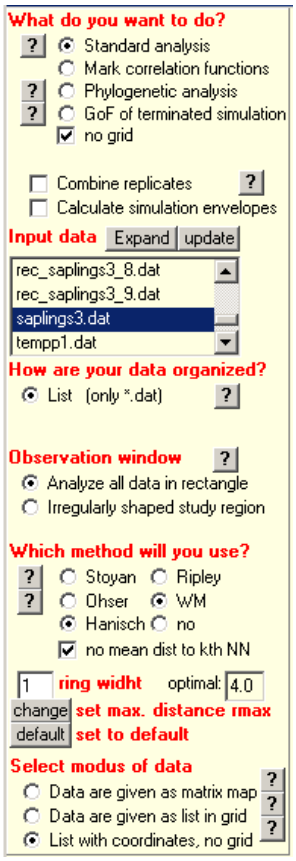
The data file must be a space or tab delimited ASCII file with the *.dat extension. If you use Excel, there is a simple, but obviously generally unknown, way of saving files of a given type with a given extension:

1. Prepare the data file in Excel following the instructions above
2. Then save as a tab delimited text file, but write "name.dat" for the name (usually you would only write name and end up with a file named name.txt. The quotation marks are important because they force Excel to save the comma delimited file under the name name.dat.

	A	B	C	D	E	F	G
1	0	500	0	500	626		
2	3.96	55.94	1	0			
3	277.66	230.78	1	0			
4	273.28	235.15	1	0			
5	296.37	99.51	1	0			
6	273.1	217.3	1	0			
7	40.28	7.81	1	0			
8	140.55	194.02	1	0			
9	180.49	300.19	1	0			
10	187.01	304.66	1	0			
11	275.27	229.1	1	0			
12	295.4	45.6	1	0			
13	19.26	54.76	1	0			
14	267.28	103.61	1	0			
15	263.08	230.49	1	0			
16	3.52	35.68	1	0			
17	0.86	52	1	0			
18	45.6	10.5	1	0			

3.1.2 Steps of analysis in standard mode and example

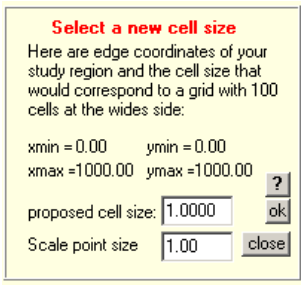
Programita estimates for data files of this type several summary statistics based on estimators detailed in Illian et al. (2008) and Chapter 3 of Wiegand and Moloney (2014). The window **Which method will you use** allows you to specify the estimators.



The standard analysis mode can be accessed with the following sequence of actions:

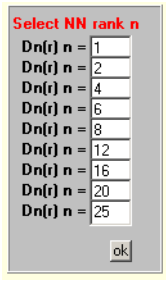
1. Select "**Standard analysis**" in window **What do you want to do?**
2. Highlight a data file in **Input data file**
3. Select "**List (only *.dat)**" in **How are your data organized?**
4. Select "**List with coordinates, no grid**" in **Select modus of data**.
5. Select "**no grid**" in **What do you want to do?**

   The window **Select a new cell size** opens and allows you to provide a bin for your analysis given in units of your data. For example, if your data are in meter units and your observation window is 500 × 500m in size, an appropriate bin would be 1m. Press "**ok**" to confirm selection of the bin.

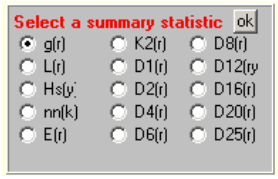


6. After selection of the bin ***Programita*** **suggests a ring width *dr*** based on equation 4.3.43 in Illian et al. (2008) [$dr = 0.2/\lambda^{0.5}$]. This equation provides a rough starting point for deciding on the ring width.

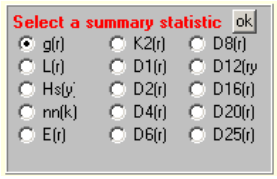
   The estimators of the pair correlation function implemented in the standard mode of *Programita* use a **default ring width of one bin** to obtain non-overlapping concentric rings. For reasons of computational efficiency you can then select only ring widths adding one bin in each direction, i.e., ring widths of 1, 3, 5, 7, … bins. You can change the ring width at the menu "**Which method will you use**". In the example file "seedlings3.dat" with 626 points within a 500 × 500m observation window and a bin of 1m this yields a ring width of $dr = 4$.



7. Selecting the option "**no grid**" opens also a small window where you can select the desired rank *k* of the distribution functions $D^k(r)$ of the distances to the kth neighbor. Default is $k = 1, 2, 4, 6, 8, 12, 16, 20$, and 25. You can thus select the rank k of nine different functions $D^k(r)$. To confirm press the small **ok** button.



8. Press button "**Calculate Index**" and *Programita* estimates a variety of summary statistics of the univariate data:

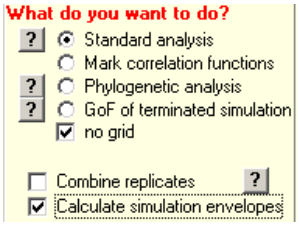


   - $g(r)$: pair correlation function
   - $L(r)$: L-function,
   - $H_s(r)$: the spherical contact distribution
   - $nn(k)$ the mean distance to the kth neighbor
   - $E(r)$ the probability that a point has no neighbor at distance within distances (r - 0.5, r + 0.5)
   - $K2(r)$ the $K2$ function
   - $D^k(r)$, the kth nn distribution functions, here with k = 1, 2, 4, 6, 8, 12, 16, 20, and 25

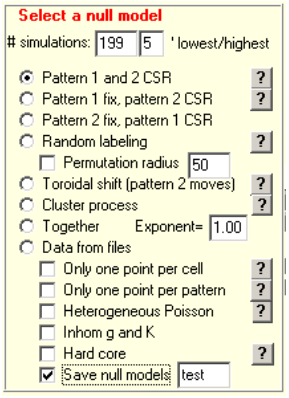
   To view the different summary statistics select the respective radio button and then the small **ok** button.

   After one analysis *Programita* saves the results of all 15 univariate (and if appropriate all 15 bivariate) summary statistics into the temporary files SumStat1.env, SumStat2.env, etc. and allows you to view all results without conducting new analysis.

9. The next step is to select a **null model or point process** model implemented in *Programita*. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface. A window will open that allows you to select a null model. In the example, we select "**Pattern 1 and 2 CSR**". In this case both patterns are independently distributed following a homogeneous Poisson processes (or Complete Spatial Randomness CSR). If the data set is univariate only the first pattern is randomized following CSR.



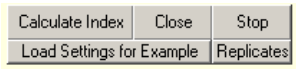
   Here you can specify the **number of simulations of the null model** (199 in the example) and the rule for the estimation of **simulation envelopes** (here the 5th lowest and highest values of the summary statistic of the 199 null model data sets).



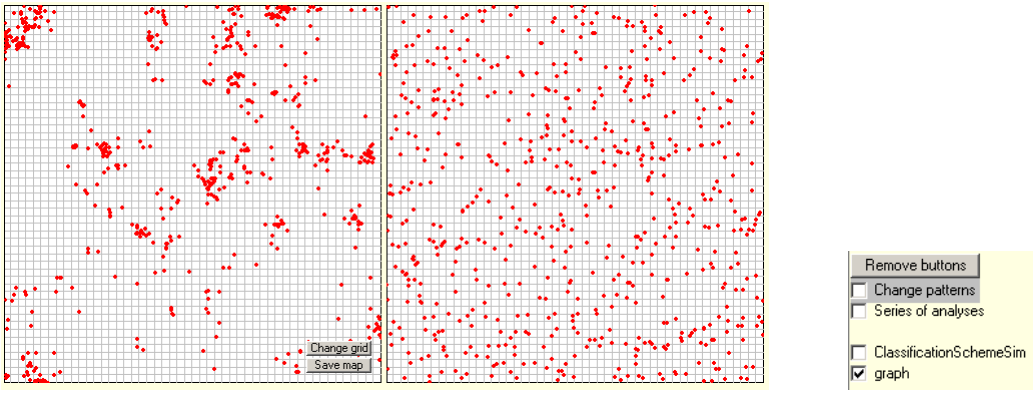
   The radio buttons in the menu "Select a null model" are the different basic options for null models or point process models whereas the check boxes are mostly additional options to specify the null model.

   The checkbox "**Save null models**" allows you to save the patterns generated by the null model as "name_n.dat"

   If all settings are specified, press "**Calculate Index**" and *Programita* conducts the simulations of the null model.

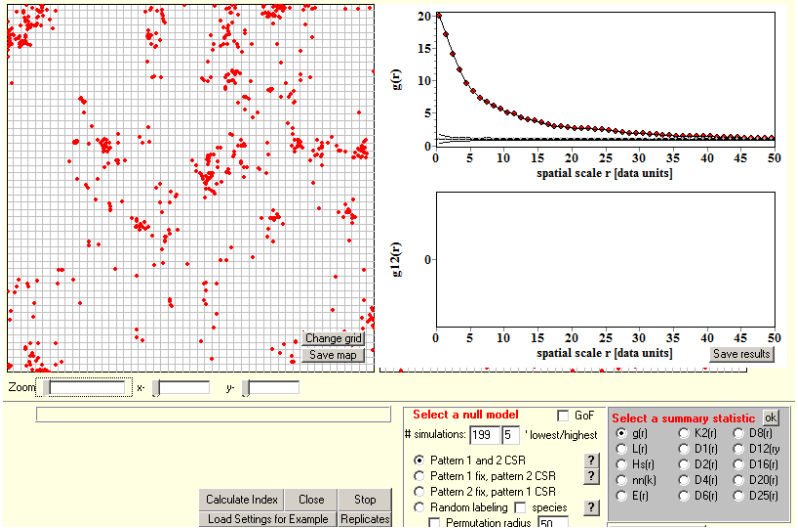


10. *Programita* shows the original point pattern being analyzed (left or top plot), and patterns of the Monte Carlo simulations of the null model (on the right or bottom) used for constructing the simulation envelopes and the GoF test.



The simulation is quicker if *Programita* does not show the plots of all simulated data. You can not show the graphs by disabling the checkbox "**graph**" at the bottom right.

After the simulations of the null model the figure with the simulated patterns of the null model disappears, and a figure with the result of the analysis appears:



The top (or left) figure shows generally the results of the univariate analysis and the bottom (right) figure shows the results of the bivariate analysis. (Exceptions are multivariate analysis using a dissimilarity matrix under the random labeling and trivariate random labeling mode). The data file in the example was univariate, therefore no figure appears for the bivariate results.

11. To save **the results of the analysis for a particular summary statistics** press the button Save results in the result graph for the bivariate analysis. *Programita* then generates a *.res file [e.g., "**g(r)_name.res**" for the pair correlation function where "name" is a name] with the summary of the results and the settings of the analysis, and a *.env file with the detailed results of the summary statistic for the data and the simulations of the null model. The *.env file can be used for the GoF test.

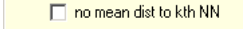
3.2 Methods for univariate standard analysis

The following examples present step-by-step instructions for the most impotent univariate analyses. If analyses of Wiegand and Moloney (2014) are repeated, I refer to them using the figure number in the book, e.g., Book _Fig4_2 refers to an analysis shown in Figure 4.2 and the corresponding data file is named Book_Fig_4_2a.dat. Other analyses are named after the null model used.

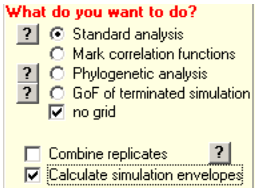
3.2.1 Homogeneous Poisson (CSR), (Book_Fig4_2.res)

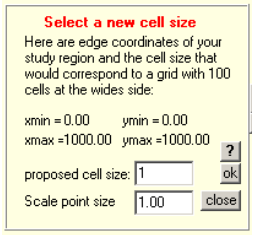
The homogenous Poisson process is characterized by two fundamental properties. First, the intensity λ of the process (i.e., the mean point density in a unit area) is a constant and therefore, the number of points in a study plot of area A follows a Poisson distribution with an expected mean of λA. Second, the points are independently distributed, which means that there is no interaction between the points of the pattern determining their locations.

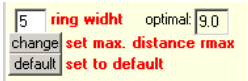
Although CSR appears in many cases overly simple, it is the basic building block for more complex null models. As example, step-by-step instructions for the analysis of Figure 4.2 in Wiegand and Moloney (2014) are provided below.

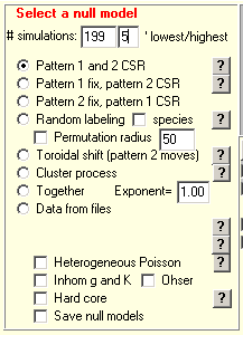
1. Execute *Programita*.
2. Highlight data file Book_Fig4_2a.dat you want to analyze in Input data file
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 5 in the menu "Which method will you use"
6. Click button "**change**" below to set maximal distance r to be analyzed. Insert 100 in small box that opens and then the small **ok** button.
7. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
8. Press button "**Calculate Index**"
9. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
10. Select "**Pattern 1 and 2 CSR**" in the window "Select a null model".
11. Specify the number of simulations of the null model (199 in the example) and the rule for the estimation of simulation envelopes (here the 5th lowest and highest values of the summary statistic of the 199 simulated null model data sets).
12. To view a large range of neighborhood ranks k in the mean distance to the kth neighbor summary statistic $nn(k)$, disable the option "**no mean distance to kth NN**" 

in the menu Which method will you use. This requires estimation of a larger nearest neighbor matrix which slows down the estimation. If the check box is enabled (default), the maximal k value is the maximal k of the $D^k(r)$'s estimated.

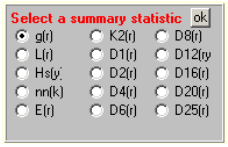


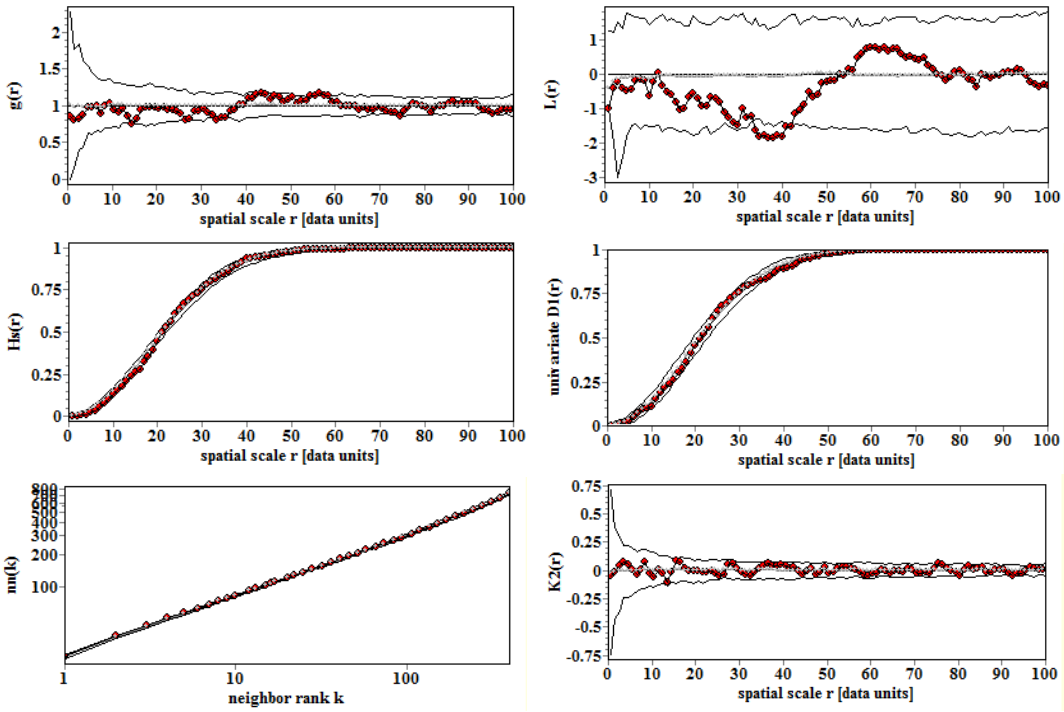






13. If all settings are specified, press the button "**Calculate Index**" and *Programita* conducts the simulations of the null model.
14. In window "Select a summary statistic" you can view the results of the analysis for the different summary statistics:



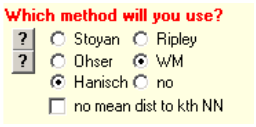


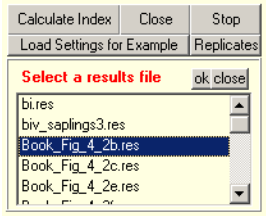
To save **the results of the analysis for a particular summary statistics** press the button Save results that appears in the result graph for the bivariate analysis and provide the name (e.g., Book_Fig.4_2b). Here is an example for the *.res results file for the spherical contact distribution $H_s(r)$:

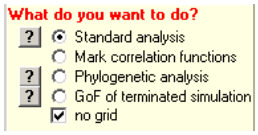
```
distance rr   Hs11(r)      E11-         E11+        Expect
   0.00 rr    0.0000000    0.0000000    0.0000000    0.0000000
   1.00 rr    0.0016000    0.0000000    0.0064000    0.0015779
   2.00 rr    0.0016000    0.0016000    0.0131000    0.0063487
   3.00 rr    0.0080000    0.0048000    0.0259000    0.0141548
   4.00 -r    0.0112000    0.0130000    0.0388000    0.0253693
   5.00 rr    0.0258000    0.0258000    0.0544000    0.0393583
   6.00 -r    0.0340000    0.0423000    0.0736000    0.0567106
   7.00 rr    0.0568000    0.0552000    0.0934000    0.0761693
   8.00 -r    0.0732000    0.0772000    0.1178000    0.0965186
   9.00 rr    0.0979000    0.0970000    0.1423000    0.1192040
  10.00 rr    0.1309000    0.1192000    0.1696000    0.1438281
  11.00 rr    0.1542000    0.1442000    0.1989000    0.1703819
  12.00 rr    0.1758000    0.1715000    0.2287000    0.1985543
  13.00 rr    0.2043000    0.2015000    0.2611000    0.2278829
  14.00 rr    0.2362000    0.2303000    0.2968000    0.2584809
  15.00 -r    0.2598000    0.2635000    0.3229000    0.2905462
  16.00 -r    0.2767000    0.2913000    0.3505000    0.3236442
  17.00 -r    0.3193000    0.3263000    0.3865000    0.3577065
```

You can load this part of the results file into a scientific graphics program to produce figures.

3.2.2 Homogeneous Poisson and estimators



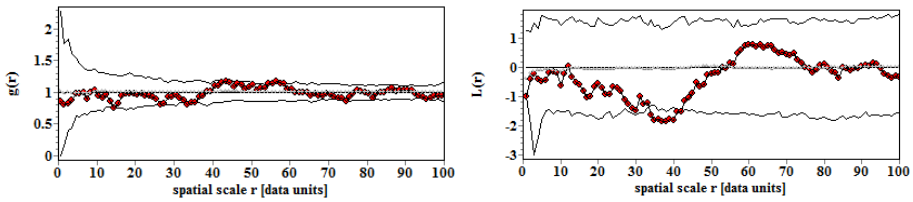




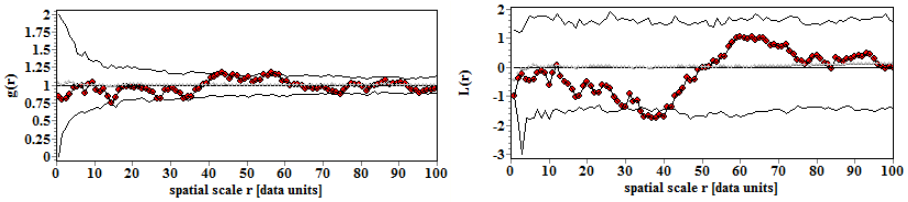
In the previous example I used the WM estimator for the pair correlation function and the *L*-function. However, *Programita* allows you to use also alternative estimators presented in Illian et al. (2008) and Wiegand and Moloney (2014). The analysis Book_Fig4_2 is therefore repeated below with different estimators.

1. Click "**Load Settings for Example**", highlight file "**Book_Fig4_2b.res**" and click small **ok**.
2. Select "**Stoyan**" in menu "Which method will you use".
3. Click "**Calculate Index**"
4. Repeat the same with "**Ripley**" and "**Ohser**".
5. To obtain the grid-based mode deselect "**no grid**"

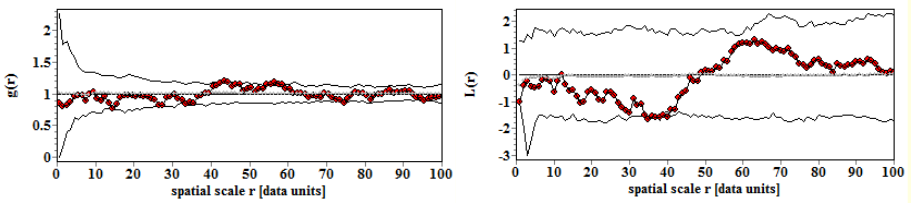
The differences among estimators are small for near random or random univariate patterns:



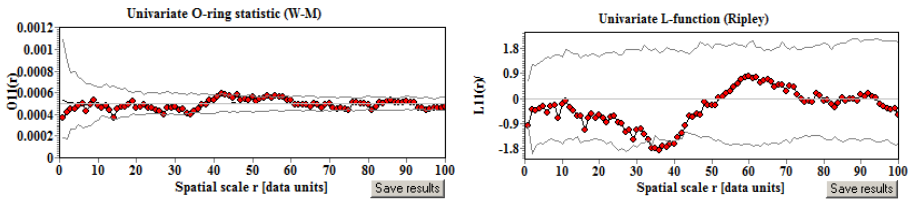
WM estimator



Stoyan estimator



Ripley estimator

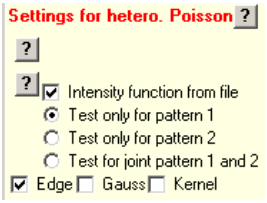


Grid-based WM estimator

3.2.3 Heterogeneous Poisson with kernel estimate (Book_Fig4_2.res)

The heterogeneous Poisson process is characterized by two fundamental properties. First, in contrast to the homogeneous Poisson process the intensity $\lambda(\mathbf{x})$ of the process depends on location **x**. Second, the points are independently distributed, which means that there is no interaction between the points of the pattern determining their locations.

The heterogeneous Poisson process is completely determined by the intensity function $\lambda(\mathbf{x})$ and therefore the estimation of the intensity function is an important ingredient of this point process model. There are basically two methods to estimate the intensity function, parametric and non-parametric methods. If you used non-parametric methods to estimate the intensity function you can read the resulting intensity file into *Programita*.



However, *Programita* allows you also to estimate the intensity function non-parametrically, directly from the data using smoothing techniques based on kernel estimators. See section 2.6.2.1 "Nonparametric Intensity Estimation" in Wiegand and Moloney (2014) for details.

Programita offers three different kernel functions

- Box kernel (neither "**Gauss**" nor "**kernel**" checked)
- Epanechnikov kernel ("**kernel**" checked)
- Gaussian kernel ("**Gauss**" checked)
- and can estimate with ("**Edge**" checked) and without edge correction ("**Edge**" not checked)

Remember that the intensity is defined basically as number of points per unit area and that an estimator of the intensity divides the number of points in a given area by the area. For a homogeneous pattern the "natural" estimator of the intensity is therefore $\lambda_n = n/A$ where n is the number of points in the observation window and A the area of the observation window. The non-parametric intensity estimators generalize this idea. Because the intensity changes along the observation window it makes sense to use smaller subareas centered at location **x** to estimate the intensity function $\lambda(\mathbf{x})$. *Programita* therefore estimates the density of points within circular moving windows $C_{(\mathbf{x})}(R)$ with radius R centered on location **x**. The moving-window estimate $\hat{\lambda}^R$ of the non-constant first-order intensity $\lambda(\mathbf{x})$ yields

$$\hat{\lambda}^R(\mathbf{x}) = \frac{\mathbf{Points}[C_{(\mathbf{x})}(R)]}{\mathbf{Area}[C_{(\mathbf{x})}(R)]}$$

where the operator $\mathbf{Points}_2[X]$ counts the points of pattern 2 in a region X, and the operator **Area**[X] determines the area of the region X. This is a box kernel estimate with fixed bandwidth R (all points located within distance R of location **x** are counted equal with weight 1 and all points at larger distance have weight zero and are not counted).

As edge correction, the number of points in an incomplete circle is divided by the proportion of the area of the circle that lies within the study region. Without edge correction, it is divided by the area of the full circle. The intensity function is then normalized to have a maximal value of one, thus ranging between zero and one.

The moving window estimator $\hat{\lambda}^R(\mathbf{x})$ involves a decision on an appropriate radius *R* of the moving window (see section 2.6.2.1 "Nonparametric Intensity Estimation"). As detailed in section 4.1.2.1 "HPP: Nonparametric Intensity Estimate to Avoid Virtual Aggregation", the heterogeneous Poisson process with box kernel intensity estimate has **a simple geometric interpretation**. While CSR basically displaces a given point with random distance and angle within the observation window, the heterogeneous Poisson process with box kernel intensity displaces each point of the pattern basically within a circular neighborhood of radius *R*.

Thus, because the bandwidth *R* is the scale of smoothing, possible departure from this null model may only occur for scales $r < R$, and for small moving windows it will closely mimic the original pattern, whereas a large moving window approximates CSR.

The two other options ("**Gauss**" or "**kernel**") are kernel functions that weight the points which are counted according to their distance to the focal location **x**. In case of the Epanechnikov kernel (enable "**kernel**") this produces smoother intensity estimates than the box kernel (Fig. 2.20 in Wiegand and Moloney 2014). The Epanechnikov kernel is based on the following weight of a point within distance *d* of location **x**:

$$w_E(d,R) = \begin{cases} 2(1-\frac{d^2}{R^2}) & d \le R \\ 0 & d > R \end{cases}$$

The other option ("**Gauss**" checked) is mostly used as analogue to the Thomas cluster process and generates an intensity function which is the superposition of two-dimensional and symmetric Gaussian curves centered in the points of the pattern (see Fig. 4.17 in Wiegand and Moloney 2014). It is based on the weight

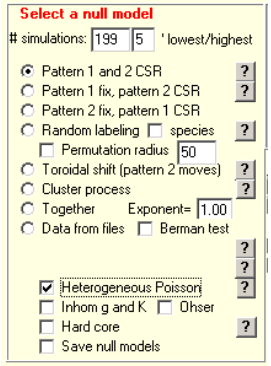
$$w_G(d,R) = \begin{cases} \frac{1}{2}\exp(-\frac{1}{2}(\frac{d}{R})^2) & d \le 3R \\ 0 & d > 3R \end{cases}$$

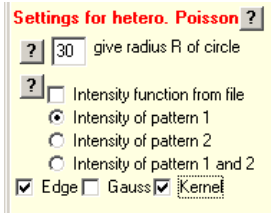
Note that the integral $\int w(r, R)\, 2\pi\, r\, dr$ over both kernel functions yields the area πR^2 of the circle. For computational reasons the Gaussian kernel is only estimated up to distances of 3R because in this case the integral yields $0.99\ \pi R^2$.

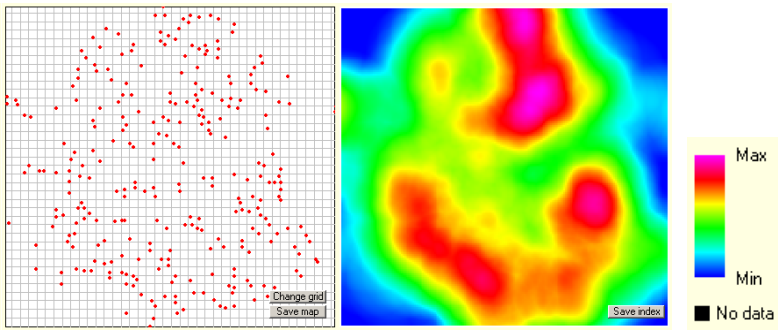
The algorithm for creating a pattern under a heterogeneous Poisson process is simple: a provisional point is placed at a random cell (x, y) in the study area, but this point is only retained with probability $\hat{\lambda}^R(x,y) / \max[\hat{\lambda}^R(x,y)]$ (the function max[X] determines the maximum of a variable X). This procedure is repeated until *n* points are distributed.

Example Book_Fig2_28h.res
The following example presents the analysis of Figure 2.28 using a heterogeneous Poisson process with non-parametric kernel estimate.

1. Execute *Programita*.
2. Highlight data file Book_Fig2_26.dat you want to analyze in Input data file
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 3 in the menu "Which method will you use"
6. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
7. Press button "**Calculate Index**"
8. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
9. Select "**Pattern 1 and 2 CSR**" in the window "Select a null model".
10. Specify the number of simulations of the null model (199 in the example) and the rule for the estimation of simulation envelopes (here the 5th lowest and highest values of the summary statistic of the 199 simulated null model data sets).
11. Click checkbox "**Heterogeneous Poisson**"
12. Go to window "Settings for hetero. Poisson" on the left and insert the bandwidth R (30m in the example), enable "**Kernel**" for the Epanechnikov kernel and select "**Intensity of pattern 1**" (because your data are univariate). Edge correction "**Edge**" is enabled by default. Click "**Calculate Index**" and *Programita* estimates the intensity function and shows the pattern and the corresponding intensity function.

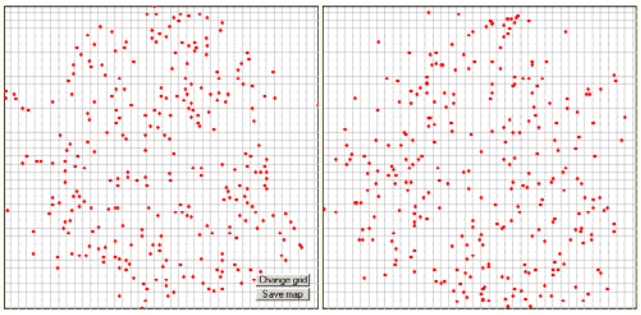




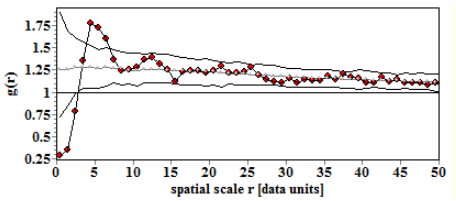


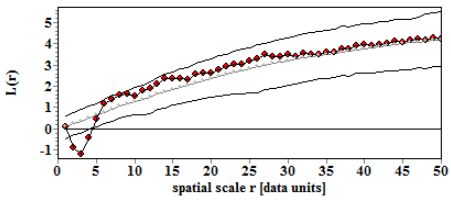
Click OK at the message box to save the intensity file. The file is saved with name int_E_Book_Fig2_26_R1_30.int where the "int_E" indicates Epanechnikov kernel, Fig2_26.dat was the data file, "_R1_30" means that the intensity was estimated with pattern 1 and bandwidth 30.

Now *Programita* conducts the analysis. You can observe during the simulations that the null model distributes the points with probability proportionally to the intensity. Here an example:

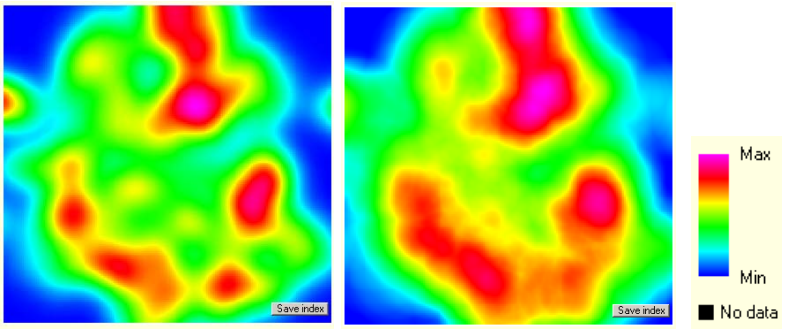


13. The result resembles that in Figure 2.28 h, i well:

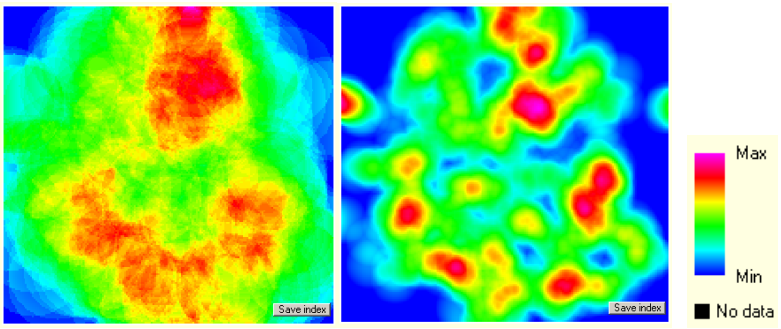




14. This is the analogous Gaussian kernel (left) with R = 10m in comparison with the Epanechnikov kernel with R = 30m (right):



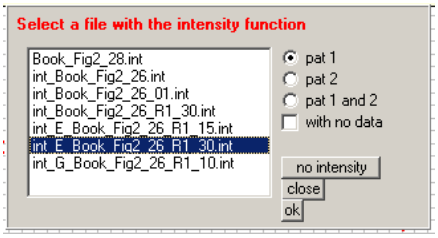
In contrast, the estimate using the box kernel with bandwidth R = 30m looks quite rugged (left) and an Epanechnikov kernel with R = 15m seems to conserve too much detail (right):



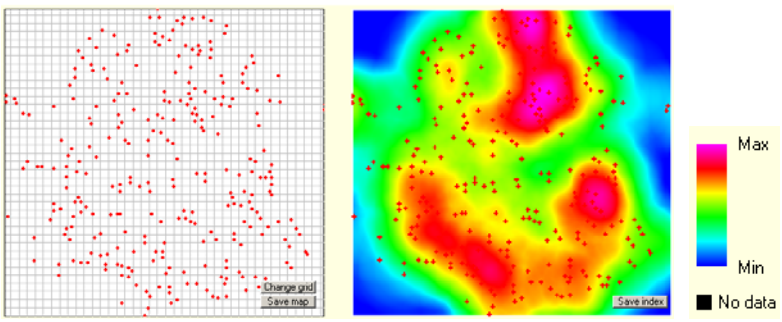
Example Book_Fig2_28h_file.res, intensity from file

This example repeats the previous analysis of Figure 2.28 using a heterogeneous Poisson process but now uses an intensity function that was saved as a *. int file.

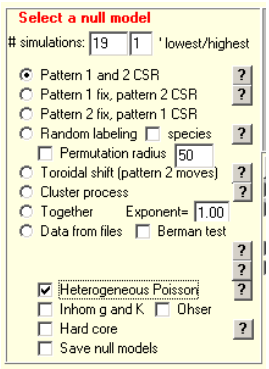
1. Execute *Programita*.
2. Highlight data file Book_Fig2_26.dat you want to analyze in Input data file
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 3 in the menu "Which method will you use"
6. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
7. Press button "**Calculate Index**"
8. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
9. Select "**Pattern 1 and 2 CSR**" in the window "Select a null model".
10. Specify the number of simulations of the null model (199 in the example) and the rule for the estimation of simulation envelopes (here the 5th lowest and highest values of the summary statistic of the 199 simulated null model data sets).
11. Click checkbox "**Heterogeneous Poisson**"
12. Go to window "Settings for hetero. Poisson" on the left and click checkbox "**Intensity function from file**". Highlight file int_E_Book_Fig2_26_R1_30.int and press the small **ok**.

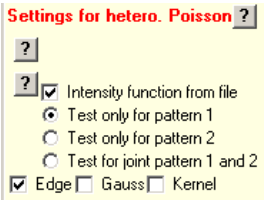


and *Programita* shows you the data on the left and the intensity function together with the data on the right.



13. Click OK at the message box and then "**Calculate Index**" and *Programita* conducts the analysis.





3.2.4 Irregularly shaped observation window (Book_Fig. 2.28.res)

Programita offers several options to analyze a univariate pattern within an observation window of irregular shape.

1. The points of the null model are only distributed inside the observation window, but otherwise no adjustments are done.
2. The observation window is explicitly reduced and the estimators of the second-order summary statistics take the reduced observation window into account. The points of the null model are only distributed inside the observation window.
3. Inhomogeneous summary statistics are used which are based on an intensity function $\lambda(\mathbf{x})$ which has a value of λ inside the observation window and zero outside (in the standard mode option 2 is implemented as option 3).

In the following example I show the first option which is based on the heterogeneous Poisson process and an intensity function which is zero outside the observation window and λ inside the observation window. In this case the CSR null model rejects points outside the observation window because they have a zero intensity and as a result the null model is CSR inside the observation window and no point of the null model will be located outside the observation window.

The **intensity file** must be an ASCII file with the *.int extension:

```
1  197  1  190  37430  1
  1  1  1  -9
  1  2  1  -9
  1  3  1  -9
  1  4  1  -9
  1  5  1  -9
  1  6  1   1
  1  7  1   1
  1  8  1  -9
  1  9  1  -9
  1  10 1  -9
….
```

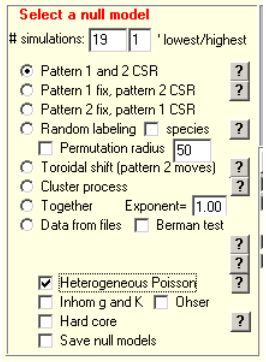
The file must describe a matrix and has therefore coordinates of a grid that run from 1 to 197 (x-coordinate) and 1 to 190 (y-coordinate). Thus, we have in total 197 × 190 = 37430 cells. Thus, the first line gives the first and last x-coordinate and the first and the last y-coordinate and the number of cells which follow. The last number is the cell size (i.e., 1 in the example).

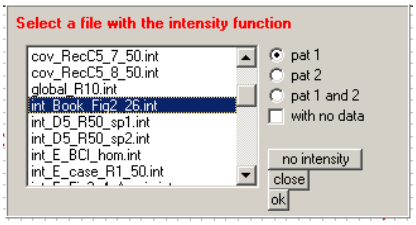
The following lines give the coordinates of all cells and its value. Each cell can have a value of -9 if it is outside the observation window or 1 if it is inside the observation window. (Note that the *.int files are always normalized between 0 and 1).

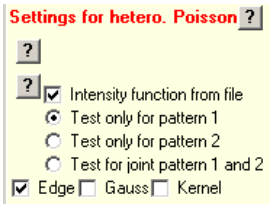
- columns 1 and 2: coordinates of the cells
- column 3: always 1
- column 4: value of normalized intensity function

Example Book_Fig2_28_opt1.res (manipulate null model, option 1)

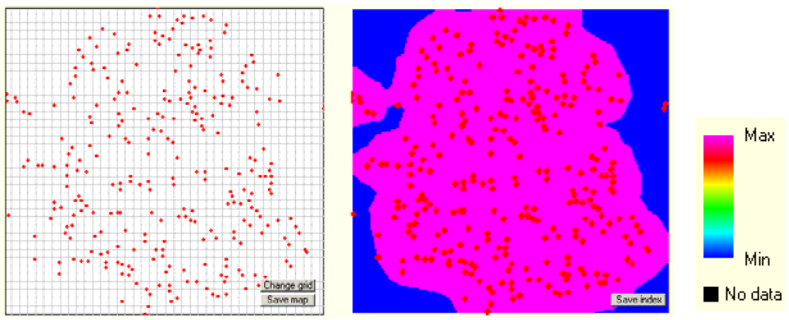
1. Execute *Programita*.
2. Highlight data file Book_Fig2_26.dat in Input data file
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 3 in the menu "Which method will you use"
6. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
7. Press button "**Calculate Index**"
8. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left.
9. Select "**Pattern 1 and 2 CSR**" in "Select a null model".
10. Specify the number of simulations of the null model (199 in the example) and the rule for the estimation of simulation envelopes (here the 5th lowest and highest values of the summary statistic of the 199 simulated null model data sets).
11. Click checkbox "**Heterogeneous Poisson**"
12. Go to window "Settings for hetero. Poisson" on the left and click checkbox "**Intensity function from file**". Highlight file int_Book_Fig2_26.int and press the small **ok**.





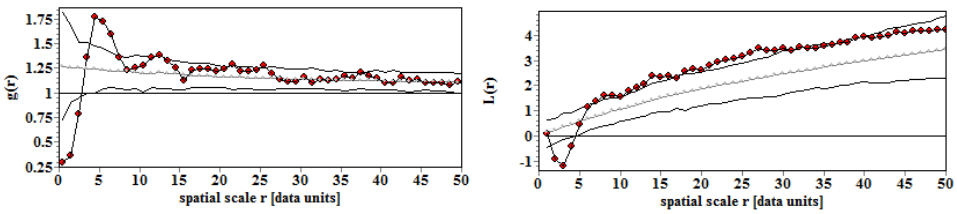


and *Programita* shows you the data on the left and the intensity function together with the data on the right:



Click OK at the message box and then "**Calculate Index**" and *Programita* conducts the analysis. You can observe during the simulations that the null model does indeed not distribute points outside the observation window.

13. The result resembles that in Figure 2.28 e, f well:



Now I show the second option to consider observation windows of irregular shape which is based on an **explicit reduction of the observation window**. In this case the estimator of the second-order summary statistics takes the reduced observation window into account and distributes the points of the null model only inside the observation window.

Two methods are available in the literature to consider observation windows of irregular shape.

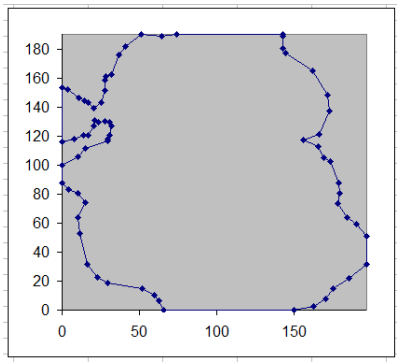
- First, Goreaud and Pélissier (1999) developed explicit equations for the Ripley edge correction for irregular observation windows where the geometrical shape is approximated by removing triangular surfaces from an initial rectangular shape. This method is not used in *Programita*.
- Second, Wiegand and Moloney (2004) approximated the geometrical shape of the observation window with an underlying grid and based the estimators of the second-order summary statistics on estimates of the mean number of points in (potentially incomplete) rings (or circles) around the points of the focal individuals and the mean area of the rings (or circles) inside the observation window. For more details on this method see section 3.1.2.2 "*O*-Ring Statistic" and equation 3.36 in section 3.1.2.7 "Ripley's *K*-Function". This method is used in the standard grid based mode in *Programita*.
- Wiegand and Moloney (2014) extended the WM and Ohser estimators of the second-order summary statistics to inhomogeneous estimators (see section 3.1.2.6 "Alternative Estimators of Inhomogeneous Pair-Correlation Functions" and equations 3.41 and 3.42 in section 3.1.2.7 "Ripley's K-Function"). These estimators can be used to consider observation windows of irregular shape by using the intensity function introduced above which is zero outside the observation window and λ inside the observation window. This method is used in the standard mode in *Programita*.

You need to tell *Programita* the shape of the observation window of irregular shape. This is done with a file that contains the coordinates of the border of the observation window which must result in a closed line.

The file with the border of the observation window must be an ASCII file with the *.shp extension. **Note that this is not an ArcGis shape file** but an ASCII file with the *.shp extension. In the example of Figure 2.28 it looks like this:

The first line gives the number of points and the following lines are the coordinates of the border. Note that the points must yield a closed curve and that the first and the last points must be the same.

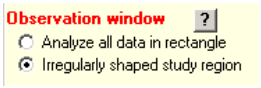
```
 70
  0.0    153.3
  3.4    152.2
 10.8    146.1
 14.2    144.3
 17.1    143.0
 20.6    139.0
 25.6    143.0
 27.7    151.2
 28.3    160.7
 32.0    162.3
 ….
```



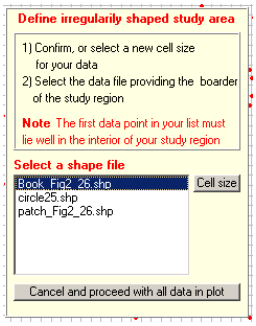
Because *Programita* has only the coordinates of the curve, it needs some information what is inside and what is outside the observation window. To help *Programita* in this task, place a point of the pattern which is located well in the center of the observation window to the beginning of the *.dat data file. *Programita* uses this point as starting point to define the cells that belong to the observation window.

Book_Fig2_28_opt2.res (manipulate estimators, option 2)

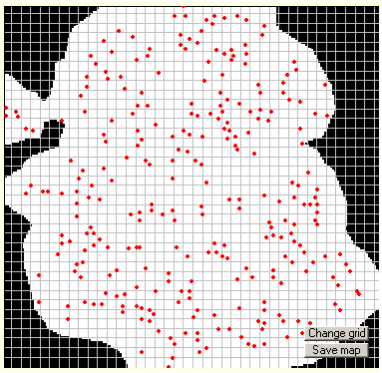
1. Execute *Programita*.
2. Highlight data file Book_Fig2_26.dat you want to analyze in Input data file
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 3 in the menu "Which method will you use"
6. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
7. Press button "**Calculate Index**"
8. Click the radio button "**Irregularly shaped study region**" in the menu "Observation window" on the top left of the interface.



9. Select file Book_Fig2_26.shp, click "**cell size**" and **ok** if the cell size appearing in the window "Select a new cell size" is ok and then the small **ok** button in the Select a shape file window. *Programita* now determines the area of the rectangle that belongs to the observation window. Basically, *Programita* generates an underlying grid with a spatial resolution of one bin (i.e., the cell size) and all cells outside are marked and excluded. *Programita* outputs the resulting intensity file as temporary file "int_temp.int".

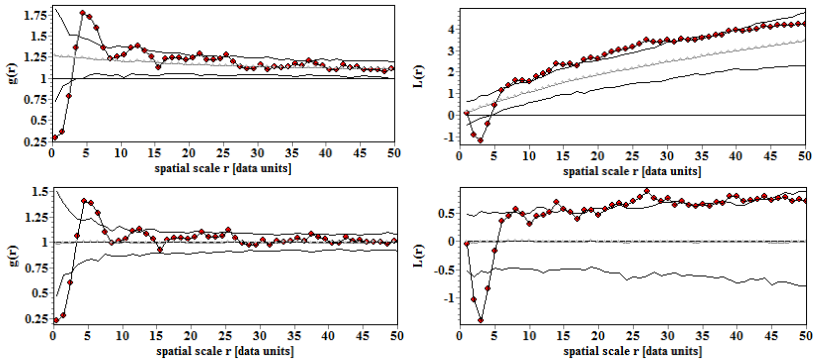


10. Click "**Calculate Index**" and *Programita* shows a plot of the data within the reduced observation window. The excluded area is marked in black.



11. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
12. Select "**Pattern 1 and 2 CSR**" in the window "Select a null model".
13. Specify the number of simulations of the null model (199 in the example) and the rule for the estimation of simulation envelopes (here the 5th lowest and highest values of the summary statistic of the 199 simulated null model data sets).
14. Click the checkbox "**Calculate simulation envelopes** and *Programita* conducts the analysis. You can observe during the simulations that the null model does indeed not distribute points outside the observation window.

15. The results are different from that of the first option because *Programita* considers explicitly the shape of the observation window by estimation of the second-order summary statistics. Top row: previous results (first option). Bottom row: results (second option):



Both, the pair correlation function and the *L*-function are now centered on the expectation of the CSR null model. This is because the estimators consider only the area of the observation window and the edge correction removed the bias seen in the first method that was caused by the heterogeneity of the pattern (i.e., the large patch in the center of the rectangle).

16. You can repeat the entire analysis also for the grid-based standard mode. The only difference is that you need to disable the option "**no grid**" in Which method will you use. As shortcut you can load the results file Book_Fig2_28_opt2.res, disable "**no grid**", and press "**Calculate Index**".

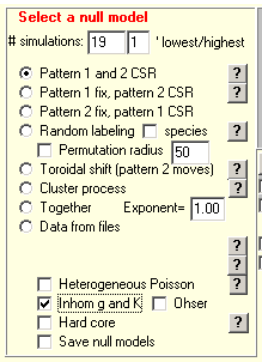
Try also the analogous analysis of Figure 3.48 based on Book_Fig3_48.dat, Book_Fig3_48.shp, and Book_Fig3_48c.res.

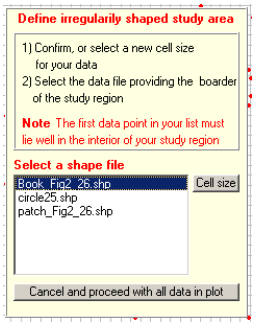
Finally, I show how the third option based on inhomogeneous summary statistics works in the standard mode. Note that the internal estimations of *Programita* are identical to that of the second option. To simplify the procedure for the user and to make it completely analogous to the grid-based mode, I programmed the second option in a way that *Programita* uses inhomogeneous summary statistics.

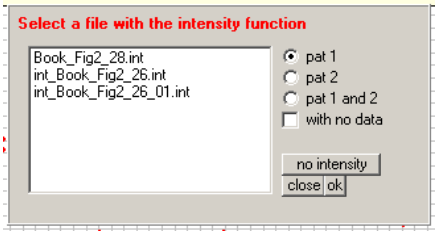
In the second option you need to provide a polygon (i.e., the *.shp file) to define the observation window and *Programita* internally converts the polygon into an intensity function which is zero outside the observation window and λ inside the observation window (this intensity function is saved as temporary file "int_temp.int"). However, in the third option you need to provide the intensity function which is the same as used in the first option to condition the null model.

Example Book_Fig2_28_opt3.res (manipulate estimators, option 3)

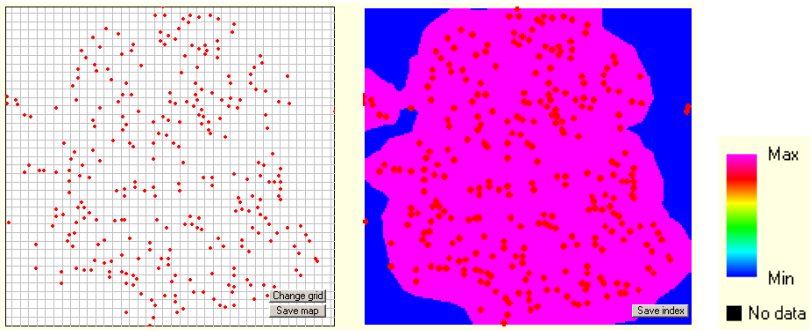
1. Rename temporary file "int_temp.int" into "Book_Fig2_28.int"
2. Execute *Programita*.
3. Highlight data file Book_Fig2_26.dat you want to analyze in Input data file
4. Select "**no grid**" in What do you want to do?
5. Select bin of 1m window Select a new cell size
6. Select a ring width of 3 in the menu "Which method will you use"
7. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
8. Press button "**Calculate Index**"
9. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
10. Select "**Pattern 1 and 2 CSR**" in the window "Select a null model".
11. Specify the number of simulations of the null model (199 in the example) and the rule for the estimation of simulation envelopes (here the 5th lowest and highest values of the summary statistic of the 199 simulated null model data sets).
12. Enable checkbox "**Inhom g and k**", highlight in the appearing window "Select a file with the intensity function" the intensity file **Book_Fig2_28.int**, and click the small **ok** button.



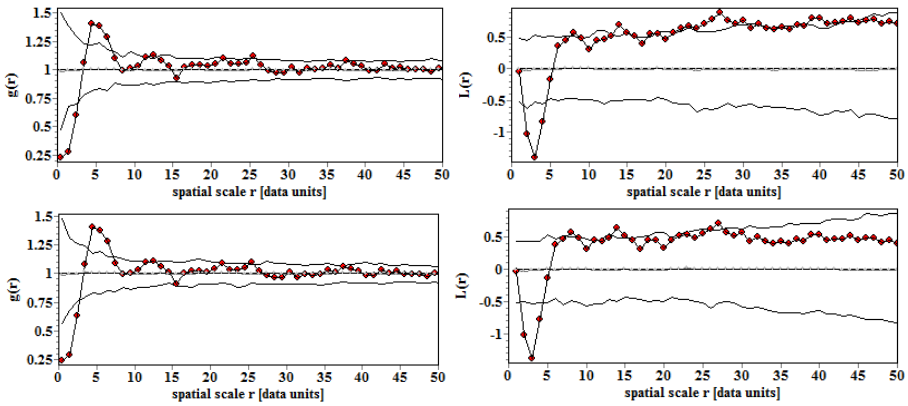




Because we analyze here univariate patterns the radiobox "pat 1" must be selected. This means that the intensity is assigned to pattern 1. *Programita* shows you the data on the left and the intensity function together with the data on the right:



14. Click OK at the message box and then "**Calculate Index**" and *Programita* conducts the analysis. You can observe during the simulations that the null model does indeed not distribute points outside the observation window.
15. The results are virtually identical to that of option 2. Top row: previous results (second option). Bottom row: results (third option):



Both, the pair correlation function and the *L*-function are now centered on the expectation of the CSR null model. This is because the estimators consider only the area of the observation window and the edge correction removed the bias seen in the first method that was caused by the heterogeneity of the pattern (i.e., the large patch in the center of the rectangle).

3.2.5 Null model from file (Book_Fig3_51)

In some cases you may not generate the null model patterns internally with *Programita*, but use patterns generated from other sources for this purpose. One important example for this case is pattern reconstruction (section 4.1.3 "Null Model of Pattern Reconstruction" in Wiegand and Moloney 2014). In this case you can generate from a given pattern statistical replicates that are optimized to closely match several summary statistics of the observed pattern. Of course, the reconstructed patterns will not be an identical copy of the observed pattern, but show the same statistical features as the observed pattern where the typical structures will appear at somewhat displaced locations.

Be sure to use the same estimator for the second-order summary statistics in pattern reconstruction and in *Programita*.

The **first method** of the pattern reconstruction software presented in Wiegand et al. (2013) uses the Ohser edge correction (see equation 3.9 in Wiegand and Moloney 2014) and two times the natural estimator of the intensity $\lambda_n = n/A$:

$$g(r) = \frac{1}{\lambda_n}\frac{1}{\lambda_n}\frac{1}{2\pi r}\sum_{i=1}^{n}\sum_{j=1}^{n,\neq} k(\left\|\mathbf{x}_i - \mathbf{x}_j\right\| - r)[\frac{1}{\bar{\gamma}_W(r)}]$$

This method is used because it corresponds to pattern reconstruction without edge correction for $g(r)$ because the term $A/\bar{\gamma}_W(r)$ does not depend on the points pair i, j but only on distance r and can therefore be factored out in the estimation of the partial energy (equation 3.317 in Wiegand and Moloney 2014).

Method 2 in the pattern reconstruction software corresponds to the WM estimator in *Programita*:

$$g(r) = \frac{1}{\lambda_n}\frac{1}{2\pi r}\sum_{i=1}^{n}\sum_{j=1}^{n,\neq} k(\left\|\mathbf{x}_i - \mathbf{x}_j\right\| - r)[\frac{2\pi r}{\sum_{i=1}^{n} \nu_{d-1}(W \cap \partial b(\mathbf{x}_i, r))}]$$

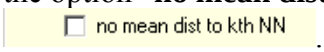
which results from replacing one of the natural estimators λ_n of the intensity in method 1 by the adapted intensity estimate

$$\hat{\lambda}_S(r) = \frac{\sum_{i}^{n} \nu_{d-1}(W \cap \partial b(\mathbf{x}_i, r))}{2\pi r\, \bar{\gamma}_W(r)}$$

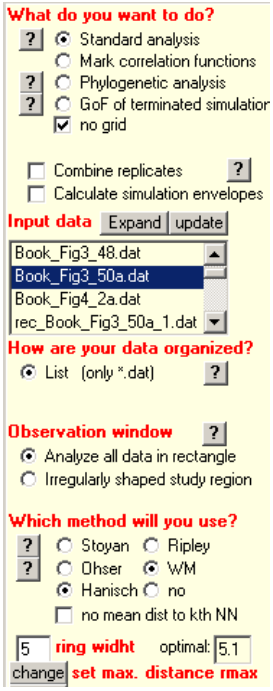
and the **method 3** in the pattern reconstruction software corresponds to the Ohser estimator in *Programita* with adapted intensity estimate where both natural estimators λ_n of the intensity in method 1 are replaced by the adapted intensity estimate $\lambda_S(r)$

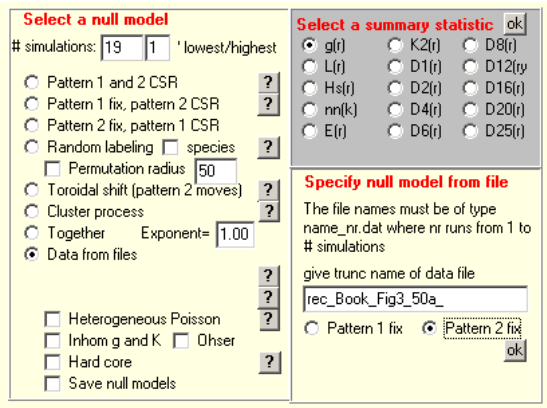
Example Book_Fig3_51.res (null model from file)

1. Execute *Programita*.
2. Highlight data file Book_Fig3_50a.dat you want to analyze in Input data file
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 5 in the menu "Which method will you use"
6. Click button "change" below to set maximal distance *r* to be analyzed. Insert 100 in small box that opens and then the small **ok** button.
7. To view a large range of neighborhood ranks *k* in the mean distance to the kth neighbor summary statistic *nn*(*k*), disable the option "**no mean distance to kth NN**"



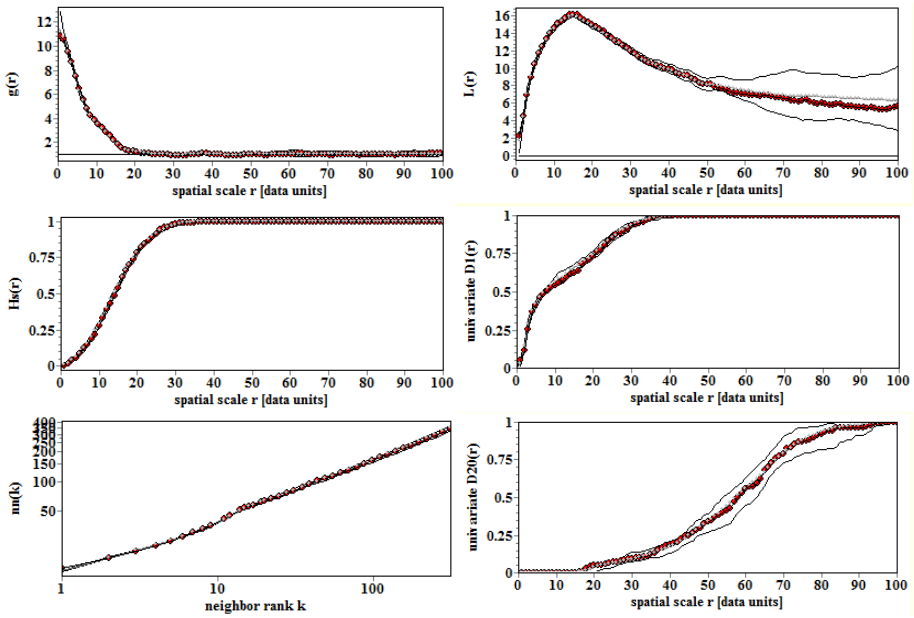
8. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
9. Press button "**Calculate Index**"
10. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
11. Select "**Data from file**" in the window "Select a null model".



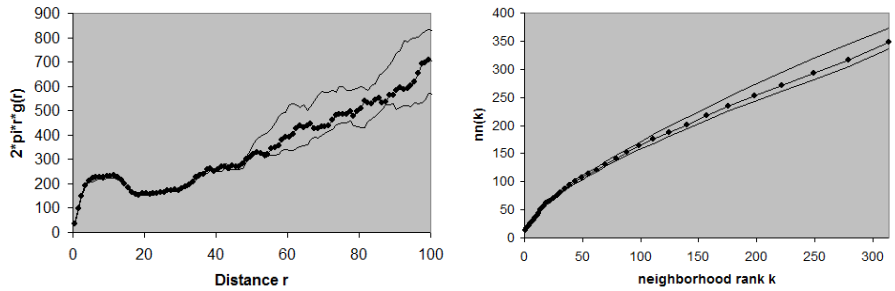


12. Insert the trunk name of the null model files (rec_Book_Fig3_50a_) in the window "Specify null model from file" that opens. This is because your data file had the name "Book_Fig3_50a.dat" and because the pattern reconstruction software names the reconstructions rec_name_n.dat where the "rec_" indicates that this is a reconstructed data file and the n is the number of the reconstructions. Thus you have null model files rec_Book_Fig3_50a_1.dat, rec_Book_Fig3_50a_2.dat, …

13. Click also the radio button "**Pattern 2 fix**". This means that the null model files are used for pattern 1. In bivariate analysis you will typically leave pattern 1 unchanged but replace pattern 2 in the null model by pattern reconstruction files and therefore select "Pattern 1 fix". To finish click the small **ok** button in the window "Specify null model files from file".Specify the number of simulations of the null model (19 in the example) and the rule for the estimation of simulation envelopes (here the 1th lowest and highest values of the summary statistic of the 19 simulated null model data sets).
14. If all settings are specified, press the button "**Calculate Index**" and *Programita* conducts the simulations of the null model.
15. In window "Select a summary statistic" you can view the results of the analysis for the different summary statistics and compare with Figure 3.51 in Wiegand and Moloney (2014):



If you take the results for the pair correlation function from the *.res file and estimate $(2\pi\, r)g(r)$ from of $g(r)$ and plot $(2\pi\, r)g(r)$ over r and $nn(k)$ non-logarithmically you find:



3.2.6 Overview on Thomas cluster processes

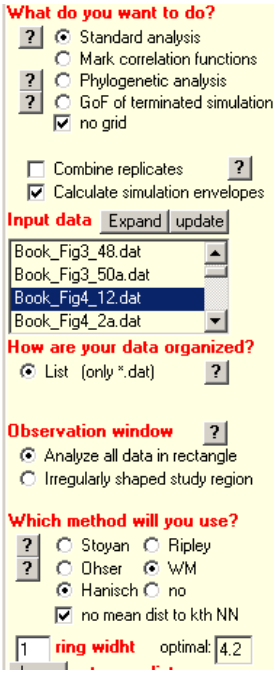
Thomas cluster processes are point process models that describe clustering in a simple way. Details can be found in section 4.1.4 "Poisson Cluster Point Processes" in Wiegand and Moloney (2014). *Programita* allows you to fit several Thomas cluster processes to univariate data:

1. a simple Thomas process with one critical scale of clustering
2. a simple bivariate parent-offspring Thomas process where the cluster centers are known
3. a bivariate parent-offspring Thomas process where the known cluster centers are themselves clustered
4. a generalized Thomas process with two nested scales of clustering where small clusters are located inside large clusters.

Additionally, the cluster processes can be independently superimposed with a CSR pattern.

The cluster processes are parametric point processes and must be fitted to the data. *Programita* allows you to do so in a straight forward way. The procedure to fit a cluster process to the data is described in sections 2.5.2.1 "Minimum Contrast Methods" and 4.1.4.3 "Fitting a Thomas Process to the Data" in Wiegand and Moloney (2014). Especially, *Programita* uses a specific technique to avoid that departures from the point process model at small scales "contaminate" the fit with the cumulative *K*-functions at larger scales. This is an important advance over current techniques and avoids a bias in the fitted parameters. This is exemplified in the first example.

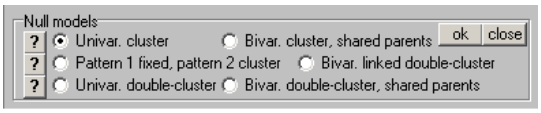
3.2.7 Thomas cluster processes with one scale of clustering (Book_Fig4_12.res)



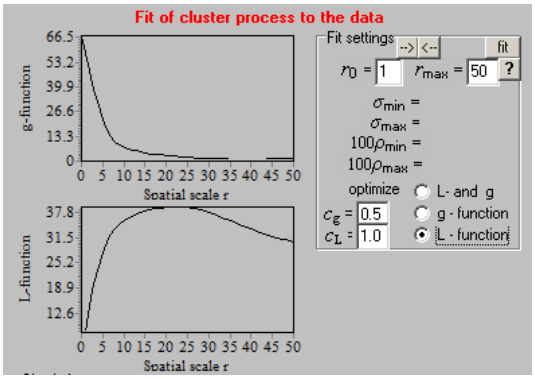
This example illustrates procedure in *Programita* that allows the fitting of a cluster process to a point pattern. It also shows how to deal with patterns that show an additional pattern at small scales not accommodated by the cluster process.

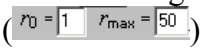
1. Execute *Programita*.
2. Highlight data file Book_Fig4_12.dat you want to analyze in Input data file. This data file was generated with a nested double cluster Thomas process with parameters of large scale clustering being $\sigma_1 = 8.69$ and $A\rho_1 = 68.7$ clusters. The additional small-scale clustering had parameters $\sigma_1 = 2.64$ and $A\rho_1 = 147$ clusters.
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 3 in the menu "Which method will you use"
6. Accept selection of neighborhood ranks of $D^k(r)$.
7. Press button "**Calculate Index**"
8. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left.

9. Select "**Cluster process**" in the window "Select a null model".
10. A window "Fit of cluster process to data" opens. Select in the section "Null models" at the bottom "**Univar. cluster**". This is the simplest Thomas process with one critical scale of clustering. Continue with the small **ok** button

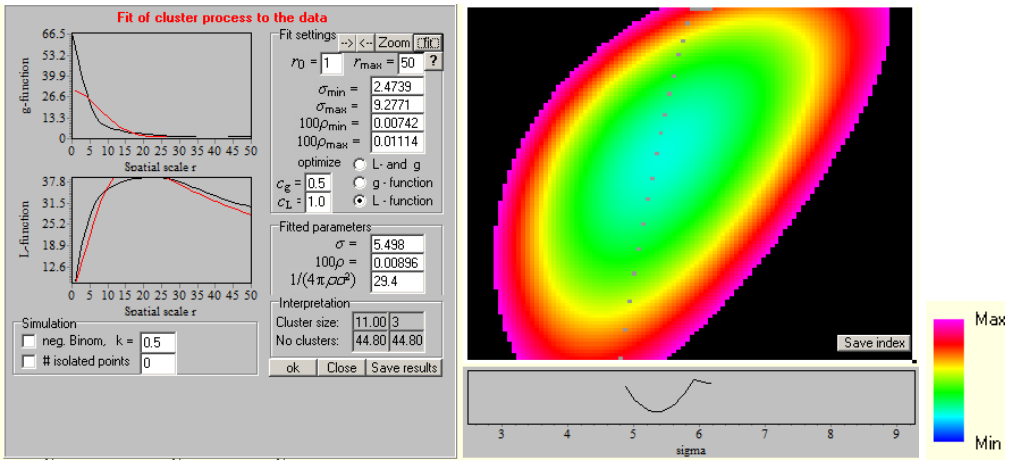


11. Now the interface for fitting appears:



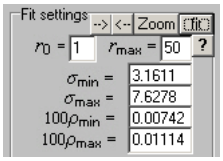
Select the radio button "**L-function**" to only use the *L*-function for fitting. The default settings now fits the L-function over distance interval 1 to 50m ().

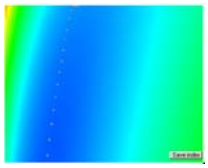
12. Click the button "**fit**" and *Programita* fits the two parameters ρ and σ of the Thomas process to the pattern. Note that ρA yields the number of clusters and 2σ the approximate radius of the "typical cluster". To iteratively encircle the parameter space around the minimum in the σ-ρ parameter space click "**Zoom**" and "**Fit**":



The graph on the right shows the deviation between observed summary statistic (here only the *L*-function) and that predicted by Thomas process over the σ-ρ parameter space indicated by σ_{min}, σ_{max}, $100\rho_{min}$, and $100\rho_{min}$. There is a clear minimum at σ = 5.5 and 44.8 clusters. However, the lower left graph show that the fit of the L-function is not satisfying. This corresponds to Figure 4.12a.

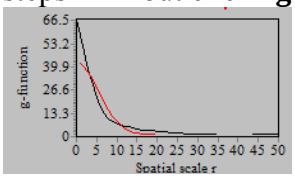
13. Instead of using the **Zoom** option to iteratively encircle the minimum in parameter space you can also manually change the σ-ρ parameter space by selecting appropriate values for σ_{min}, σ_{max}, $100\rho_{min}$, and $100\rho_{min}$:





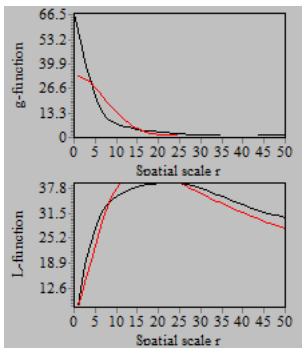
This may be required if the (initially) selected parameter space does not contain the minimum and the fit is poor. In this case a message "increase the maximal value of sigma", "increase the maximal value of roh", "decrease the minimal value of sigma" or "decrease the minimal value of roh" may appear. In this case the observed minimum is located at the edge of the selected parameter space (see right figure above).

14. To use only the pair correlation function for the fit over the 1 - 50m interval repeat steps 1-12 but click "**g - function**". Again, as in Figure 4.12b, the fit is not satisfying:

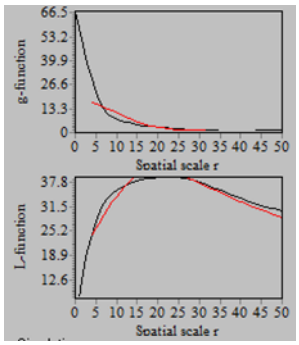


15. The important issue is that you recognize from the shape of the pair correlation function that the pattern contains probably a second critical scale of small-scale clustering. By manipulating the distance r_0 (i.e., the lower limit of the interval of the fit) you can determine the scale of small-scale clustering. For this use both, the pair correlation function and the *L*-function for fitting, i.e., select "**L- and g - function**":

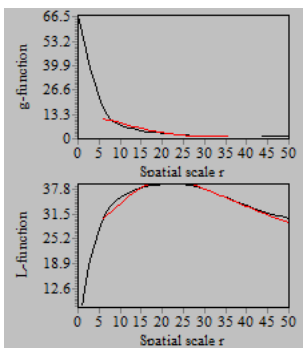
1-50m



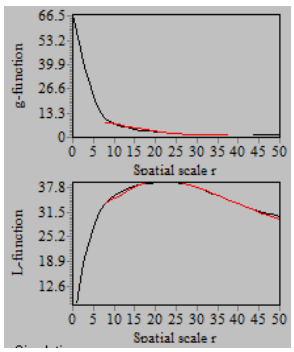
4-50m:



6-50m:

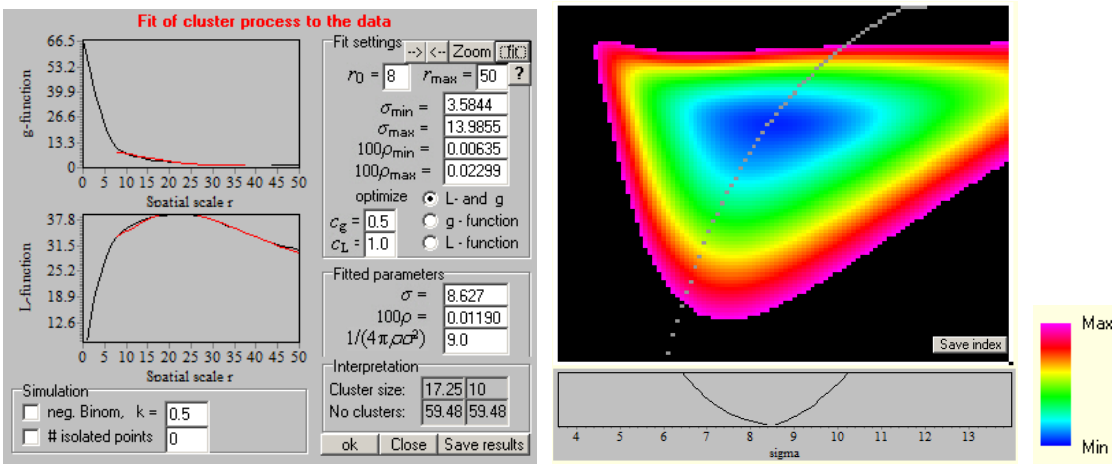


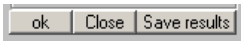
8-50m



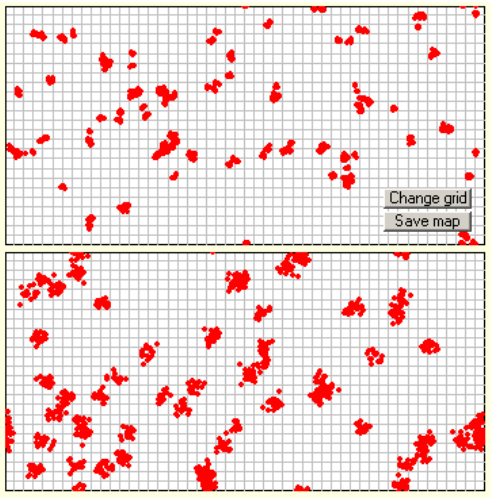
16. The interval of 8-50m provides the best approximation. Thus, the contribution of the additional small-scale clustering to the pair correlation function disappears after 8m. Note that the transformation (equation 4.12) "attaches" the left value of the *L*-function [i.e., $L(r_0)$] to the observed value, thereby removing the "memory" of the *L*-function

Fitting the cluster process only for distances > 8m allows you to determine the parameters of the large-scale clustering. We obtain the parameter estimates σ_1 = 8.6 and 59.6 clusters which are very close to the parameters σ_1 = 8.69 and 68.7 clusters that were used to generate the pattern. The parameter space also indicates a deep and clear minimum (i.e., with dark blue).

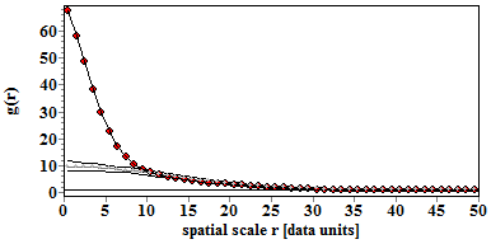


17. If you are satisfied with the fit, click the **ok** button 

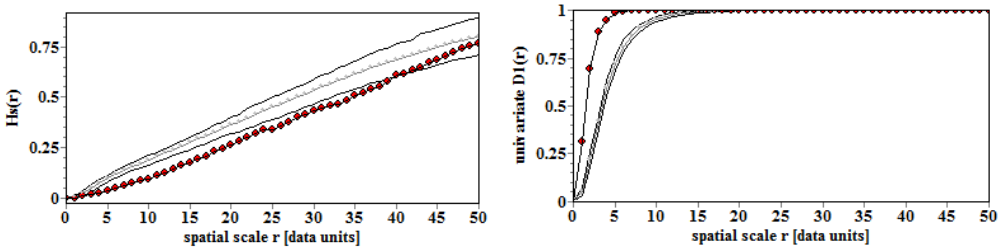
and then "**Calculate Index**". *Programita* now conducts the simulations of the fitted Thomas process:



Here the top graph shows the observed pattern and the bottom graph the simulated patterns. As expected, the fitted point process fits the pair correlation function for distances $r > 8$m very well, but an additional clustering is visible at smaller distances:



The spherical contact distribution is overestimated which means that the gaps in the simulated pattern are too small and the distribution function of the distances to the nearest neighbor is underestimated which means that the nearest neighbor in the simulated patterns are in general too far away.



Clearly, this is because the additional small-scale clustering is missing where 147 small clusters are nested inside the 68.7 large clusters. Thus each large cluster contains on average 2.2 clusters with radius $2\sigma = 5.3$m. For this reason we notice that the observed clusters look somewhat smaller than the simulated ones.

In step 16 you can also save the results of the parameter fitting. In this case a file Book_Fig4_12.fit is generated.

- the first part of the file shows the settings of the *.res file.
- the second part of the file show details on the fit such as the interval in parameter space, the best fitting parameters and their interpretation.
- the third part of the file contains the history of the fitting settings
- the fourth part of the file contains the observed and fitted *g*- and L-*f*unction.
- the final part of the file contains the final parameter space and the associated errors.

```
This file contains settings and results of fitting your data to a cluster1 Thomas cluster process ---------
-----------------------------------------------------------------

Your settings for the point-pattern analysis were:

Pointpattern analysis of file T:\towi\thorsten\text\ManualProgramita2013\Programita\Book_Fig4_12.dat
Method Wiegand-Moloney (ring) with 199 replicates for simulation envelopes, ring width =   3   5 th lowest and highest values of   199 simulations
Test Model= cluster1   8.6270 0.00011900
the null model assumed homogeneous pattern(s)
Analysis modus= points        gridless    WM    NN Hanisch
several points per cell allowed
All cells within the rectangle were considered for calculating the indices
number points of pattern 1 = 1160
number points of pattern 2 =    0
the rectangular area contains 1000*500 = 500000   cells  (= dim1*dim2)
 x-grid-size= 1000 y-grid-size=  500 cell-size =  1.0000 units. rmax=     50,  max distance for NN functions:    279 mean dist to kth neighbor only
partly determined
-----------------------------------------------------------------

 Your settings for the fit of the L-and g-function with a Thomas cluster process were:

 Interval (r0, rmax)  = (8,50)
 Interval for sigma   = (3.5844,13.9855)
 Interval for 100*roh = (0.00635,0.02299)
 The power transformations:
   c = 1.0  for L
   c = 0.5  for g
 The fitted parameters are:
   sigma   = 8.627
   100*roh = 0.01190 which corresponds to 59.48 parents in the study region and to 59.48 parents in the rectangle
 You optimezed the L- and g-function simultaneously
 Only fits with an error <0.02500 are accepted for the estimation of the confidence interval
 The error for the best fit was: 0.00068
 The confidence interval for sigma was:     ( 4.4249,13.9855). This is the interval where the error is <0.02500
 The confidence interval for 100*roh was:   ( 0.0080, 0.0210). This is the interval where the error is <0.02500
```

```
------------------------------------------------------------
 Here is the setting history of your fit
step sigmin   sigmax   100romin   100romax     R0   rmax    power_g    power_L    sigma     100roh     g          error   L_g_Lg
12     3.58    13.99   0.006350   0.022990      8     50     0.5000     1.0000     8.63   0.011897       8.99    0.00068   3
11     3.49    11.65   0.006390   0.019160      8     50     0.5000     1.0000     8.60   0.011937       9.01    0.00069   3
10     3.49    11.65   0.006390   0.019160      6     50     0.5000     1.0000     7.94   0.011034      11.43    0.00208   3
 9     3.33     9.71   0.006340   0.015970      6     50     0.5000     1.0000     7.91   0.011009      11.56    0.00208   3
 8     3.33     9.71   0.006340   0.015970      4     50     0.5000     1.0000     6.81   0.010036      17.09    0.00613   3
…


-------------------------------------------------------------

Here are the data and the fits:
  r  Ldata   Lfit   gdata   gfit
  1   7.7842  33.1815 58.2340   9.9568
  2  14.4385  32.6138 48.5176   9.8670
  3  19.9061  32.3140 38.3178   9.7193
  4  24.3846  32.2546 29.6160   9.5167
  5  27.8025  32.4019 22.5303   9.2631
  6  30.4101  32.7187 16.9736   8.9633
  7  32.3811  33.1676 13.1916   8.6231
  8  33.7127  33.7127 10.4626   8.2486
  9  34.8369  34.3211  8.8404   7.8463
 10  35.6848  34.9637  7.4954   7.4230
 11  36.3997  35.6152  6.6477   6.9856
…

-------------------------------------------------------------
  Here are the best parameters together with the error: (Note that all errors > 0.02500 are set to the value 0.02500)
 sigma  100*roh     error
3.5844  0.00635  0.025000
3.5844  0.00652  0.025000
3.5844  0.00669  0.025000
3.5844  0.00685  0.025000
3.5844  0.00702  0.025000
```

3.2.8 Generalized simple Thomas processes (Book_Fig4_11.res)

This example illustrates the generalization of the simple Thomas process with one scale of clustering that allows for clumping of the number of points over the clusters (see section 4.1.4.2. “Thomas process” in Wiegand and Moloney 2014).

Remember that the simple Thomas process assigned the points randomly to the clusters, thus yielding a Poisson distribution for the distribution p_S of the number of points S per cluster. However, if the distribution p_S follows a more general negative Binominal distribution with clumping parameter k, we can generate more realistic patterns where some clusters have more points than expected by random allocation of the points over the clusters, and others have less than expected. For $k \to \infty$ the negative Binominal distribution collapses to the Poisson distribution.

The nice feature of the Thomas process is that a negative Binominal distribution does not change the functional form of the analytical solution of the pair correlation function (and the K-function) of the Thomas process. We obtain:

$$g(r,\rho,\sigma) = 1 + \frac{f_k}{\rho}\frac{\exp(-r^2/4\sigma^2)}{4\pi\sigma^2}$$

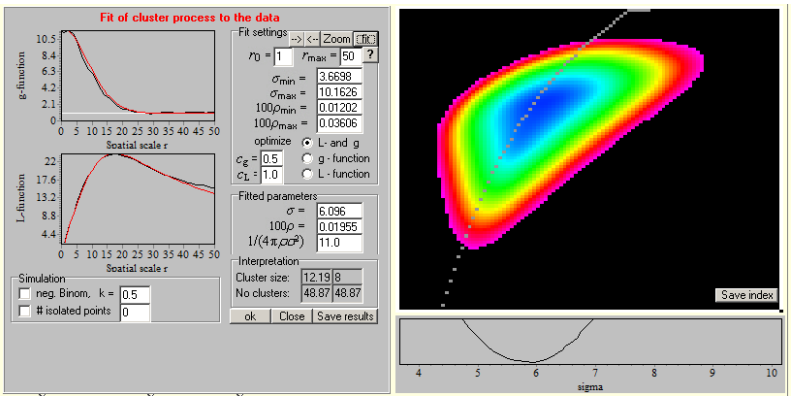
where the factor $f_k = (k + 1)/k$ yields $f_k = 1$ for the simple Thomas process using a Poisson distribution for the distribution p_S. In practical terms this means that clumping of points over the clusters does not change the functional form of the Thomas process, and that we can therefore find for each value of k a simple Thomas process (where p_S is a Poisson distribution) with exactly the same pair correlation (and K-) function. The parameter of this simple Thomas process that describes the number of clusters yields $\rho_S = \rho f_k$ where ρ is the parameter of the generalized Thomas process. That means that clumping of the points over the clusters generates a pair correlation function that seems to have more clusters.

This also means that fitting with second order properties alone does not allow us to determine the parameter k of the generalized Thomas process. However, other summary statistics of different nature such as the spherical contact distribution $H_s(r)$ and the distribution function $D^k(r)$ of the distances to the kth neighbor may allow us to approximate the value of k. Thus, we need to fit first with the pair correlation and the L-function and then simulate several generalized Thomas process with different values of the clumping parameter k, but adjust the number of clusters of the simulated generalized Thomas process in a way that the pair correlation function does not change (i.e., $\rho_S = \rho f_k$). We then can check the fit of $H_s(r)$ and $D(r)$ for different values of k and indirectly determine the value that fits $H_s(r)$ and $D^k(r)$ best.

Example Book_Fig4_11e_k=1000.res (fit parameters σ, ρ, and k)

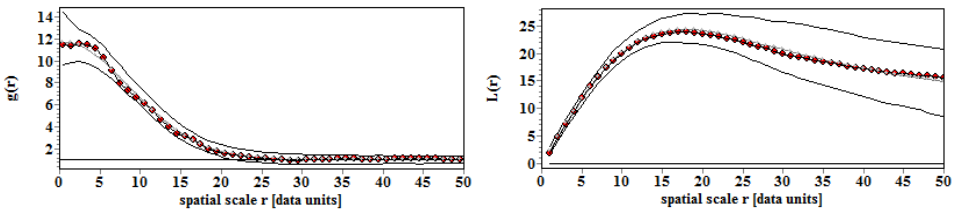
This example file was generated with a generalized Thomas process with parameters σ =6, $A\rho$ = 50, and k = 1 (f_k = 2).

1. Execute *Programita*.
2. Highlight data file Book_Fig4_11e.dat you want to analyze in Input data file.
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 3 in the menu "Which method will you use"
6. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
7. Press button "**Calculate Index**"
8. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
9. Select "**Cluster process**" in the window "Select a null model".
10. A window "Fit of cluster process to data" opens. Select in the section "Null models" at the bottom "**Univar. cluster**". Continue with the small **ok** button.
11. Fit the parameters σ, and ρ. As expected, you obtain a good fit (σ =6.1, $A\rho$ = 49 in the example below).

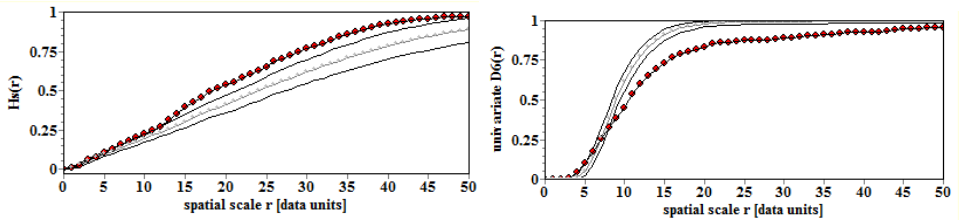


If you now click the "**ok**"  and the "**Calculate Index**" you simulate the simple Thomas process (i.e., $k = \infty$).

12. As expected, the pair correlation and L- function are perfectly fitted:

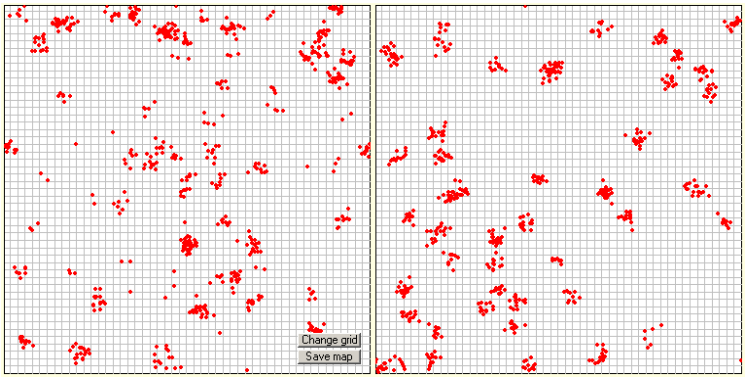


but the spherical contact distribution $H_s(r)$ and the distribution function $D^k(r)$ of the distances to the kth neighbor (here the 6th neighbor) not:



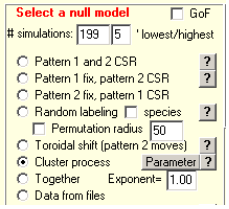
The spherical contact distribution is underestimated which means that the gaps in the simulated pattern are too large and the distribution function of the distances to the 6th neighbor is overestimated which means that the 6th neighbors in the simulated patterns are in general too close.

When looking at the observed pattern and a realization of the simple Thomas process we detect only subtle differences, the observed pattern has somewhat more scattered points and the simulated pattern has more well delineated clusters with larger gaps:

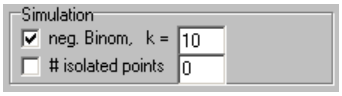


Example Book_Fig4_11e_k=10.res (fit parameters σ, ρ, and k)

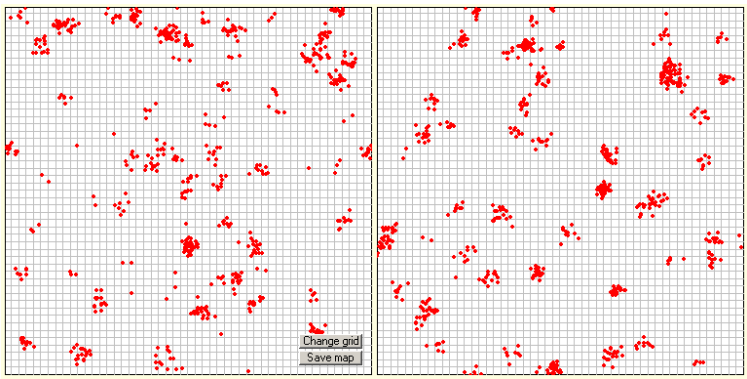
This example continues the above example Book_Fig4_11e_k=1000.res, but now simulates a generalized Thomas process with parameter $k = 10$.



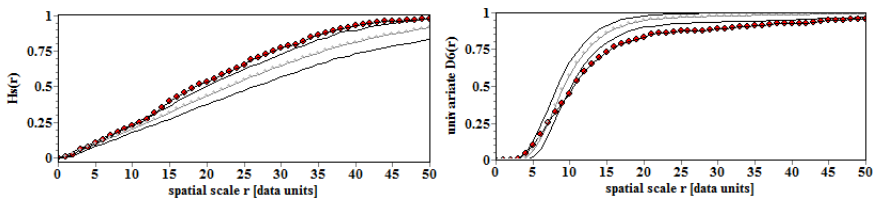
1. The simulation of the simple Thomas process is terminated. To access the menu of the Thomas process click the button "**Parameters**" in the window "Select a null model".
2. To select a value of $k = 10$ that corresponds to $f_k = 1.1$, click the check box "**neg. Binom**" and write 10 in the corresponding text box:



3. If you now click the "**ok**"  and the "**Calculate Index**" you simulate the generalized Thomas process with $k = 10$ that fits the observed pair correlation and L-function. The simulated patterns now resemble the observed one better but there are still too large gaps:

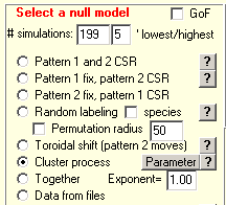


and the spherical contact distribution $H_s(r)$ and the distribution function $D^k(r)$ of the distances to the kth neighbor (here the 6th neighbor) are only slightly better fitted than before:

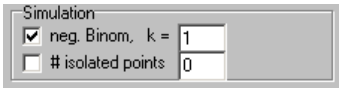


Example Book_Fig4_11e_k=1.res (fit parameters σ, ρ, and k)

This example continues the above example Book_Fig4_11e_k=1000.res, but now simulates a generalized Thomas process with parameter $k = 1$ (the parameter used to generate the pattern).

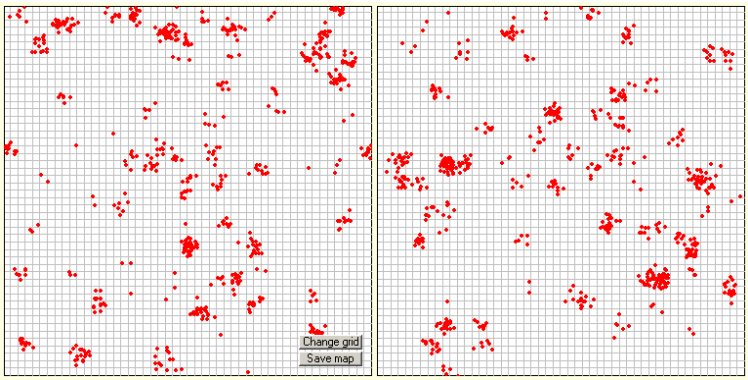


1. The simulation of the simple Thomas process is terminated. To access the menu of the Thomas process click the button "**Parameters**" in the window "Select a null model".
2. To select a value of $k = 1$ that corresponds to $f_k = 2$, click the check box "**neg. Binom**" and write 1 in the corresponding text box:

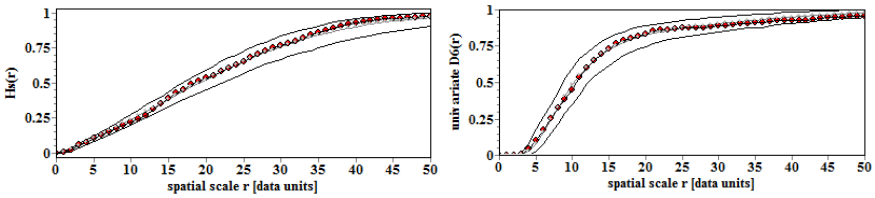


3. If you now click the "**ok**" 

and the "**Calculate Index**" you simulate the generalized Thomas process with $k = 1$ that fits the observed pair correlation and L-function. The simulated patterns now resemble the observed one very well:

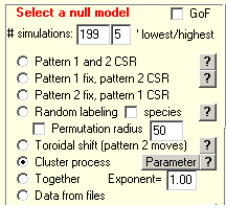


and the spherical contact distribution $H_s(r)$ and the distribution function $D^k(r)$ of the distances to the kth neighbor (here the 6th neighbor) are also well fitted:

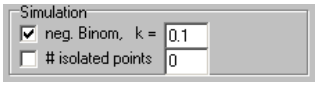


Example Book_Fig4_11e_k=01.res (fit parameters σ, ρ, and k)

This example continues the above example Book_Fig4_11e_k=1000.res, but now simulates a generalized Thomas process with parameter $k = 0.1$.

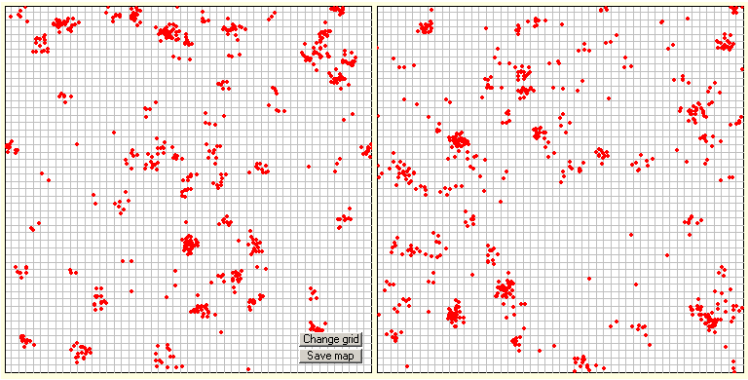


1. The simulation of the simple Thomas process is terminated. To access the menu of the Thomas process click the button "**Parameters**" in the window "Select a null model".
2. To select a value of $k = 0.1$ that corresponds to $f_k = 11$, click check box "**neg. Binom**" and write 1 in the corresponding text box:

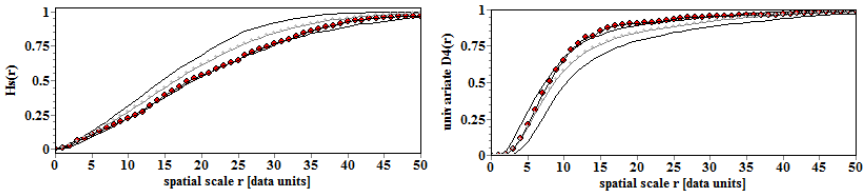


3. If you now click the "**ok**" 

and the "**Calculate Index**" you simulate the generalized Thomas process with $k = 0.1$ that fits the observed pair correlation and L-function. The simulated patterns now show a somewhat to dispersed pattern with too much scattered points:



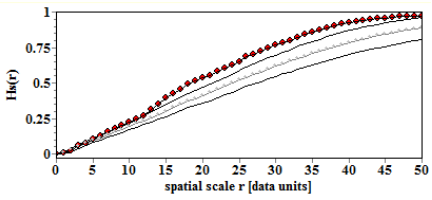
and the spherical contact distribution $H_s(r)$ and the distribution function $D^k(r)$ of the distances to the kth neighbor (here the 4th neighbor) are not well fitted:

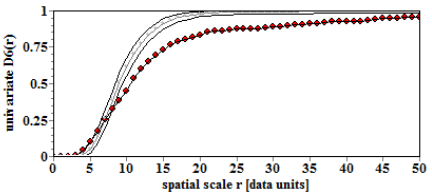


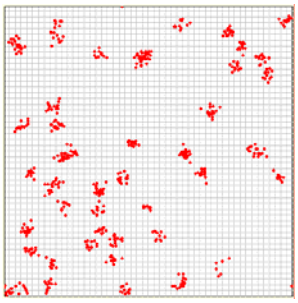
the simulated pattern now has too small gaps and the 4th neighbor is too far.

Summarizing the fits with the $D^k(r)$ we see that the spherical contact distribution $H_s(r)$ and the distribution function $D^k(r)$ of the distances to the kth neighbor (i.e., the simulation envelopes) change systematically with changing clumping parameter k:

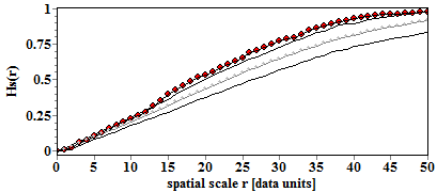
$k = \infty$:

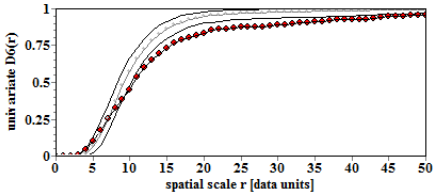


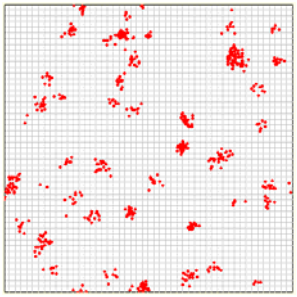




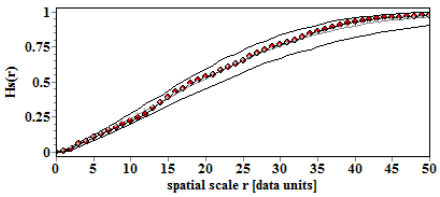
$k = 10$:

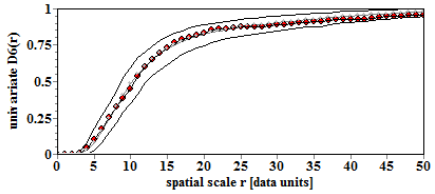


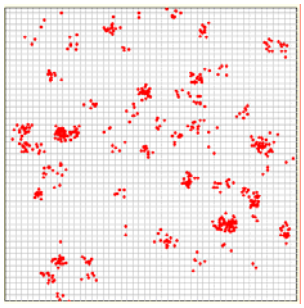




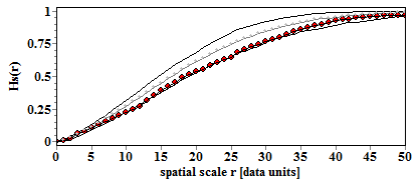
$k = 1$:

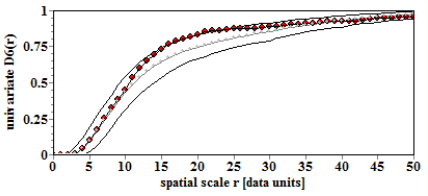


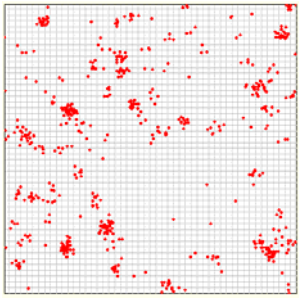




$k = 0.1$:







3.2.9 Superposition of CSR and a simple Thomas process (Fig4_11CSR0.res)

This example illustrates a second generalization of the simple Thomas process with one scale of clustering that allows description of a clustered pattern with more "isolated" points than expected by the simple Thomas process.

The first option to accomplish this was using the negative Binominal distribution p_S to describe the number of points S per cluster. Clearly, if k is relatively small (e.g., $k = 1$ or 0.1) there will be much more clusters with only one point (i.e., isolated points) than expected by the simple Thomas process (see e.g., insets in Figures 4.11b, f, and j). For example, the generalized Thomas process with $k = 0.1$ yields 550 clusters, 422 are empty (i.e., $S = 0$) and 42 have only one point (= 32% of all clusters with at least one point) while the corresponding simple Thomas process does not show isolated points (see Fig. 4.11b).

The second option is an independent superposition of a simple Thomas process with a CSR pattern (for details see sections 3.3 and 3.3.5 "Examples of the Superposition of a Thomas Process with a Random Pattern"). The nice feature of the Thomas process is the superposition with CSR does not change the functional form of the analytical solution of the pair correlation function (and the K-function) of the Thomas process. We obtain:

$$g(r,\rho,\sigma) = 1 + \frac{c^2}{\rho}\frac{\exp(-r^2/4\sigma^2)}{4\pi\sigma^2}$$

where the parameter c is the proportion of points of the pattern belonging to the Thomas process. In practical terms this means that superposition with CSR does not change the functional form of the Thomas process, and that we can therefore find for each value of c a simple Thomas process (where $c = 1$) with exactly the same pair correlation (and K-) function. The parameter of this simple Thomas process that describes the number of clusters yields $\rho_S = \rho/c^2$ where ρ is the parameter of the generalized Thomas process. That means that superposition with CSR generates a pair correlation function that seems to have more clusters.

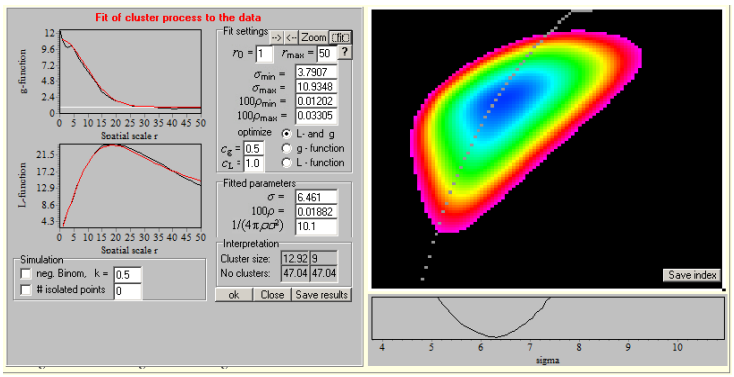
This also means that fitting with second order properties alone does not allow us to determine the proportion (1- c) of isolated CSR points. However, other summary statistics of different nature such as the spherical contact distribution $H_s(r)$ and the distribution function $D^k(r)$ of the distances to the kth neighbor may allow us to approximate the value of c. Thus, we need to fit first with the pair correlation and the L-function to obtain estimates of σ and ρ and then simulate several generalized Thomas processes with different proportions 1- c of random points, but adjust the number of clusters of the simulated generalized Thomas processes in a way that the pair correlation function does not change (i.e., $\rho_S = \rho/c^2$). We then can check the fit of $H_s(r)$ and $D(r)$ for different values of c and indirectly determine the value that fits $H_s(r)$ and $D^k(r)$ best.

Note that you cannot simulate an independent superposition of a random pattern with a generalized Thomas process with a negative Binominal distribution. Such a process is not implemented in *Programita* and it would be difficult to determine the parameters with any confidence.

Example Fig4_11CSR0.res (fit parameters σ, ρ, and c)

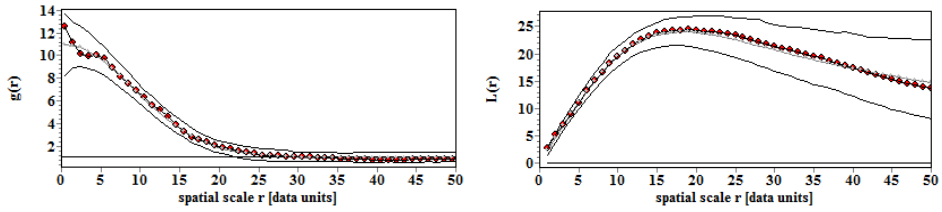
This example file was generated with a generalized Thomas process with parameters σ =6, $A\rho$ = 50, and k = 1 (c = 0.84; 100 isolated points).

1. Execute *Programita*.
2. Highlight data file Fig4_11csr100.dat you want to analyze in Input data file.
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 3 in the menu "Which method will you use"
6. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
7. Press button "**Calculate Index**"
8. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
9. Select "**Cluster process**" in the window "Select a null model".
10. A window "Fit of cluster process to data" opens. Select in the section "Null models" at the bottom "**Univar. cluster**". Continue with the small **ok** button.
11. Fit the parameters σ, and ρ. As expected, you obtain a good fit (σ =5.4, $A\rho$ = 47 in the example below).

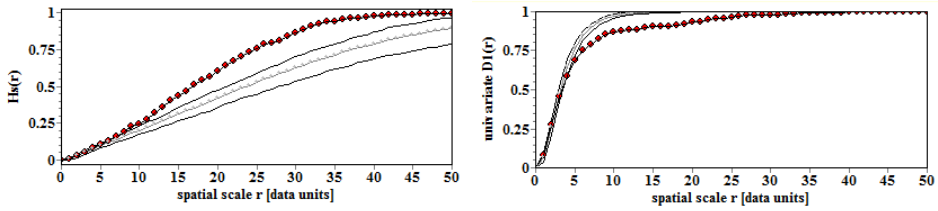


If you now click the "**ok**"  and the "**Calculate Index**" you simulate the simple Thomas process (i.e., $k = \infty$, c = 1).

12. As expected, the pair correlation and L- function are perfectly fitted:

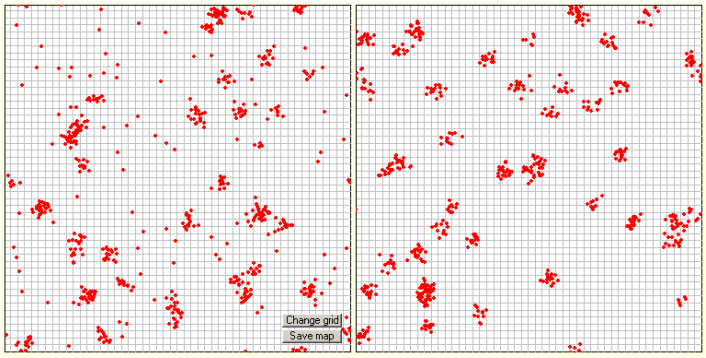


but the spherical contact distribution $H_s(r)$ and the distribution function $D^k(r)$ of the distances to the kth neighbor (here the 1th neighbor) not:



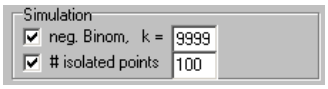
The spherical contact distribution is underestimated which means that the gaps in the simulated pattern are too large (clearly the interspersed random points are missing) and the distribution function of the distances to the nearest neighbor is overestimated which means that the nearest neighbors in the simulated patterns are in general too close. Clearly, the isolated points of the CSR component have their nearest neighbor not within a cluster but usually much farther away.

When looking at the observed pattern and a realization of the simple Thomas process we detect only subtle differences, the observed pattern has somewhat more isolated points:



This example continues the above example Fig4_11CSR0.res, but now simulates a generalized Thomas process with the parameter $c = 0.84$ (i.e., 100 isolated points) that were used to generate it.

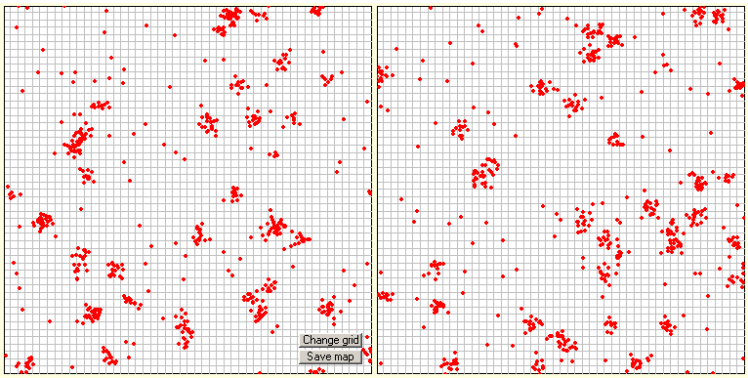
1. The simulation of the simple Thomas process is terminated. To access the menu of the Thomas process click the button "**Parameters**" in the window "Select a null model".
2. To select 100 random points click the checkbox "**neg. Binom**" and "**# isolated points**" and write 100 in the corresponding text box:



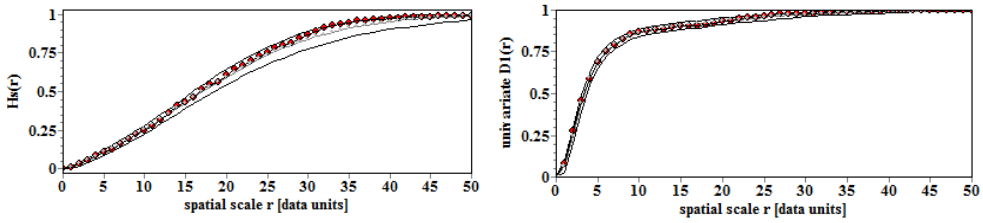
A value of $k = 9999$ appears automatically to represent a Poisson distribution for p_S. If you would select a different value for k it will be reset to 9999 since this more complex combination of processes is not implemented in *Programita*.

3. If you now click the "**ok**" 

and the "**Calculate Index**" you simulate the simple Thomas process with 526 points, independently superposed with 100 random points that fits the observed pair correlation and L-function. The simulated patterns now resemble the observed one well:



and the spherical contact distribution $H_s(r)$ and the distribution function $D^k(r)$ of the distances to the kth neighbor (here the 1th neighbor) fitted well:



3.2.10 Simple bivariate parent-offspring Thomas process (Book_Fig4_13.res)

This example illustrates use of a Thomas process where the cluster centers are known. In this case we have two patterns, pattern 1 are the cluster centers and pattern 2 the pattern that is assumed to follow a simple Thomas process.

Programita offers two point process models to analyze this data type:

- In the first case the bivariate second-order statistics between the points (pattern 2) and the cluster centers (pattern 1) is fitted.
- In the second case the standard Thomas process is used to fit the univariate pattern 2, but the simulation of the point process uses the known cluster centers (pattern 1).

The bivariate pair correlation functions of the bivariate pattern yields (equation 4.13 in Wiegand and Moloney 2014):

$$g_{12}(r,\sigma_2,\lambda_1) = 1 + \frac{1}{\lambda_1}\frac{\exp(-r^2/2\sigma_2^2)}{2\pi\sigma_2^2}$$

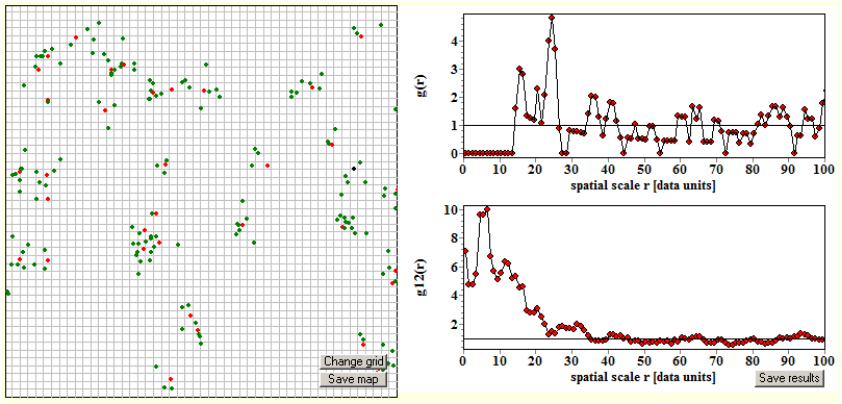
where $\lambda_1 = \rho_2$ is the intensity of pattern 1 (i.e., the cluster centers).

Programita fits the parameters λ_1 and σ_2 of this point process and then simulates the univariate Thomas process with known parents.

Example Book_Fig4_13_bi.res

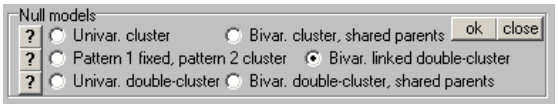
This example file was generated with a simple Thomas process with 157 points and parameters σ_2 = 13.3 and A λ_1 = 35 random clusters.

1. Execute *Programita*.
2. Highlight data file Book_Fig4_13.dat you want to analyze in Input data file.
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 3 in the menu "Which method will you use"
6. Click button "change" below to set maximal distance r to be analyzed. Insert 100 in small box that opens and then the small **ok** button.
7. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
8. Press button "**Calculate Index**"
9. *Programita* then shows the pattern, the univariate pair correlation function of the cluster centers and the bivariate pair correlation function of the points around their parents:

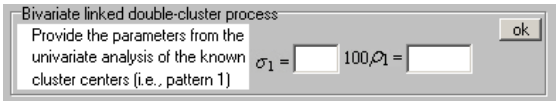


The pair correlation function of pattern 1, which is not of interest here, is somewhat rugged because pattern 1 has few points (i.e., 34 clusters).

10. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
11. Select "**Cluster process**" in the window "Select a null model".
12. A window "Fit of cluster process to data" opens. Select in the section "Null models" at the bottom the button "**Bivar. linked double-cluster**".



Now this window appears that asks you to provide the parameters from the univariate analysis of pattern 1 (i.e., the cluster centers):



This is because the implementation of *Programita* uses here the equation for the more general point process where the cluster centers follow a simple Thomas process. See section 4.1.4.4 "Bivariate Parent–Offspring Thomas Processes" in Wiegand and Moloney (2014). The equation used for fitting is equation 4.14:

$$g_{12}(r,\rho_1,\sigma_1,\rho_2,\sigma_2) = \frac{1}{\rho_2}h(r,\sigma_2)+\frac{1}{\rho_1}h(r,\sqrt{2\sigma_1^2+\sigma_2^2})$$

$$= 1+\frac{1}{\rho_2}\frac{\exp(-r^2/(2\sigma_2^2))}{2\pi\sigma_2^2}+\frac{1}{\rho_1}\frac{\exp(-r^2/(4\sigma_1^2+2\sigma_2^2))}{4\sigma_1^2+2\sigma_2^2}$$

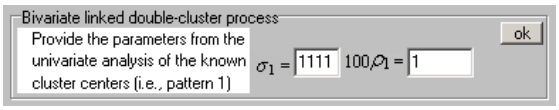
Because pattern 1 is in the example pattern "Book_Fig4_13.dat" a random pattern, you can recover the equation of the more simpler point process where the cluster centers following CSR, i.e.,

$$g_{12}(r,\sigma_2,\lambda_1) = 1 + \frac{1}{\lambda_1}\frac{\exp(-r^2/2\sigma_2^2)}{2\pi\sigma_2^2}$$

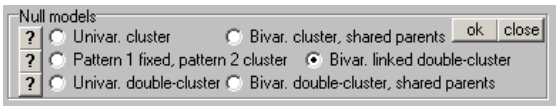
by selecting parameters σ_1 and ρ_1 that correspond to CSR for pattern 1, e.g., σ_1 = 1111 and $100\rho_1$ = 1. In this case the second term

$$\frac{1}{\rho_1}h(r,\sqrt{2\sigma_1^2+\sigma_2^2})$$

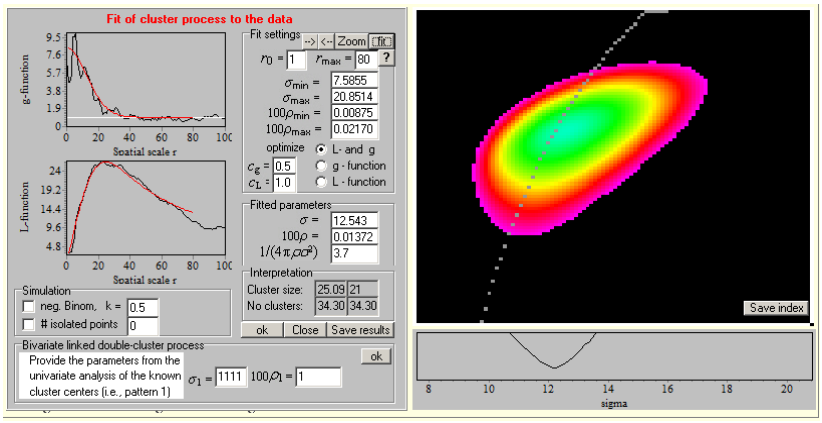
approximates zero. Thus, insert σ_1 = 1111 and $100\rho_1$ = 1



and click the small "**ok**" button and then again the small "**ok**" button in the next window:



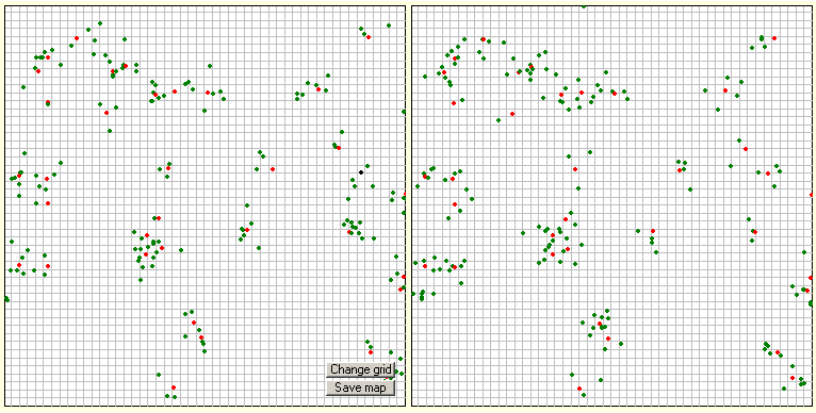
13. Fit the parameters σ, and ρ over the distance 1 to 80:



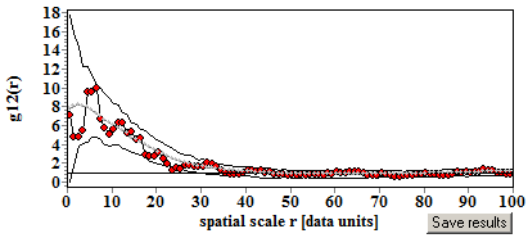
As indicated by the *L*-function, the fit is good and even the fitted number of cluster centers (34.3) coincides well with the known number (i.e., 34).

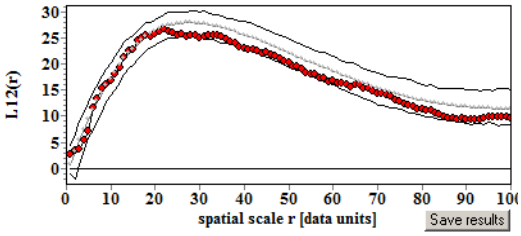
If the fitted number of cluster centers is larger than the number of points of pattern 1 (here 34), a warning appears and the known number of points of pattern 1 is used instead of the fitted parameter $A\rho_2$ for the number of cluster centers. If the fitted number of clusters is smaller than the known number of points of pattern 1 a reduced number of cluster centers is randomly selected among the points of pattern 1.

Click the small "**ok**" buttons and the "**Calculate Index**". *Programita* now simulates the pattern 2 of the fitted point process using the locations of pattern 1 as cluster centers:

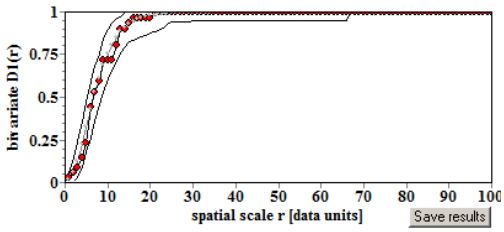


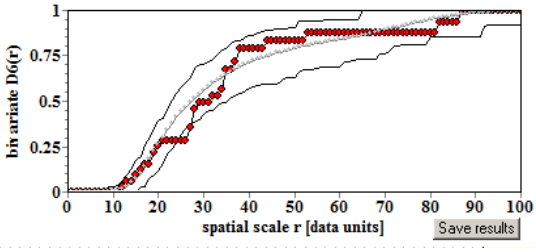
14. The pair correlation and *L*- function are well fitted:



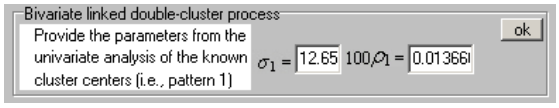


and the distribution functions $D^k(r)$ of the distances to the kth neighbors as well (here the 1th and 6th neighbor):





Note that fitting the bivariate parent-offspring Thomas process where the known cluster centers (i.e., pattern 1) are themselves clustered follows exactly the same procedure. The only difference is that you need to provide in step 14 the parameters fitted to the clustered pattern 1. Test for example with the data of file Book_Fig4_14.dat:

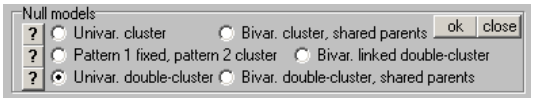


Note also that the bivariate parent-offspring Thomas processes with known parents are somewhat sensitive to the assumption that the parents follow CSR or a simple Thomas process. If this assumption is not well met, you may better use the equivalent Cox process presented in example Book_Fig4_18.res below which makes no assumptions on the pattern of the cluster centers.

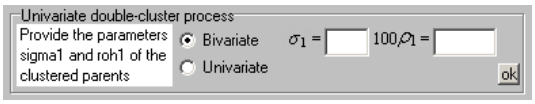
Example Book_Fig4_13_uni.res

This example analyzes the same data set as example Book_Fig4_13_bi.res, but with a different method. In this case pattern 2 is fitted to a simple Thomas process and then in the simulation of the fitted point process the points of pattern 1 are used as cluster centers.

1. Execute *Programita*.
2. Highlight data file Book_Fig4_13.dat you want to analyze in Input data file.
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 3 in the menu "Which method will you use"
6. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
7. Press button "**Calculate Index**"
8. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
9. Select "**Cluster process**" in the window "Select a null model".
10. A window "Fit of cluster process to data" opens. Select in the section "Null models" at the bottom the button "**Univar. double cluster**".

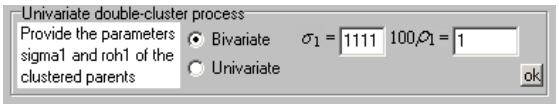


Now the below window appears. Select radio button "Bivariate" because you use the points of pattern 1 as cluster centers. Provide the parameters from the univariate analysis of pattern 1 (i.e., the cluster centers):

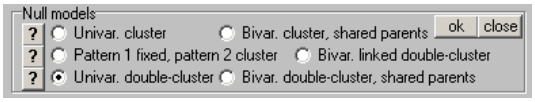


Again, this is because the implementation of *Programita* uses here the equation for the more general point process where the cluster centers follow a simple Thomas process. See section 4.1.4.5 "Generalized Thomas Process with Two Nested Scales of Clustering" in Wiegand and Moloney (2014). The equation used for fitting is equation 4.17.

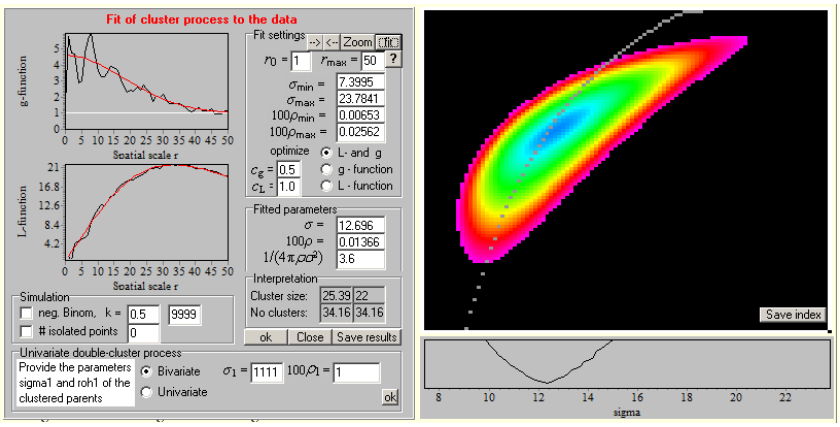
Because pattern 1 is in the example pattern "Book_Fig4_13.dat" a random pattern, you can, in the same way as in the previous example, recover the equation of the simple Thomas process where the cluster centers following CSR by selecting parameters σ_1 and ρ_1 that correspond to CSR for pattern 1, e.g., $\sigma_1 = 1111$ and $100\rho_1 = 1$. Thus, insert $\sigma_1 = 1111$ and $100\rho_1 = 1$



and click the small "**ok**" button and then again the small "**ok**" button in the next window.



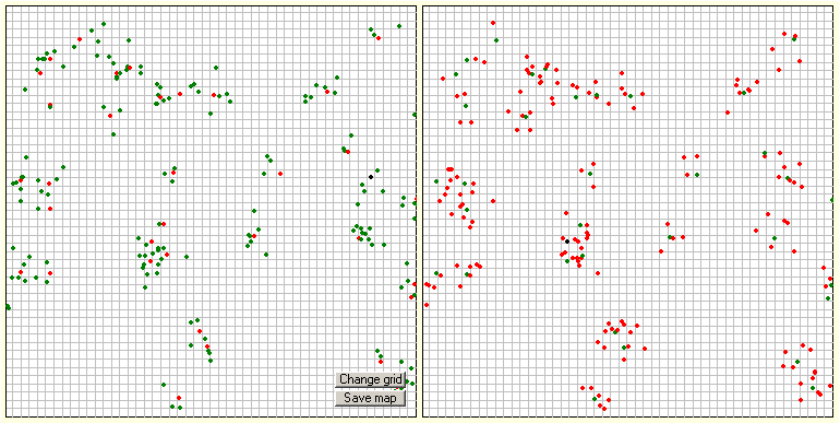
11. Fit the parameters σ, and ρ over the distance 1 to 50:



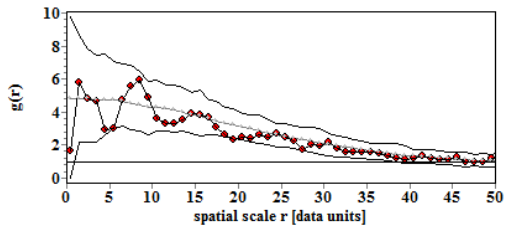
As indicated by the *L*-function, the fit is good and even the fitted number of cluster centers (34.2) coincides well with the known number (i.e., 34).

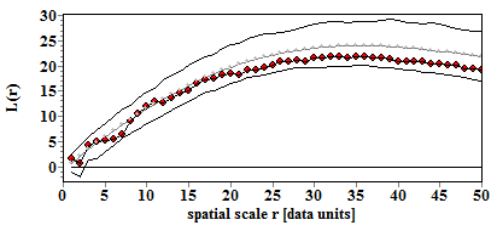
If the fitted number of cluster centers is larger than the number of points of pattern 1 (here 34), a warning appears and the known number of points of pattern 1 is used instead of the fitted parameter $A\rho_2$ for the number of cluster centers. If the fitted number of clusters is smaller than the known number of points of pattern 1 a reduced number of cluster centers is randomly selected among the points of pattern 1.

12. Click the small "**ok**" buttons and the "**Calculate Index**". *Programita* now simulates the pattern 2 of the fitted point process using the locations of pattern 1 as cluster centers (note that the colors are exchanged in the simulated pattern):

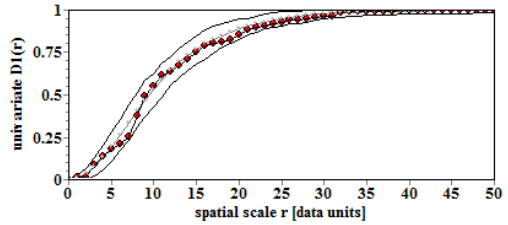


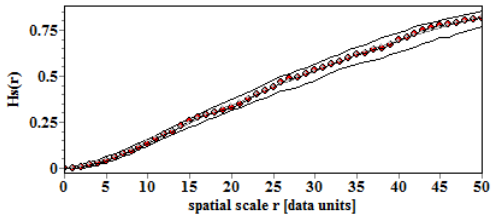
The pair correlation and *L*- function are well fitted:



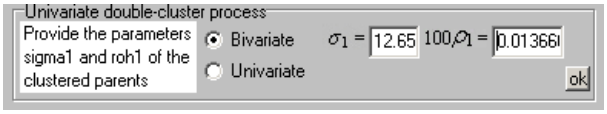


and the distribution functions $D^k(r)$ of the distances to the kth neighbors (here the 1th neighbor) and the spherical contact distribution as well:





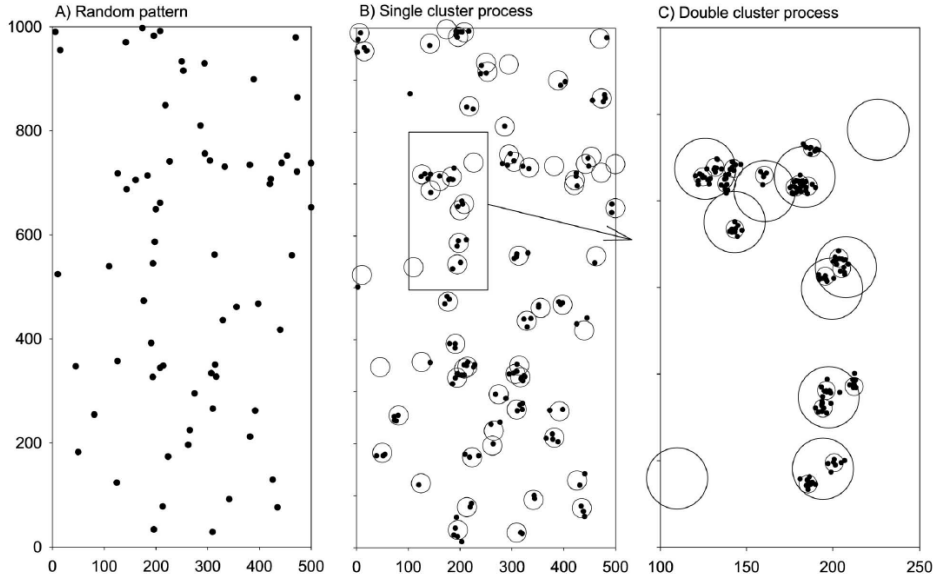
Note that fitting the bivariate parent-offspring Thomas process where the known cluster centers (i.e., pattern 1) are themselves clustered follows exactly the same procedure. The only difference is that you need to provide in step 12 the parameters fitted to the clustered pattern 1:



Test for example with the data file Book_Fig4_14.dat

3.2.11 Thomas process with two nested scales of clustering

This point process is a generalization of the (generalized) simple Thomas process where the pattern of cluster centers does not follow CSR (as in the generalized simple Thomas process), but can itself be a (generalized) simple Thomas process. This leads to clusters inside clusters and a nested cluster structure as shown below:



Details on this point process model can be found in section 4.1.4.5 "Generalized Thomas Process with Two Nested Scales of Clustering" in Wiegand and Moloney (2014).

The pair correlation function of this point process is shown in equation 4.17:

$$g_{22}(r,\rho_1,\sigma_1,\rho_2,\sigma_2) = 1 + \underbrace{\frac{f_{k2}}{\rho_2} h(r,\sqrt{2}\sigma_2)}_{\text{small-scale clustering}} + \underbrace{\frac{f_{k1}}{\rho_1} h(r,\sqrt{2}\sigma_{sum})}_{\text{combined effect of small- and large-scale clustering}}$$

where $h(r,\sigma) = \dfrac{\exp(-0.5r^2/\sigma^2)}{2\pi\sigma^2}$ and $\sigma_{sum}^2 = \sigma_1^2 + \sigma_2^2$. This point process has 6 parameters:

- σ_2: the parameter that describes the size of the small clusters
- ρ_2: the intensity of the small clusters
- $f_{k2} = (k_2 + 1)/k_2$ where k_2 is the parameter of the negative Binominal distribution that governs the distribution of points over the small clusters
- σ_1: the parameter that describes the size of the large clusters
- ρ_1: the intensity of the large clusters
- $f_{k1} = (k_1 + 1)/k_1$ where k_1 is the parameter of the negative Binominal distribution that governs the distribution of the number of small clusters over the large clusters

The above equation for the pair correlation function of the double cluster process is composed of three summands. The first summand (1) is the contribution which remains if the two other summands disappears when the pattern is a CSR pattern. The second summand is only due to the small-scale clustering, and the third summand contains an interaction between small- and large-scale clustering, indicated by σ_{sum}.

As we saw before when introducing the generalized simple Thomas process, the parameters k_1 and k_2 cannot be fit with the *g*- or *L*-function but needs to be indirectly determined by comparing the resulting fit of other summary statistics such as the spherical contact distribution and the distribution function of the distances to the kth neighbor. Thus, we are left with the task to fit four unknown parameters to the second-order summary statistics.

Programita uses a simple two-step procedure which is based on separation of scales. If the two critical scales of clustering are well separated, i.e., $\sigma_2 << \sigma_1$

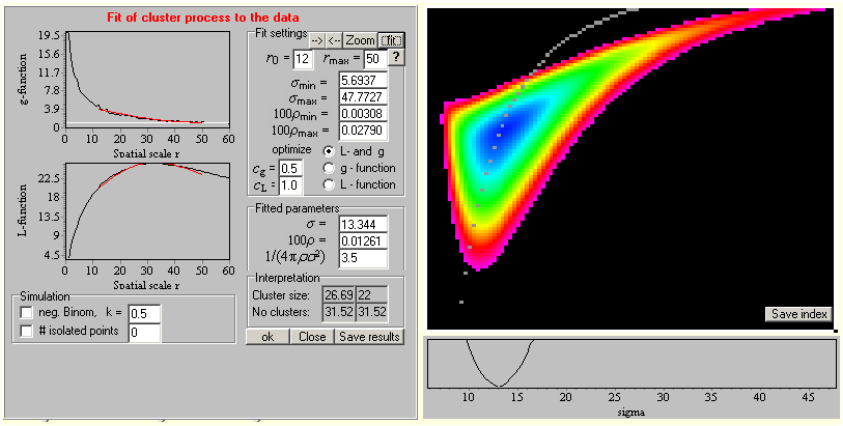
- we can first fit the parameters ρ_1 and σ_1 of the large scale clustering using a distance interval that starts with distance larger than $\approx 2\sigma_2$. In this case the second summand will be very small and because of $\sigma_2 << \sigma_1$ the third summand will be dominated by σ_1. Thus, in this case we basically fit the generalized simple Thomas process to the data and obtain unbiased estimates for the parameters ρ_1 and σ_{sum} of the large scale clustering.
- Second, we use the estimates of ρ_1 and σ_1 and fit the unknown parameters ρ_2 and σ_2 of the small scale clustering now using the entire distance interval. Finally, we use the estimate of σ_2 to estimate σ_1 using $\sigma_1^2 = \sigma_{sum}^2 + \sigma_2^2$. Note that the fitting procedure of *Programita* uses σ_{sum} in the third summand and estimates σ_2 via the second summand.

Example Book_Fig4_15b.res (fit parameters σ_1, ρ_1, σ_2, ρ_2)

This example file is shown in Figure 4.15a in Wiegand and Moloney (2014). The data are the locations of small saplings of the species *Shorea congestiflora* from the Sinharaja plot in Sri Lanka. This data set was analyzed in detail in Wiegand et al. (2007).

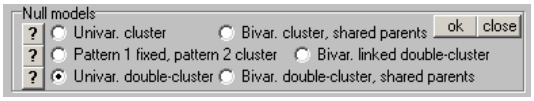
To analyze this data set we first fit the Thomas process with two nested scales of clustering to the data, following the two-step procedure outlined above, and then simulate this point process with different parameters f_{k1} and f_{k2} of the negative Binominal distribution.

1. Execute *Programita*.
2. Highlight data file Book_Fig4_15a.dat you want to analyze in Input data file.
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
6. Press button "**Calculate Index**"
7. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
8. Select "**Cluster process**" in the window "Select a null model".
9. A window "Fit of cluster process to data" opens. Select in the section "Null models" at the bottom "**Univar. cluster**". Continue with the small **ok** button.
10. Fit the parameters σ_{sum} and ρ of large-scale clustering over the distance interval 12 – 50m. (See example Book_Fig4_12.res how to determine the value of r_0). The fit is quite good, in the example below we obtain σ_{sum} =13.34 and $A\rho$ = 31.5:

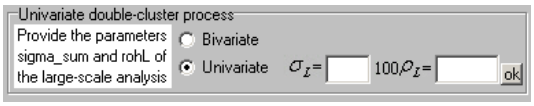


11. Execute another copy of *Programita*.
12. Highlight data file Book_Fig4_15a.dat you want to analyze in Input data file.
13. Select "**no grid**" in What do you want to do?
14. Select bin of 1m window Select a new cell size
15. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
16. Press button "**Calculate Index**"
17. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
18. Select "**Cluster process**" in the window "Select a null model".

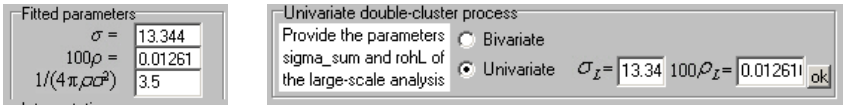
19. A window "Fit of cluster process to data" opens. Select in the section "Null models" at the bottom "**Univar. double-cluster**".



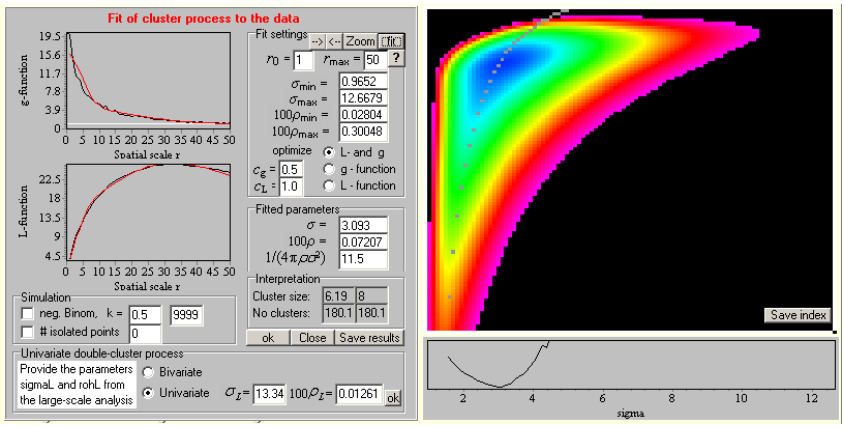
and "Univariate" in the window that appears:



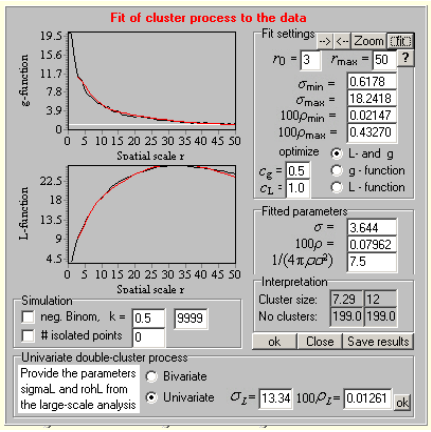
20. Copy the fitted parameters σ_{sum} and ρ_1 of the large-scale clustering from the first copy of *Programita* into the window for the univariate double-cluster process and click the small "**ok**" button and the next small **"ok"** button in the null model window:



21. Now you can fit the parameters σ_2 and ρ_2 of small-scale clustering over the entire distance interval 1 – 50m:

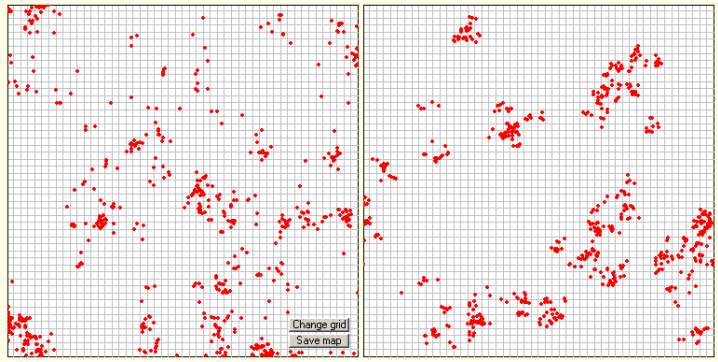


22. You notice that there is probably another level of small-scale clustering (because the pair correlation function at very small distances does not fit). Therefore use an interval of 3-50m:

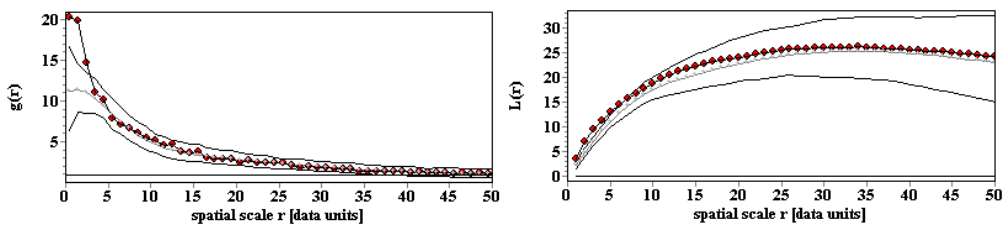


which produces a somewhat better fit. In the example below we obtain σ =3.6 and $A\rho$ = 199. Click the small **"ok"** button and then "**Calculate Index**" to simulate the fitted point process.

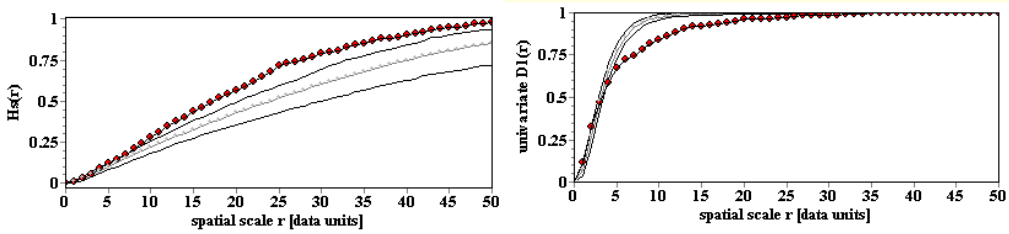
23. The simulated patterns are similar to the observed one, but there are not enough isolated points:



24. The pair correlation function is well fitted, but as expected, there is an additional clustering at very small distances < 3m which also causes some smaller departures in the *L*-function:



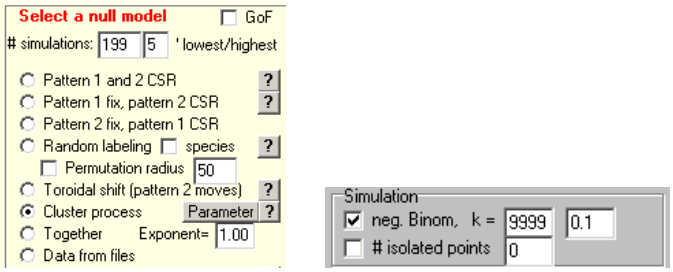
However, the spherical contact distribution is clearly underestimated (i.e., the gaps are too large in the simulated patterns) and the distribution function of the distances to the nearest neighbor are overestimated at distances > 5m which indicates that the data have more isolated points than the data generated by the cluster process:



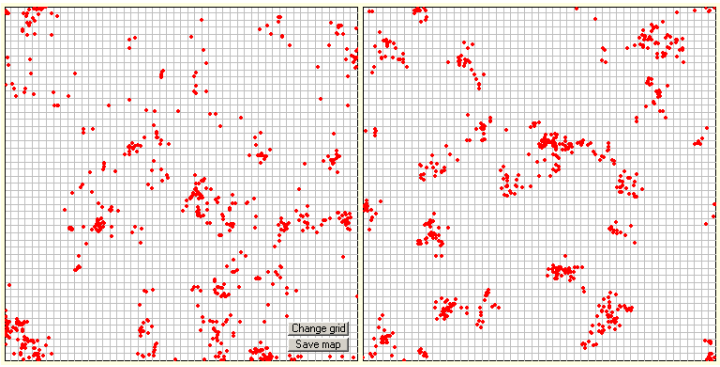
Examples Book_Fig4_15f.res (fit parameters f_{k1}, f_{k2})

This example continues the previous example Book_Fig4_15b.res by useing a negative Binomial distribution to generate a clumped distribution of the number of saplings over the small clusters. We select as in Figure 4.15f $k_1 = \infty$ ($f_{k1} = 1$) and $k_2 = 0.1$ ($f_{k2} = 11$).

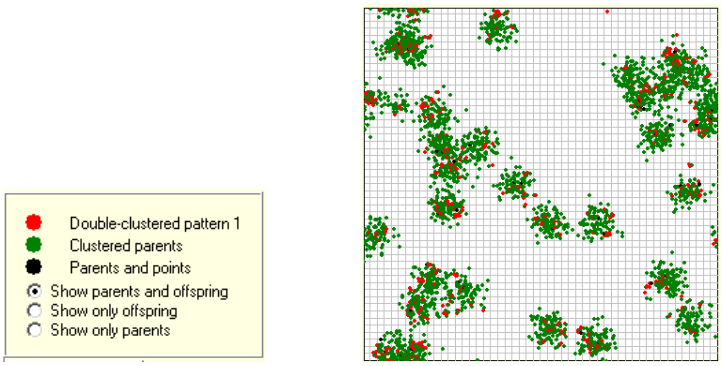
25. To access the parameter fitting window click "Parameters" at the Select a null model window, click "**neg. Binom**" at the "simulation" window, insert $k = 9999$ for the large scale clustering, $k = 0.1$ for the small scale clustering, and then click the small **"ok"** button:



26. After pressing the "**Calculate Index**" button *Programita* simulates the generalized cluster process with two nested scales of clustering. Comparison between data and simulated patterns shows that the simulated pattern still has too large gaps:

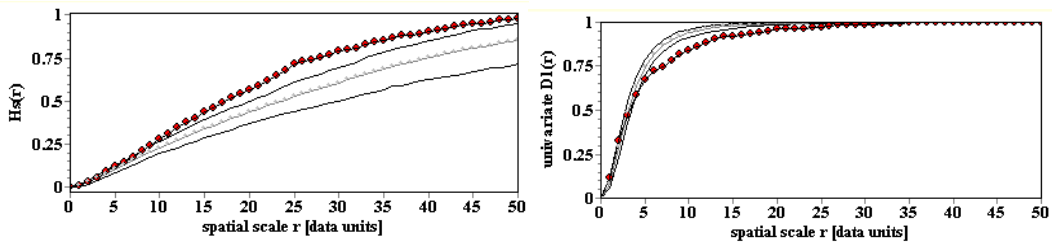


If you select in the small window below the simulated pattern "Show parents and offspring" you can see additionally to the simulated pattern (red points) the cluster centers (of the small-scale clustering) as green points:



27. Clearly, because of $k_2 = 0.1$ and $f_{k2} = 11$ you have 11 times as much cluster centers as in the example above with $f_{k2} = 1$.

    The fit of the spherical contact distribution does not improve but the fit of the distribution function of the distances to the nearest neighbor somewhat improves, thus the important feature of the data is probably not the distribution of the number of points over the small clusters:

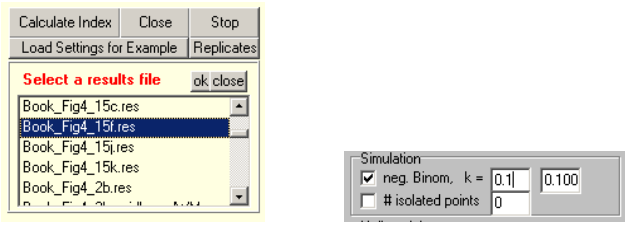


Examples Book_Fig4_15j.res (fit parameters f_{k1}, f_{k2})

You can easily verify that the point process analogous to Book_Fig4_15f.res with parameters $k_1 = 0.1$ ($f_{k1} = 11$) and $k_2 = \infty$ ($f_{k2} = 1$) does also not much improve the fit of the spherical contact distribution and the distribution function of the distances to the nearest neighbor.

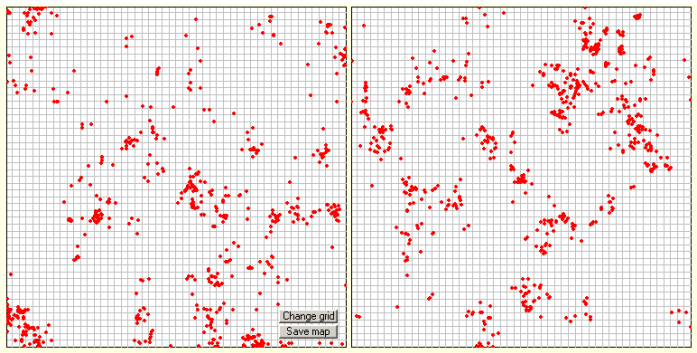
This example therefore continues the previous example Book_Fig4_15f.res by using a negative Binomial distribution to generate a clumped distribution of both, the number of small clusters over the large clusters and the number of saplings over the small clusters. We select as in Figure 4.15j parameters $k_1 = 0.1$ ($f_{k1} = 11$) and $k_2 = 0.1$ ($f_{k2} = 11$).

1. To access the parameter settings of example Book_Fig4_15f.res click "**Load Settings for Example**", highlight file Book_Fig4_15f.res, and click the small **"ok"** button.

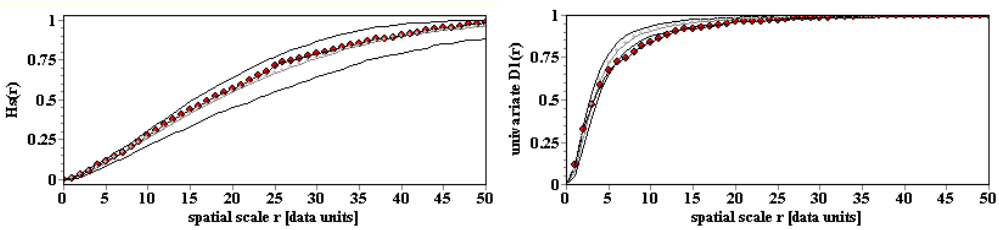


2. Now change in the "simulation" window the parameters k to insert $k = 0.1$ for the large scale clustering and $k = 0.1$ for the small scale clustering, and then click the small "**ok**" button.

3. After pressing the "**Calculate Index**" button *Programita* simulates the generalized cluster process with two nested scales of clustering. Comparison between data and simulated patterns shows that the simulated pattern looks now very similar to the observed pattern:



The fit of the $H_s(r)$ is now perfect, and that of the $D_1(r)$ substantially improved. Note also that the GoF test of the $D^k(r)$ was not significant for $k > 3$:



3.2.12 Superposition of CSR and a double cluster process (Book_Fig4_16e.res)

In the previous examples Book_Fig4_15b.res, Book_Fig4_15f.res, and Book_Fig4_15j.res we generalized the double-cluster Thomas process to produce a pattern with the same second-order structure but more isolated points. This was done by manipulating the distribution of the number of points over the clusters and by increasing the number of clusters.

As already shown for the simple Thomas process in example Fig4_11CSR100.res, we can also generate a pattern with the same second-order structure but more isolated points if we independently superimpose a simple double cluster Thomas process (i.e., $f_{k1} = 1$ and $f_{k2} = 1$) with a CSR pattern (for details see section 4.1.4.6 "Independent Superposition of CSR within a Double Cluster Process"). A nice feature of the Thomas processes is that superposition with CSR does not change the functional form of the analytical solution of the pair correlation function (and the *K*-function). We obtain:

$$g_{22}(r,\sigma_1,\rho_1,\sigma_2,\rho_2) = 1 + \underbrace{\frac{p_C^2}{\rho_2} h(r,\sqrt{2}\sigma_2)}_{\substack{\text{small-scale}\\ \text{clustering}}} + \underbrace{\frac{p_C^2}{\rho_1} h(r,\sqrt{2\sigma_1^2 + 2\sigma_2^2})}_{\substack{\text{combined effect of small-}\\ \text{and large-scale clustering}}}$$

where $h(r,\sigma) = \dfrac{\exp(-0.5r^2/\sigma^2)}{2\pi\sigma^2}$

and the parameter p_C is the proportion of points of the pattern belonging to the double-cluster Thomas process.

In practical terms this means that superposition with CSR does not change the functional form of the Thomas process, and that we can therefore find for each value of p_C a double cluster Thomas process (where $p_C = 1$) with exactly the same pair correlation (and *K*-) function. The parameter of this double cluster Thomas process that describes the number of clusters yields $\rho_S = \rho/p_C^2$ where ρ is the parameter fitted to the superposition process. That means that superposition with CSR generates a pair correlation function that seems to have more clusters. Comparison with the pair correlation function of the generalized double cluster process

$$g_{22}(r,\rho_1,\sigma_1,\rho_2,\sigma_2) = 1 + \underbrace{\frac{f_{k2}}{\rho_2} h(r,\sqrt{2}\sigma_2)}_{\substack{\text{small-scale}\\ \text{clustering}}} + \underbrace{\frac{f_{k1}}{\rho_1} h(r,\sqrt{2\sigma_1^2 + 2\sigma_2^2}}_{\substack{\text{combined effect of small-}\\ \text{and large-scale clustering}}}$$

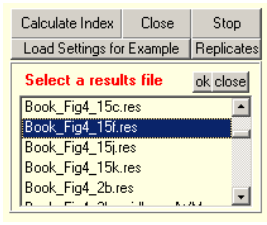
shows that the parameter p_C^2 of the superposition process plays the same role as the parameters f_{k1} and f_{k2} in the generalized double cluster Thomas process.

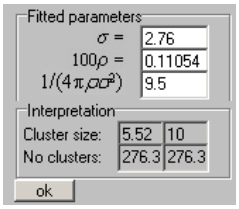
As before, this means that fitting with second order properties alone does not allow us to determine the proportion (1- p_C) of isolated CSR points. However, other summary statistics of different nature such as the spherical contact distribution $H_s(r)$ and the distribution function $D^k(r)$ of the distances to the kth neighbor may allow us to approximate the value of p_C. The procedure for this is exactly the same as for the generalized double cluster Thomas process described above.

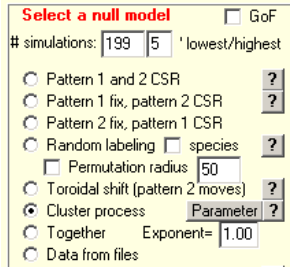
Examples Book_Fig4_16e.res (superposition with random pattern)

This example continues the previous example Book_Fig4_15c.res by independently superimposing a simple double cluster Thomas process (i.e., $f_{k1} = 1$ and $f_{k2} = 1$) with CSR.

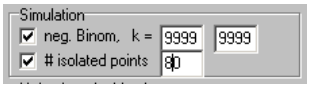
1. To access the parameter settings of example Book_Fig4_15c.res click "**Load Settings for Example**", highlight file Book_Fig4_15c.res, and click the small "**ok**" button. Then click the small "**ok**" button at the window "Fitted parameters" and the button "Parameters" at the Select a null model window:





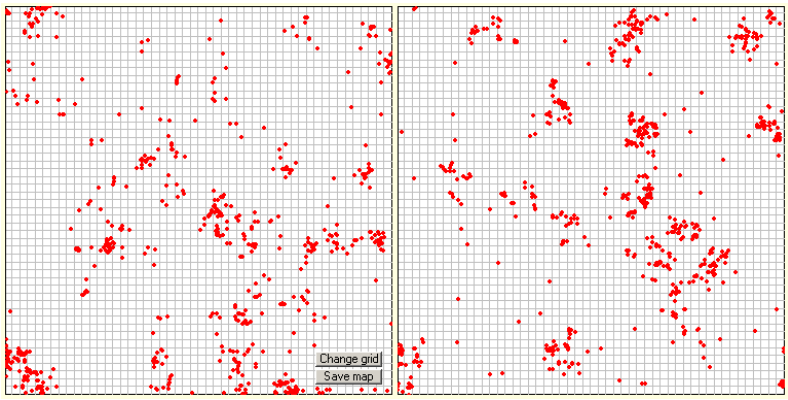


2. Now click "# isolated points", insert the number of points of the pattern that belong to the CSR pattern (80 in the example),

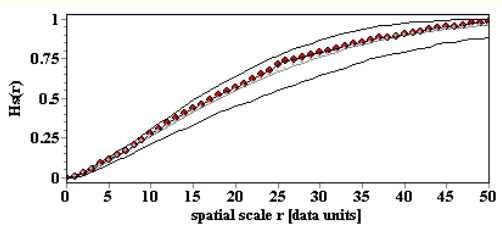


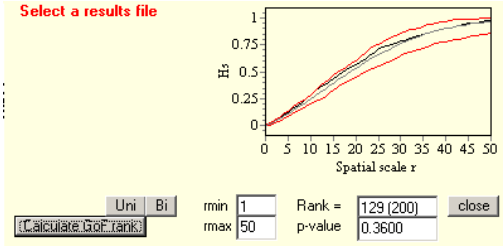
and finally click the small "**ok"** button at the window "Fitted parameters".

3. After pressing the "**Calculate Index**" button *Programita* simulates the superposition point process. Comparison between data and simulated patterns shows that the simulated pattern looks very similar to the observed pattern:

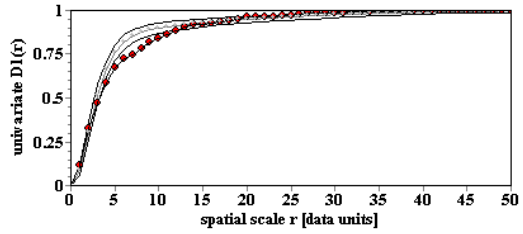


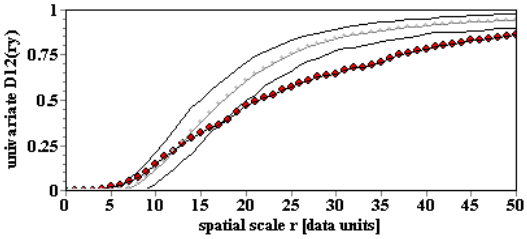
The fit of the $H_s(r)$ is very good, the GoF test over distance interval 1-50m yields a P-value of 0.36:





However, while the distribution function of the distances to the nearest neighbor is relatively close to the data, the $D_k(r)$ fails for higher neighborhood ranks k, for example for $k = 12$:





That means that the internal structure of the clusters is not well represented by the superposition process. This indicates that the generalized double cluster Thomas process presented above in example Book_Fig4_15f.res is a more likely model for this data set.

3.2.13 Superposition of two Thomas processes (Fig4_15super.res)

The previous example Book_Fig4_15b.res showed a Thomas process with two scales of clustering where the clusters where nested; i.e., small clusters where located inside large clusters. However, this is not the only possibility to generate univariate patterns with two scales of clustering. Stoyan and Stoyan (1996) proposed a point process resulting from the independent superposition of two simple Thomas processes. The additional parameter of this point process is the proportion p_1 of the points that belong to the Thomas process with large-scale clustering. The pair correlation function for this point process yields:

$$g_{22}(r,\sigma_1^s,\rho_1^s,\sigma_2^s,\rho_2^s) = 1 + \underbrace{\frac{(1-p_1)^2}{\rho_2^s}h(r,\sqrt{2}\sigma_2^s)}_{\text{small-scale clustering}} + \underbrace{\frac{p_1^2}{\rho_1^s}h(r,\sqrt{2}\sigma_1^s)}_{\text{large-scale clustering}}$$

where $h(r,\sigma) = \dfrac{\exp(-0.5r^2/\sigma^2)}{2\pi\sigma^2}$. For example, if both component processes have the same number of points (i.e., $p_1 = 0.5$), the contribution of clustering $h(r, 2^{0.5}\sigma)/\rho$ of each component is only one quarter of that of the original process. Thus, clustering of the superposition process is substantially reduced compared with the clustering of the two component processes. (If we have also $\sigma_1 = \sigma_2$ we obtain, as expected, a process with only one scale of clustering, but with the double number of clusters).

For comparison, the pair correlation function nested double cluster process yields:

$$g_{22}(r,\rho_1,\sigma_1,\rho_2,\sigma_2) = 1 + \underbrace{\frac{1}{\rho_2}h(r,\sqrt{2}\sigma_2)}_{\text{small-scale clustering}} + \underbrace{\frac{1}{\rho_1}h(r,\sqrt{2\sigma_1^2+2\sigma_2^2}}_{\text{combined effect of small- and large-scale clustering}}$$

Thus, the pair correlation function of the superposition of two simple Thomas processes has the same structure as that of the nested double cluster process. We can therefore fit the parameters ρ_1, σ_1, ρ_2, and σ_2 by using the procedure of the nested double cluster process. However, comparing the two equations suggests that small adjustments in the parameters are needed. If we fit with the procedure for the nested double cluster process we obtain parameters ρ_1, σ_1, ρ_2, and σ_2, but to describe the superposition process we have parameters ρ^s_1, σ^s_1, ρ^s_2, and σ^s_2. The parameter σ_2 does not change (i.e., $\sigma_2^s = \sigma_2$), but we need to adjust the parameter σ_1^s that determines the size of the large clusters to

$$\sigma_1^s = \sqrt{\sigma_1^2 - \sigma_2^2}$$

and the parameters ρ_1 and ρ_2 that determine the number of cluster centers must be transformed to $\rho_1^s = \rho_1 p_1^2$ and $\rho_2^s = \rho_1 (1-p_1^2)$. *Programita* adjusts this automatically, but the *.res file outputs the parameters ρ_1, σ_1, ρ_2, and σ_2.

Thus, the two components Thomas processes of the superposition process have fewer cluster centers than the corresponding nested process. Or in other words, to yield the same pair correlation function for the nested and the superposition double cluster process, the superposition process must have substantially less clusters than the corresponding nested process (i.e., $\rho^s_1 = \rho_1 p^2_1$ and $\rho^s_2 = \rho_2 (1- p^2_1)$.

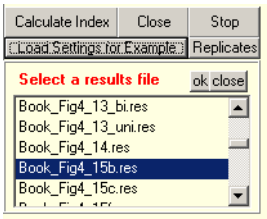
One consequence of this is that in cases where the proportion p_1 of the points that belong to the Thomas process with large-scale clustering is small or large, we may have cases with (formally) less than one small or large cluster, respectively. Thus, one has to check if the parameters of the process make sense. In cases where one component process has less than one cluster *Programita* gives a warning and you can change the parameter p_1 to yield exactly one cluster. However, with one cluster the point process is not well represented; it should have several clusters to make sense.

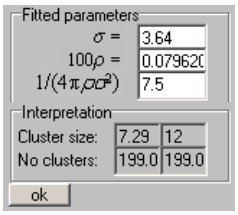
As before, fitting with second order properties alone does not allow us to determine the proportion p_1 of the points that belong to the Thomas process with large-scale clustering. However, other summary statistics of different nature such as the spherical contact distribution $H_s(r)$ and the distribution function $D^k(r)$ of the distances to the kth neighbor may allow us to approximate the value of p_1.

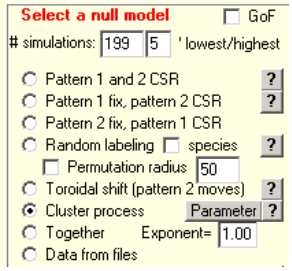
Examples Fig4_15super.res (superposition of two Thomas processes)

This example continues the previous example Book_Fig4_15b.res by fitting the point process that independently superimposes two simple Thomas processes to the data.

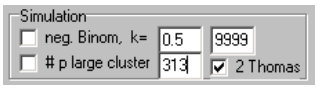
1. To access the parameter settings of example Book_Fig4_15b.res click "**Load Settings for Example**", highlight file Book_Fig4_15b.res, and click the small "**ok**" button. Then click the small "**ok"** button at the window "Fitted parameters" and the button "Parameters" at the Select a null model window:





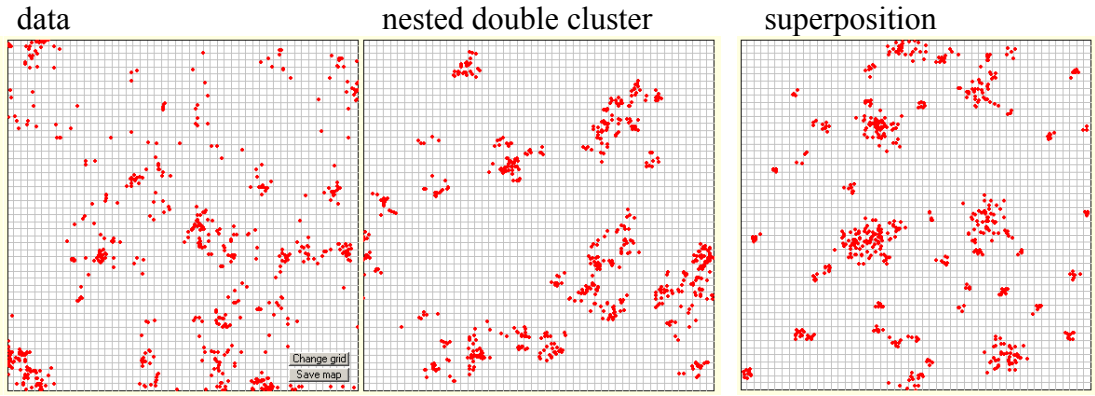


2. Now click "**2 Thomas**", and insert the number of points of the large-scale Thomas process (313 in the example yielding a proportion $p_1 = 0.5$),

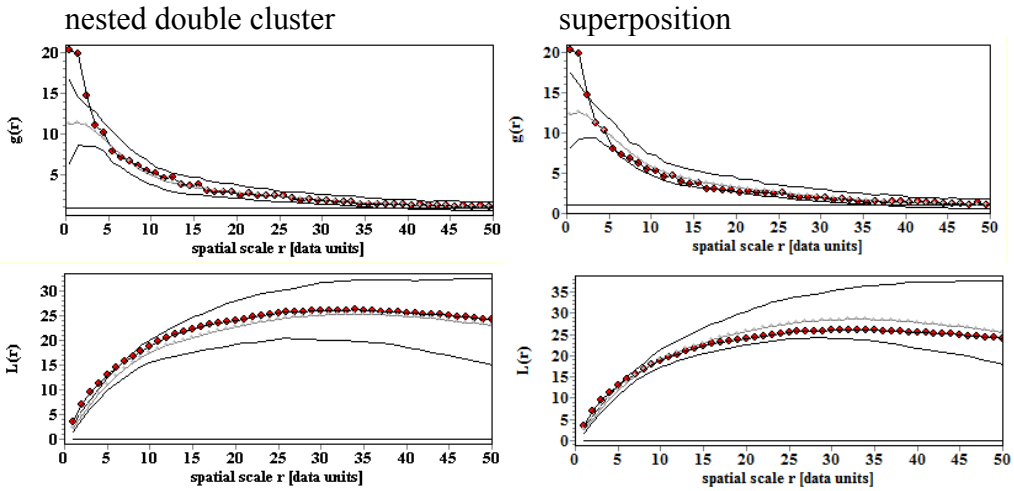


   and finally click the small "**ok"** button at the window "Fitted parameters".
3. After pressing the "**Calculate Index**" button *Programita* simulates the superposition point process. Comparison between the simulated nested double-cluster process and the superposition process show clear differences.

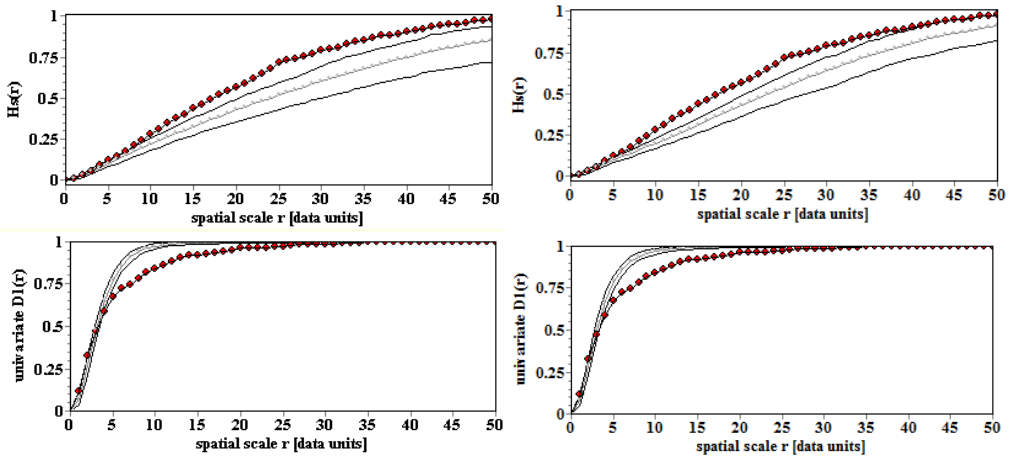
The superposition process shows many small clusters (49 vs. 198 for the double cluster process) and few large clusters (7 vs. 31 for the double cluster process):



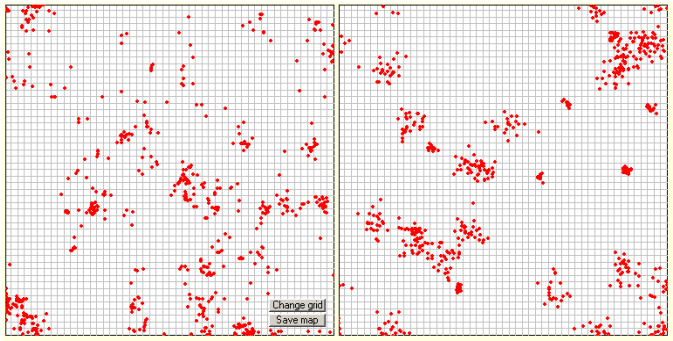
As expected, the fit of the pair correlation function and the *L*-function of the superposition and the nested process are similar:



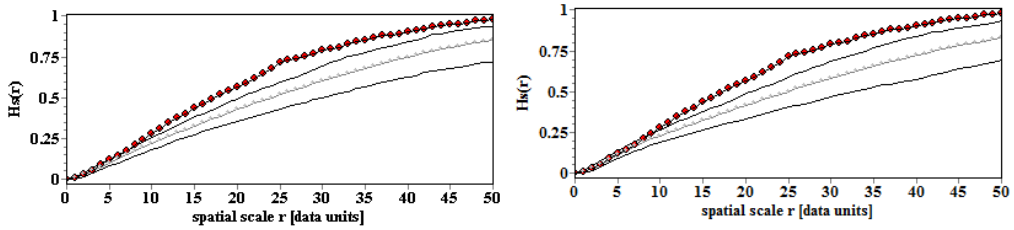
Interestingly, the fit for the spherical contact distribution and the nearest neighbor distribution function are also similar:



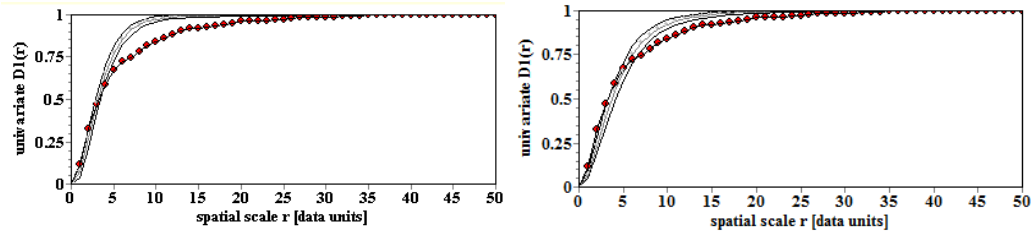
4. Using instead a superposition process with a large-scale cluster process with 500 points (and 126 points being part of the small-scale cluster process) we have 20 large clusters but 8 small clusters:



5. The spherical contact distributions of the nested double cluster and the superposition process are similar:



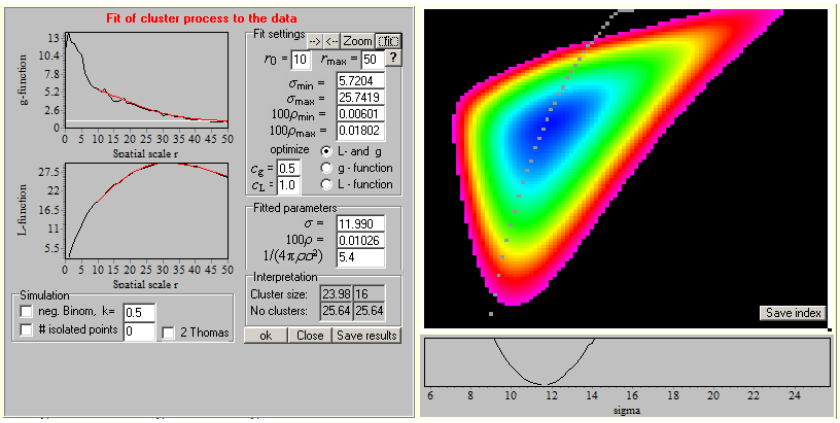
but the nearest neighbor distribution function changes:



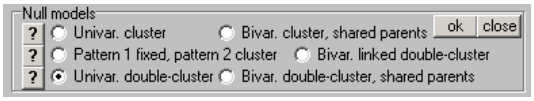
Examples TwoThomas313.res (superposition of two Thomas processes)

In this example we use a pattern generated with the fitted superposition process in example Fig4_15super.res where both component processes had the same number of points (= 313). We fit this pattern to the superposition process of two simple Thomas processes.

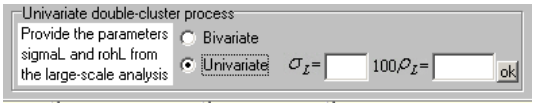
1. Execute *Programita*.
2. Highlight data file TwoThomas313.dat you want to analyze in Input data file.
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
6. Press button "**Calculate Index**"
7. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
8. Select "**Cluster process**" in the window "Select a null model".
9. A window "Fit of cluster process to data" opens. Select in the section "Null models" at the bottom "**Univar. cluster**". Continue with the small **ok** button.
10. Fit the parameters σ and ρ of large-scale clustering over the distance interval 10 – 50m. (See example Book_Fig4_12.res how to determine the value of r_0). The fit is quite good, in the example below we obtain σ =11.99 and $A\rho$ = 25.64:



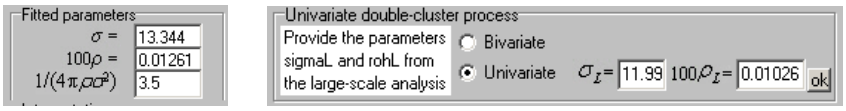
11. Execute another copy of *Programita*.
12. Highlight data file TwoThomas313.dat you want to analyze in Input data file.
13. Select "**no grid**" in What do you want to do?
14. Select bin of 1m window Select a new cell size
15. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
16. Press button "**Calculate Index**"
17. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
18. Select "**Cluster process**" in the window "Select a null model".
19. A window "Fit of cluster process to data" opens. Select in the section "Null models" at the bottom "**Univar. double-cluster**".



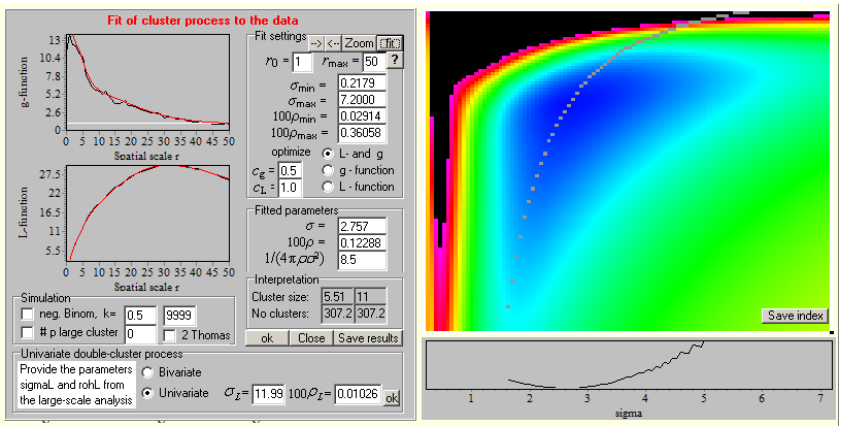
and “Univariate” in the window that appears:



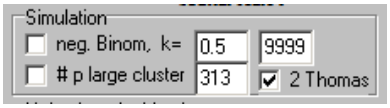
20. Copy the fitted parameters of the large-scale clustering from the first copy of *Programita* into the window for the univariate double-cluster process and click the small “**ok**” button and the next small “**ok”** button in the null model window:



21. Now you can fit the parameters σ and ρ of small-scale clustering over the entire distance interval 1 – 50m:

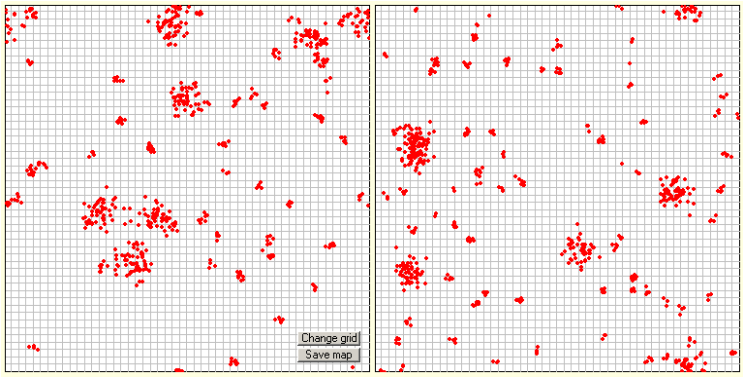


22. The fit with parameters σ =2.7 and $A\rho$ = 307 looks good.
23. Select the checkbox “2 Thomas” and insert true number of points of large scale clustering (313):

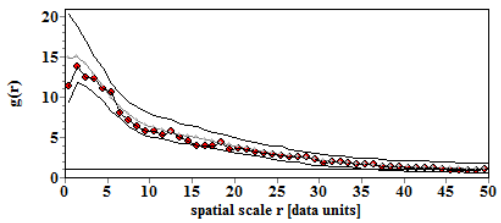


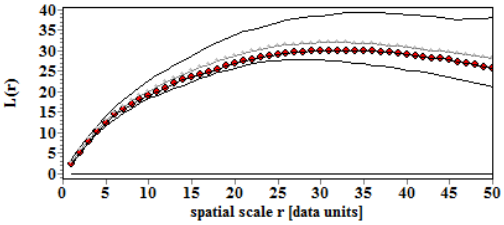
24. Now click the small “**ok”** in the “Fitted parameter” window and the “**Calculate Index**”. *Programita* now simulates the superposition point process:

25. As expected, the simulated and the observed pattern are very similar, we have 6 large clusters and 77 small clusters:

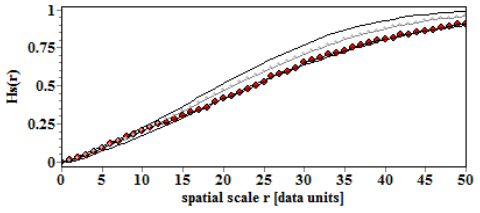


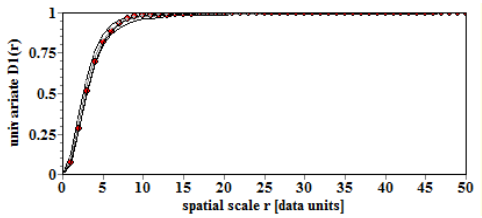
As expected, the pair correlation function and the *L* function are well fitted:





The spherical contact distribution is slightly overestimated but the distribution function of the nearest neighbor distances is well fitted:





The parameters of the original point process are reasonably recovered: $\sigma_2 = 2.76$ (vs. 3.64), $\sigma_1 = 12$ (vs. 13.3), $A\rho_2 = 307$ (vs. 199), and $A\rho_1 = 25.7$ (vs. 31.5). Because of the stochasticity of the point process and the low number of clusters we cannot expect a better fit between parameters used to simulate the point process and the fitted parameters of a realization of the point process.

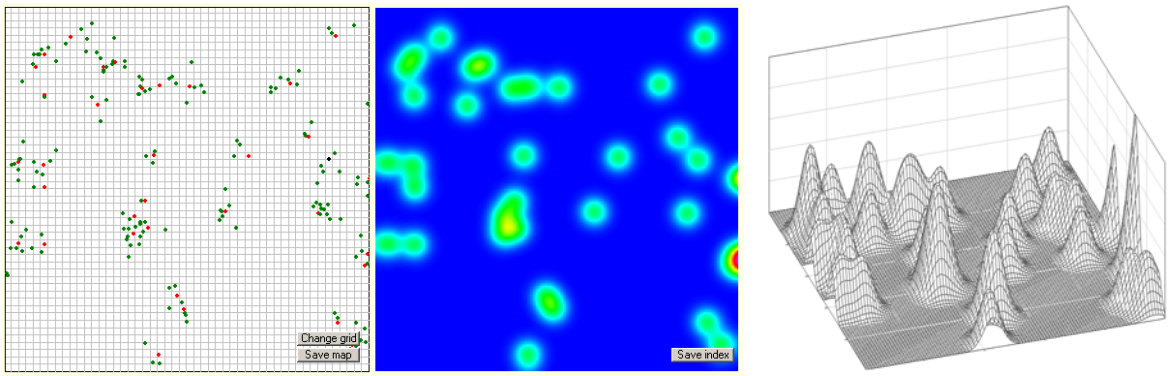
As exercise you can simulate the superposition process with different numbers of points of the large scale clustering. With 500 points (*TwoThomas313_500.res*) the spherical contact distribution is now underestimated and the distribution function of the nearest neighbor distances is also underestimated.

With 250 points (*TwoThomas313_250.res*) the spherical contact distribution is now substantially overestimated but the distribution function of the nearest neighbor distances is well fitted.

3.2.14 Cox processes

Cox processes are a broad class of point processes that encompass and can combine heterogeneous Poisson processes and cluster processes. They are based on a generalized intensity function. More detail can be found in section 4.1.5 “Cox Processes” in Wiegand and Moloney (2014) and especially in Section 6.4 of the book of Illian et al. (2008, pp. 379–386).

For example, a simple Thomas process can also be generated with an intensity function that is based on superposition of two dimensional normal distributions that are centered on the locations of the cluster centers. Here is for example on the left the simple parent-offspring pattern Book_Fig4_13.dat shown in Figure 4.13 (see example Book_Fig4_13.res). The cluster centers are shown as red points and the points of the simple Thomas process as green points. The pattern was simulated based on 34 cluster centers and a parameter $\sigma = 13.3$ governing the normal distribution. In the middle and right is the intensity function that results from the superposition of the normal distribution with parameter $\sigma = 13$ centered on the 34 cluster centers (saved as file int_G_Book_Fig4_13_R1_13.int):



The Cox/simple Thomas process can be simulated in a straight forward way using the methods for the heterogeneous Poisson process. Points with random coordinates **x** are proposed and a point is accepted with a probability proportionally to $\lambda(\mathbf{x})/\lambda^*$where λ^* is the maximal value of $\lambda(\mathbf{x})$. These random trials are repeated as long as all n points of the pattern are distributed.

If the cluster centers are the same for all simulations of the point process, we have a parent-offspring Thomas process which is basically a heterogeneous Poisson process with a fixed intensity function $\lambda(\mathbf{x})$. However, we may also generate for each simulation of the point process a separate set of cluster centers as done in the simple Thomas process. This yields a “double stochastic” point process where the intensity function is itself stochastic and a realization of a stochastic process $\Lambda(\mathbf{x})$.

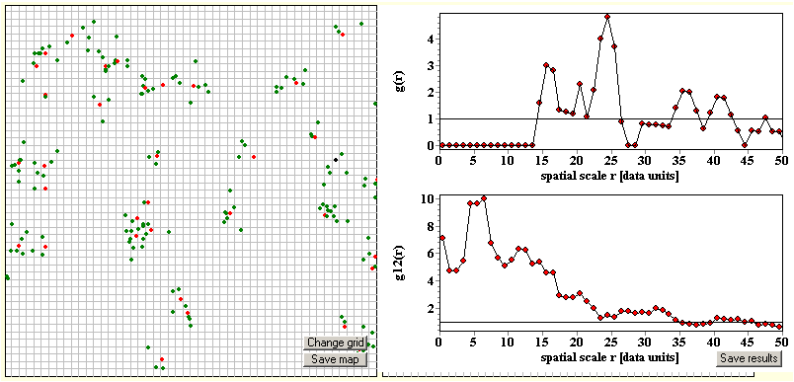
It is clear that superposition of an intensity function generated by homogeneous Thomas process with another intensity function that represents a large-scale trend, for example due to different habitat suitability (where the probability that a point is accepted depends on habitat suitability), generates an inhomogeneous Thomas process. This is an interesting feature of Cox processes that allows fitting inhomogeneous Thomas processes to the data.

Example Book_Fig4_18.res

In this example we reanalyze example Book_Fig4_13.res to illustrate the duality between a Cox process and a parent-offspring Thomas process where the cluster centers (i.e., parents) are known and are the same in each simulation of the point process.

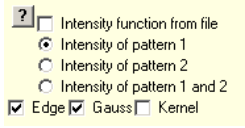
The example file Book_Fig4_13.dat was generated with a simple Thomas process with parameter $\sigma_2 = 13.3$, $A\ \lambda_1 = 34$ random clusters and 157 points and parameters.

1. Execute *Programita*.
2. Highlight data file Fig4_13.dat you want to analyze in Input data file.
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 3 in the menu "Which method will you use"
6. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
7. Press button "**Calculate Index**"
8. *Programita* shows the pattern, the univariate pair correlation function of the cluster centers and the bivariate pair correlation function of the points around their parents:

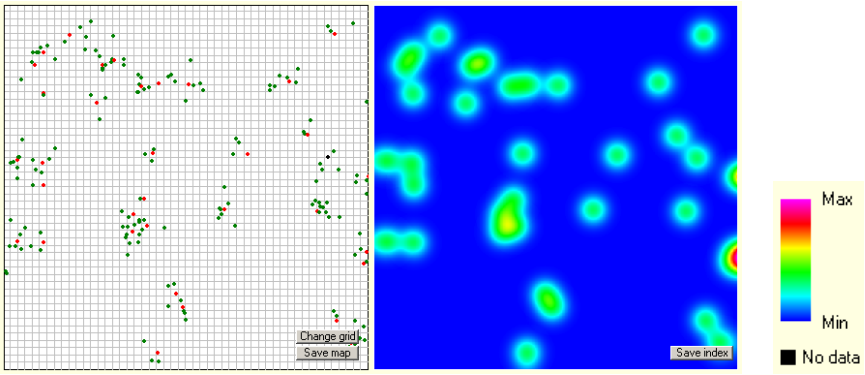


The pair correlation function of pattern 1 which is not of interest here is somewhat rugged because pattern 1 has few points (i.e., 34 clusters).

9. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
11. Select in the window Select a null model "**Pattern 1 fix, pattern 2 CSR**".
12. Specify the number of simulations of the null model (199 in the example) and the rule for the estimation of simulation envelopes (here the 5th lowest and highest values of the summary statistic of the 199 simulated null model data sets).
13. Click checkbox "**Heterogeneous Poisson**".
14. Go to window "Settings for hetero. Poisson" on the left and insert the bandwidth R (13m in the example) and enable "**Gauss**" for the Gaussian kernel. Now select "**Intensity of pattern 1**". In this case *Programita* estimates the intensity of pattern 1 (i.e., the cluster centers) based on a Gaussian kernel. (If you would select "Intensity of pattern 2" *Programita* would estimate the intensity of pattern 2). Edge correction "**Edge**" is enabled by default. *Programita* then uses this intensity function in the heterogeneous Poisson null model.

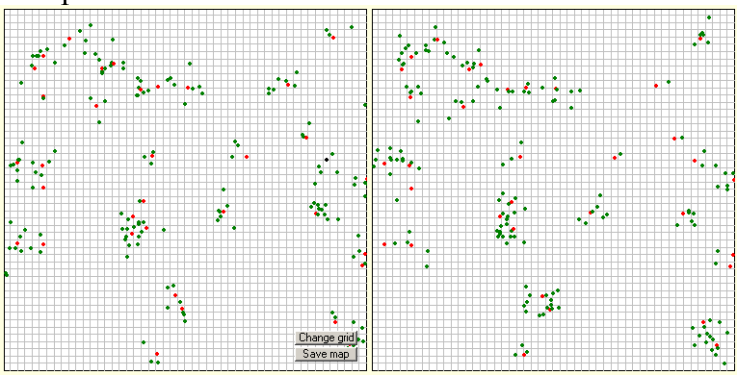


15. Click "**Calculate Index**" and *Programita* estimates the intensity function and shows the pattern and the corresponding intensity function.

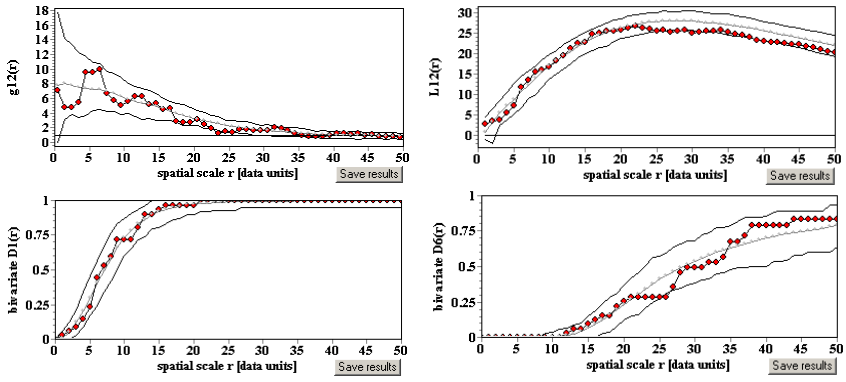


16. Click OK at the message box to save the intensity file. The file is saved with name int_G_Book_Fig4_13_R1_13.int where the "int_GE" indicates the Gaussian kernel, Book_Fig4_13.dat was the data file, "_R1_13" means that the intensity was estimated with pattern 1 and bandwidth 13.

    Now *Programita* conducts the analysis. You can observe during the simulations that the null model distributes the points of pattern 2 with probability proportionally to the intensity function. (If you would select e.g., null model "Pattern 1 and 2 CSR", pattern 1 and 2 would be randomized following this intensity function) Here an example:



The result reproduce that of Fig. 4.18d,-f and resembles that of example Book_Fig4_13bi.res well, outlining the equivalence of the Thomas and the Cox process:

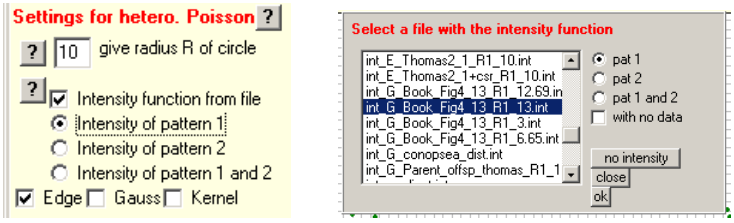


Example Book_Fig4_18_file.res

If you have the intensity file already saved, you can do the analysis also using this file. Starting with step 13 above:

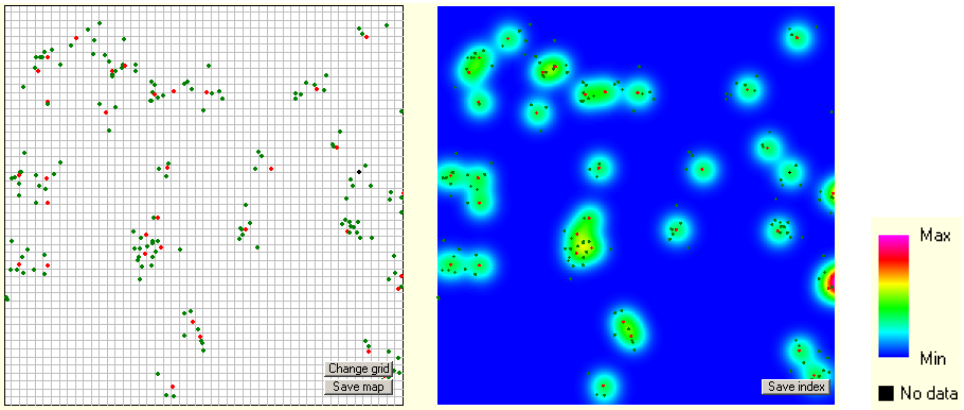
13. Click checkbox "**Heterogeneous Poisson**".
14. Go to window "Settings for hetero. Poisson", select "**Intensity of pattern** 1" (because you use an estimate of the intensity of pattern 1) and then select "**Intensity file function from file**" on the left.

    The window "Select a file with the intensity function" appears where you can highlight the intensity file you want to use (i.e., **int_G_Book_Fig4_13_R1_13.int**). Select "**pat 1**" (because you use an estimate of the intensity of pattern 1), and then click the small **ok** button:



Programita then uses this intensity function in the selected null model. If your null model was "Pattern 1 and 2 CSR" the same intensity function is used for the heterogeneous Poisson process of pattern 1 and pattern 2, if your null model was "Pattern 1 fix and pattern 2 CSR" the intensity function is used for the heterogeneous Poisson process of pattern 2, and if your null model was "Pattern 2 fix and pattern 1 CSR" the intensity function is used for the heterogeneous Poisson process of pattern 1.

Programita shows the observed pattern (left) and the intensity function together with the data (right):



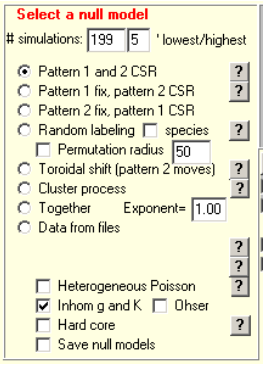
Click OK at the message box and "**Calculate Index**". Now *Programita* conducts the same analysis as before in example Book_Fig4_18.res

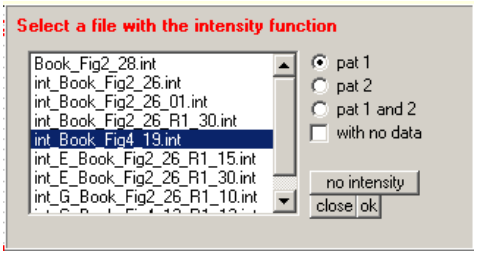
3.2.15 Inhomogeneous Thomas process (Book_Fig4_19b.res)

We now fit an inhomogeneous Thomas process to the data shown in Figure 4.19b in Wiegand and Moloney (2014). The pattern is a realization of an inhomogeneous Thomas process based on a non-parametric estimation of the intensity function of all living individuals of the species *Ocotea whitei* from the 2000 census at the BCI plot (details on the habitat model can be found in Table 4.1 in Wiegand and Moloney 2014). The parameters of the underlying homogeneous Thomas process used to generate the pattern were $\sigma = 4.8$ and $A\rho = 581$ cluster centers.

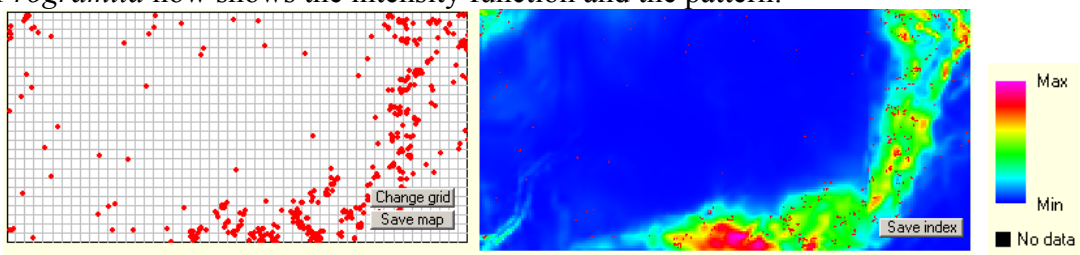
1. Execute *Programita*.
2. Highlight data file Book_Fig4_19b.dat you want to analyze in Input data file.
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 3 in the menu "Which method will you use"
6. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
7. Press button "**Calculate Index**"
8. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.

9. Enable check box "**Inhom g and K**"
10. The window "Select a file with the intensity function" appears where you select the intensity file you want to use for estimation of the inhomogeneous second-order summary statistics (i.e., int_Book_Fig4_19.int). Select "**pat 1**" (because it is the intensity of pattern 1) and then click the small "**ok**" button





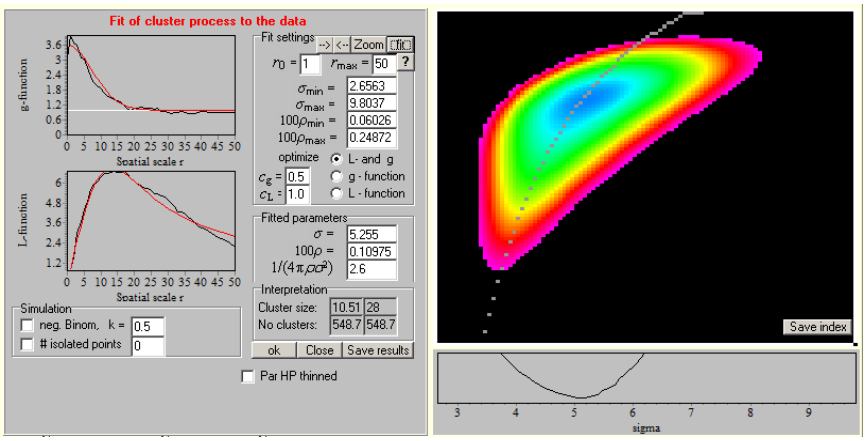
11. *Programita* now shows the intensity function and the pattern:



It is clear that most of the area is unsuitable (dark blue) and that many of the 581 clusters of the underlying homogeneous Thomas process will disappear.

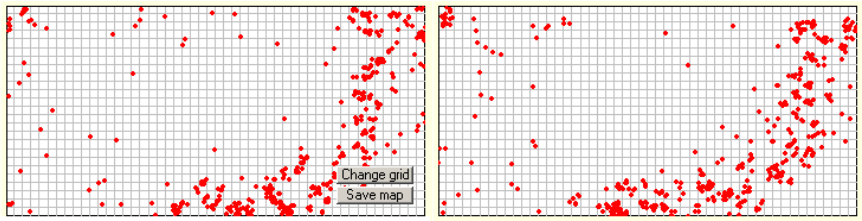
12. To fit the inhomogeneous Thomas process to the data select "**Cluster process**" in the window "Select a null model".
13. A window "Fit of cluster process to data" opens. Select in the section "Null models" at the bottom "**Univar. cluster**". Continue with the small **ok** button.

14. Fit the parameters σ, and ρ. You obtain a good fit for σ =5.3, $A\rho$ = 548 which is close to the parameters σ = 4.8 and $A\rho$ = 581 used to generate the pattern. Note that one cannot expect a perfect agreement between the parameters used for generating the pattern and the fitted parameters of one realization. This is because the Thomas process is a stochastic process and because many of the original clusters of the underlying homogeneous Thomas process disappeared during thinning with the intensity function.

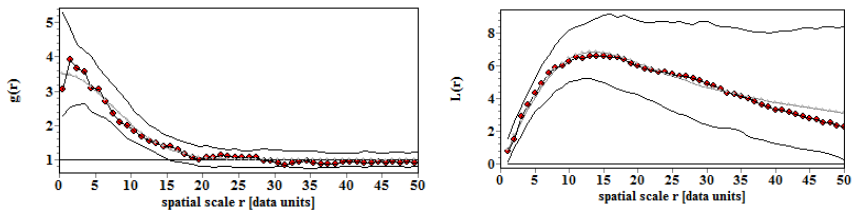


15. If you now click the "**ok**" 

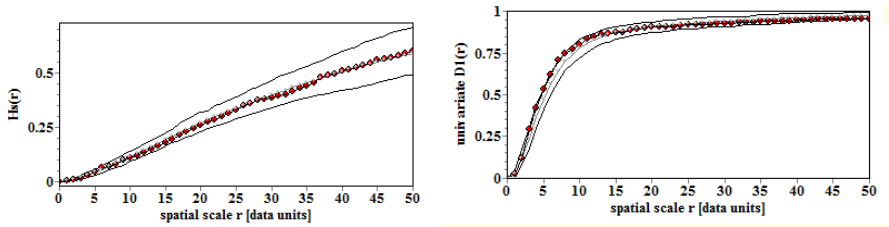
and the "**Calculate Index**" you simulate the inhomogeneous Thomas process. As expected, the simulated pattern resembles the observed pattern well:



The pair correlation and *L*- function are well fitted:



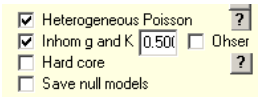
the spherical contact distribution $H_s(r)$ and the distribution function $D^k(r)$ of the distances to the kth neighbor (here the 1th neighbor) as well (the GoF test for $D_1(r)$ yields a *P*-value of 0.19):



3.2.16 Inhomogeneous *g*- and *K* functions (Book_Fig4_19b_Ohser.res)

The default estimator of the inhomogeneous second-order summary statistics is the inhomogeneous WM estimator detailed in equation 3.31 in Wiegand and Moloney (2014). This is an adapted estimator that conducts internally an estimation of a heterogeneous Poisson process based on the selected intensity function (the pattern $\mathbf{a}_j$ in equation 3.31) to estimate the expected area of rings or circles around the points of the pattern. Thus, *Programita* simulates internally an auxiliary heterogeneous Poisson process for the estimation of the summary statistics of both, the observed pattern and that of the simulated patterns. This makes the estimation somewhat slow.

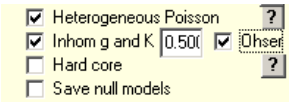
The number of points of the auxiliary heterogeneous Poisson process used to estimate the expected area of rings or circles around the points of the pattern is dim1*dim2/4 (where dim1 and dim2 are the dimensions of the study area in units of the bin you selected), but with a minimum of 50,000 points. If you want to use more (or less) points than that of the default you can provide a factor in the Select a null model window:



The more points you use the slower the estimation.

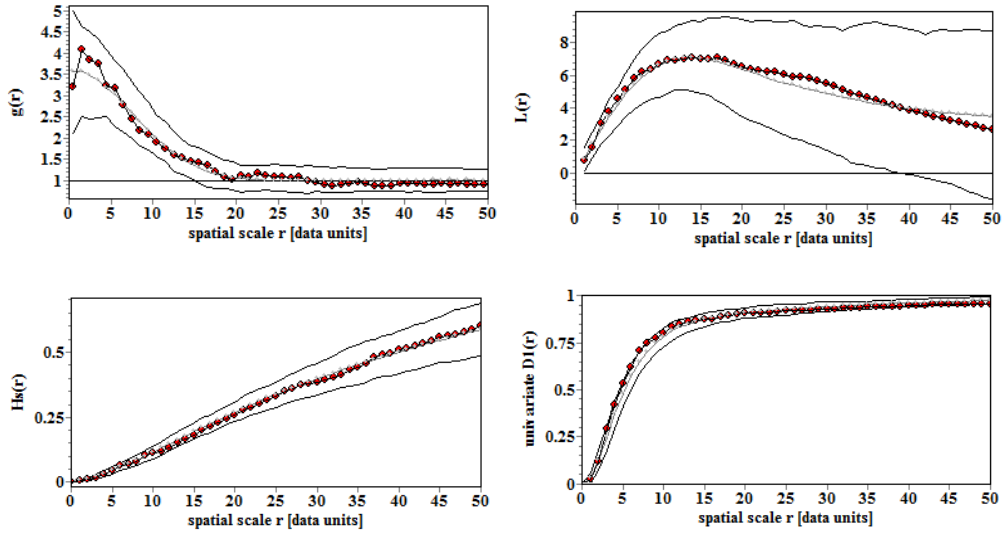
To speed up the estimation of the inhomogeneous summary statistics you can also use the alternative generalized Ohser estimator which is detailed in equation 3.29 and 3.30 in Wiegand and Moloney (2014). This estimator is not adapted in a sense that it does not estimate the expected area of rings or circles around the actual points of the pattern (as the corresponding WM estimator), but it estimates a generalization of the isotropized set covariance (equation 3.30) that yields basically the expected area of rings or circles around the points of a heterogeneous Poisson process based on the intensity function selected. The advantage is that the generalized isotropized set covariance needs to be estimated only once (because it is independent on the actual locations of the points of the pattern) and once estimated it can be applied to both, the estimation of the inhomogeneous summary statistics of the observed pattern and the simulated patterns.

To use the inhomogeneous Ohser estimator click the check box “Ohser”



The number of points of the auxiliary heterogeneous Poisson process used to estimate the generalized isotropized set covariance is dim1*dim2/8 (where dim1 and dim2 are the dimensions of the study area in units of the bin you selected), but with a minimum of 25,000 points.

Using the Ohser estimator instead of the WM estimator yields the same results:



You can also fit an inhomogeneous double cluster Thomas process; try for example the data set of Figure 3.13.

You can also use the generalized inhomogeneous double cluster Thomas process with negative Binomial distributions to govern the distribution of the number of points over the clusters of the underlying homogeneous point process.

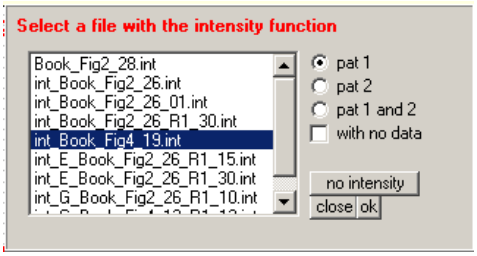
3.2.17 Variability in estimation of inhomogeneous summary statistics

Because estimation of the inhomogeneous summary statistics using the WM estimator uses an auxiliary (heterogeneous Poisson) pattern to estimate the edge correction it introduces a small stochastic variability. This variability can be easily assessed.

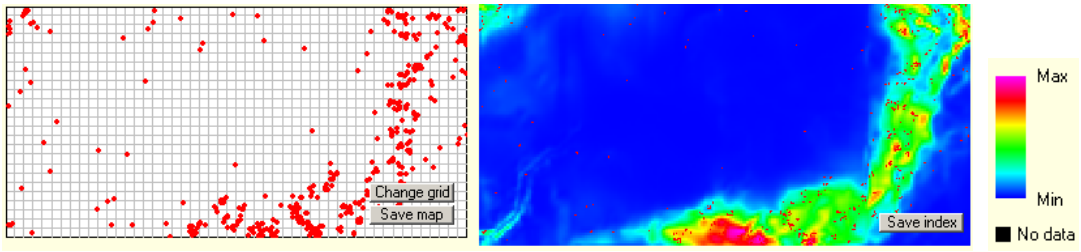
Example Book_Fig4_19_var.res

1. Execute *Programita*.
2. Highlight data file Book_Fig4_19b.dat you want to analyze in Input data file.
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 3 in the menu "Which method will you use"
6. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
7. Press button "**Calculate Index**"
8. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
9. Enable check box "**Inhom g and K**"

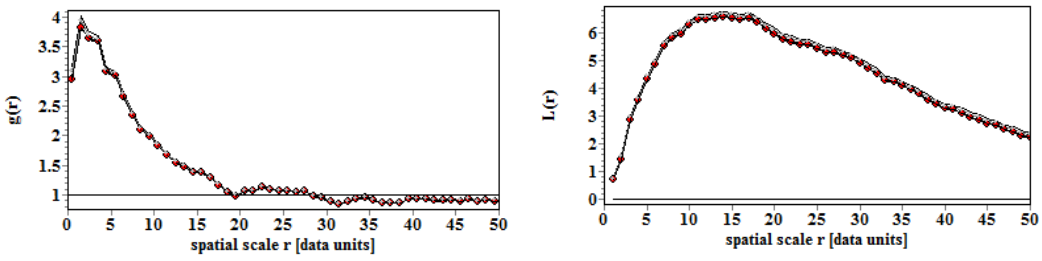
10. The window "Select a file with the intensity function" appears where you select the intensity file you want to use for estimation of the inhomogeneous second-order summary statistics (i.e., int_Book_Fig4_19.int). Select "**pat 1**" (because it is the intensity of pattern 1) and then click the small "**ok**" button:



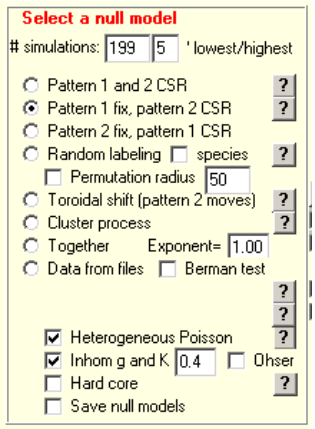
Programita now shows the intensity function and the pattern:



11. Specify the number of simulations of the null model (199 in the example) and the rule for the estimation of simulation envelopes (here the 5th lowest and highest values of the summary statistic of the 199 simulated null model data sets).
12. Select in the window Select a null model "**Pattern 1 fix, pattern 2 CSR**". This null model does therefore estimate 199 times the summary statistics of the observed pattern, thereby assessing the variability in the estimators of the inhomogeneous second-order summary statistics.
13. Click "**Calculate Index**". *Programita* now estimates the inhomogeneous summary statistics based on an auxiliary heterogeneous Poisson pattern with a default of dim1*dim2/4 = 1000*500/4 = 125000 points.
14. The variability in the estimators of the inhomogeneous second-order summary statistics is extremely small:

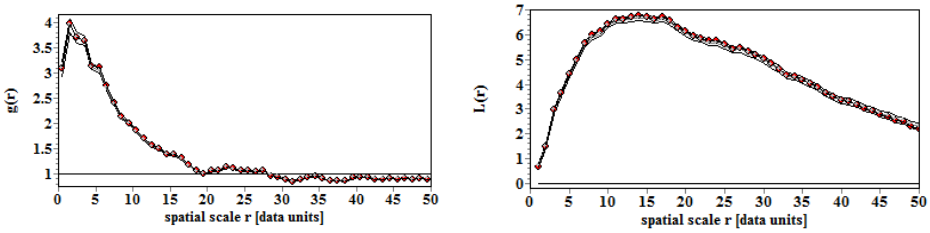


Example Book_Fig4_19_var04.res

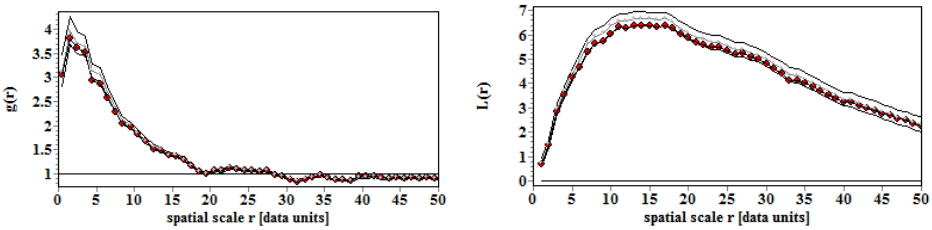


To investigate how the variability in the estimators of the inhomogeneous second-order summary statistics depends on the number of points of the auxiliary pattern change the number of points in the Select a null model window by giving a factor that increases/decreases the number of points. In the example we use the minimum (50,000 auxiliary points) which yields a factor 0.4.

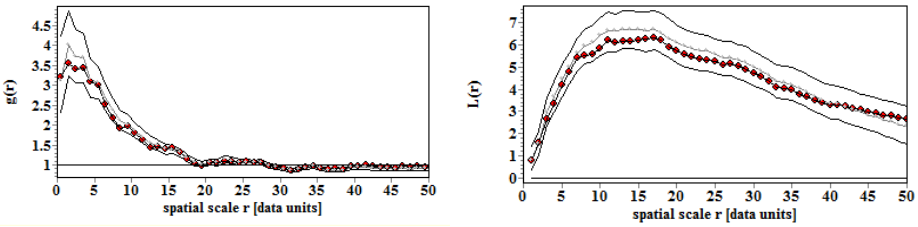
First, with 50,000 auxiliary points the variability in the estimators of the inhomogeneous second-order summary statistics is low:



Second, with 10,000 auxiliary points (cannot be selected in *Programita* because minimal number of auxiliary points is 50,000) the variability in the estimators of the inhomogeneous second-order summary statistics becomes notable:



Third, when using 1,250 auxiliary points the variability in the estimators of the inhomogeneous second-order summary statistics becomes unacceptably large:



When using the WM estimator for inhomogeneous second-order summary statistics it is a good idea to first check if the number of auxiliary points provides a good compromise between estimation speed and variability.

3.2.18 Hard core processes

Cluster processes show an elevated neighborhood density where the typical point has more neighbors nearby than expected under a CSR process. In contrast, point processes showing regularity (or hyperdispersion) have a reduced neighborhood density. In the extreme case there is a minimal distance $2r_0$ between two points. This corresponds in the simplest case to randomly distributed disks with radius r_0 which do not overlap (i.e., a hard core process).

The hard-core process has a pair correlation function

$$g(r) = \begin{cases} 0 & \text{for } r \le 2r_0 \\ 1 & \text{for } r > 2r_0 \end{cases}$$

which is zero for distances r smaller than the diameter $2r_0$ of the disk and one for larger distances.

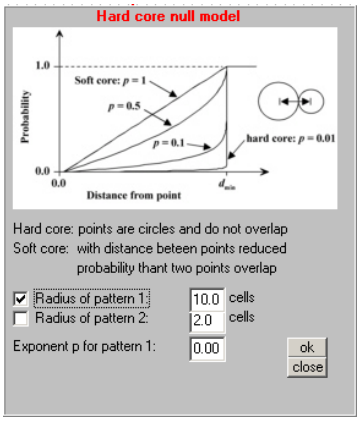
Hard core processes are often too simple to characterize inhibition processes producing observed point patterns in ecology and more complex Gibbs or Markov Point Processes are used that can consider interaction functions of different shape (see section 4.1.6.2 “Gibbs or Markov Point Processes” in Wiegand and Moloney (2014) or sections 3.6 and 6.5 in Illian et al (2008)). However, in general Gibbs or Markov Point Processes cannot be simulated in a straight forward way as for example cluster or heterogeneous Poisson processes. They are governed by the so-called location density function (a high-dimensional probability density function) which yields basically the likelihood of a given point configuration. Simulation of this point processes requires optimization techniques where points of an initial pattern are deleted and replaced by randomly drawn points, which are accepted if the new point configuration becomes more likely, given the location density function. *Programita* has not implemented this type of point processes. However, to provide you the possibility to simulate simple point patterns with hyperdispersion, *Programita* includes a simple algorithm to simulate a so-called “random sequential absorption” (RSA) process to produce hard core patterns.

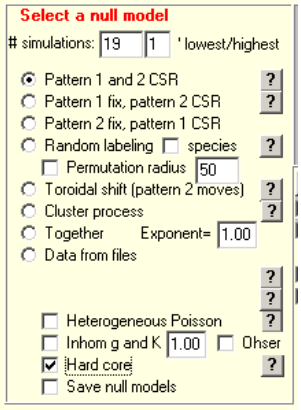
The RSA algorithm implemented in *Programita* is simple. It is constructed by placing iteratively and randomly points within an observation window W which are thought to be the centers of disks with radius r_0. If a newly placed disk overlaps with an already accepted disk, it is not accepted,and new points are placed in this way until the total number of points of the pattern is reached or until no further point can be placed because the pattern is “jammed”.

Example Book_Fig2_11.res (RSA inhibition process)

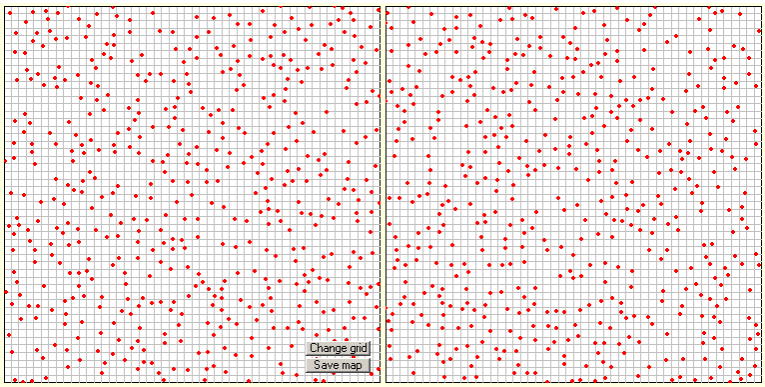
This pattern has been generated with a RSA algorithm to simulate non-overlapping disks with radius r_0 = 10m.

1. Execute *Programita*.
2. Highlight data file Book_Fig2_11.dat you want to analyze in Input data file.
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 3 in the menu "Which method will you use"
6. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
7. Click the button "change" in set maximal radius rmax to define the maximal scale r of the analysis and insert "100"
8. Press button "**Calculate Index**"
9. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
10. Select null model "Pattern 1 and 2 CSR" to start with the basic CSR algorithm.
11. Click checkbox "Hard core" and go to the window "Hard core null model" to define details of the RSA null model. Click "Radius of pattern 1" because you have a univariate pattern and provide the radius (10.0) in our case. To confirm settings click small "**ok"** button

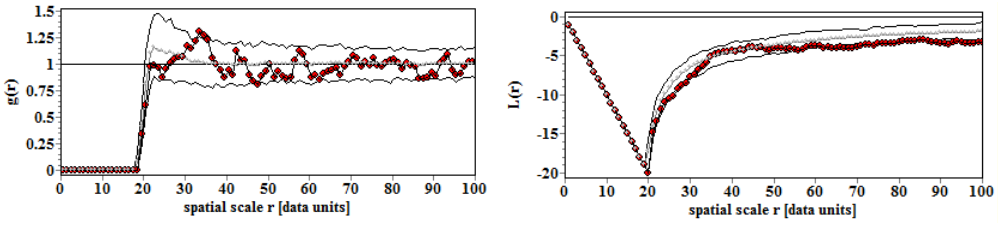




12. To simulate the point process press "**Calculate Index**". As expected, the simulated patterns look very similar to the observed pattern:

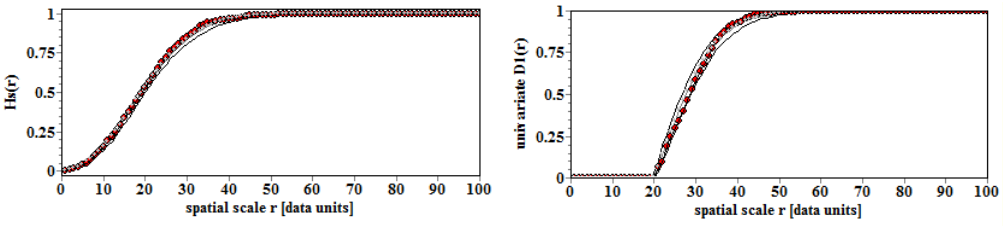


13. The pair correlation function and the *L*-function agree well with the simulated point process and show the typical "hard-core" shape:



Note however that the expectation of the RSA null model for the pair correlation function at distances slightly larger than the diameter $2r_0$ of the disk is not exactly one but somewhat larger. The reason for this is that in cases where already many points are placed the rejection rule causes acceptance of slightly more points close to an already placed point than farther away (because suitable gaps become scarce). As a consequence, we have a slight cluster effect.

14. The same is true for the spherical contact distribution and the nearest neighbor distribution function:

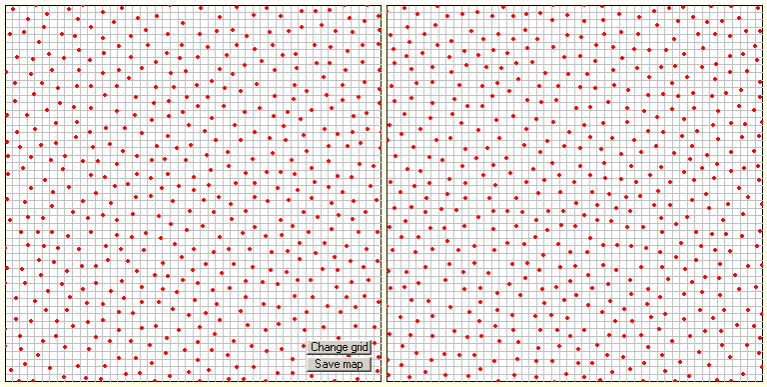


Example Book_Fig2_11jam.res (RSA inhibition process)

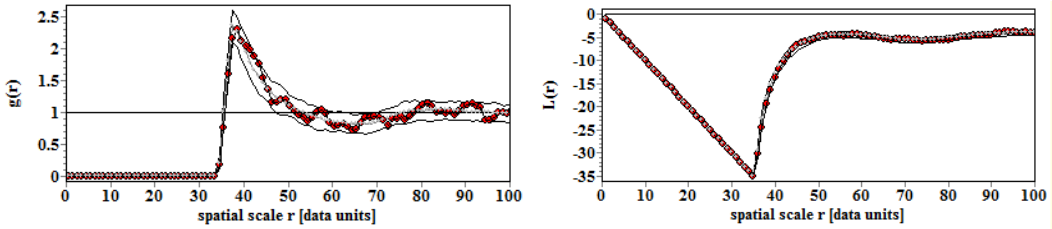
To see what happens if the pattern is close to jamming (i.e., no further points can be added) we increase the radius of the non-overlapping disks to r_0 = 18m. In this case the fraction of the observation 1000 × 1000 window covered by the disks yields $A_A = (500 \times \pi\, 18^2)/1000^2 = 0.509$. The maximum possible value of A_A for the RSA process yields $A_A = 0.547$ (which corresponds to a radius just below 19m). The file Book_Fig2_11_jam.dat is a realization of this point process with 500 points.

1. Execute *Programita*.
2. Highlight data file Book_Fig2_11jam.dat you want to analyze in Input data file.
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 3 in the menu "Which method will you use"
6. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
7. Click the button "change" in set maximal radius rmax to define the maximal scale r of the analysis and insert "100"
8. Press button "**Calculate Index**"

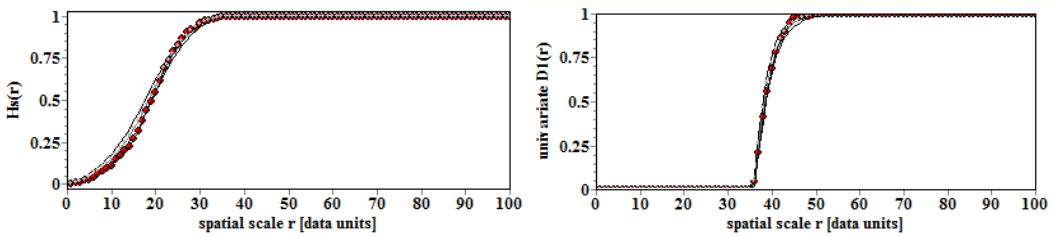
9. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
10. Select null model "Pattern 1 and 2 CSR" to start with the basic CSR algorithm.
11. Click checkbox "Hard core" and go to the window "Hard core null model" to define details of the RSA null model. Click "Radius of pattern 1" because you have a univariate pattern and provide the radius (18.0) in our case. To confirm settings click small **"ok"** button.
12. To simulate the point process press "**Calculate Index**". You notice that the pattern is close to jamming because the simulation takes more time. This is because the algorithm needs many attempts to find a place for the last points. The patterns are very regular patterns that almost yield a regular grid:



13. The pair correlation function at distances larger than the diameter $2r_0$ of the disk is now clearly elevated, it yields at the distance $r = 40$m a value of $g(r) \approx 2$ and then declines with one oscillation to the expected value of one. The closer the pattern to the jamming point, the higher the peak at distance $2r_0$. It is clear that point processes with the hard core pair correlation function shown above need more refined simulation methods:



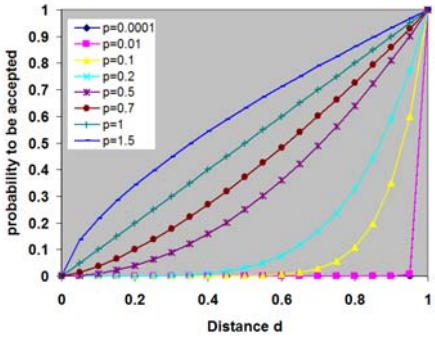
The distribution function of the distances to the nearest neighbor switches over a very narrow range between $r = 36$ and $r = 46$ from zero to one. That means that all points have approximately the same distance to the nearest neighbor:



3.2.19 Soft-core processes

A soft core pattern arises if the radiuses of the disks are not the same or if the disks have a certain probability to overlap that depends on the distance to the nearest neighbor. *Programita* extends the sequential RSA process explained above where all radiuses are the same to a simple point process that yields a soft-core pattern. *Programita* uses a probability p_{HC} of a provisional point to be accepted that varies between 0 and 1, depending on the distance d to the nearest (accepted) neighbor, and an exponent p that gives the degree of "softness":

$$p_{HC}(d) = \begin{cases} d^{1/p} & \text{for } d \le 2r_0 \\ 1 & \text{for } d > 2r_0 \end{cases}$$

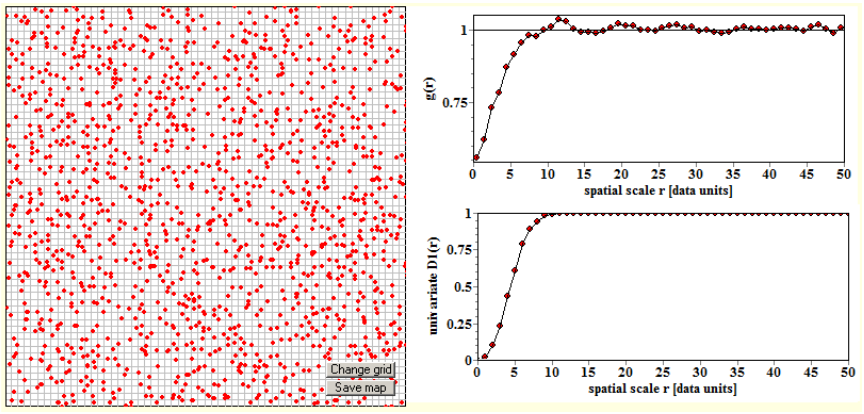


For $p = 0$, we obtain the RSA hardcore model, for $p > 0$ a soft core model, and for $p \to \infty$ we obtain CSR. The figure on the right shows how the rejection probability depends on the distance d to the nearest neighbor and the exponent p.

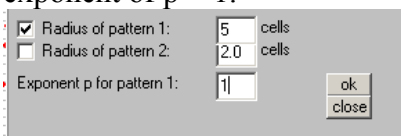
Example Book_Fig4_20rec.res (RSA soft core process)

Figure 4.20A in Wiegand and Moloney (2014) shows a 300 × 300m window from the BCI plot with all trees with a diameter larger than 20cm. The pair correlation function of this pattern shows a typical soft core shape (Fig. 4.20b). We analyze here instead a pattern with the same properties that was generated with pattern reconstruction based on the original BCI pattern for a 300 × 300m window.

1. Execute *Programita*.
2. Highlight data file Fig4_20rec.dat you want to analyze in Input data file.
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 3 in the menu "Which method will you use"
6. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
7. Press button "**Calculate Index**". As confirmed by the pair correlation function and the distribution function of the distances to the nearest neighbor, the pattern is a typical soft core pattern:

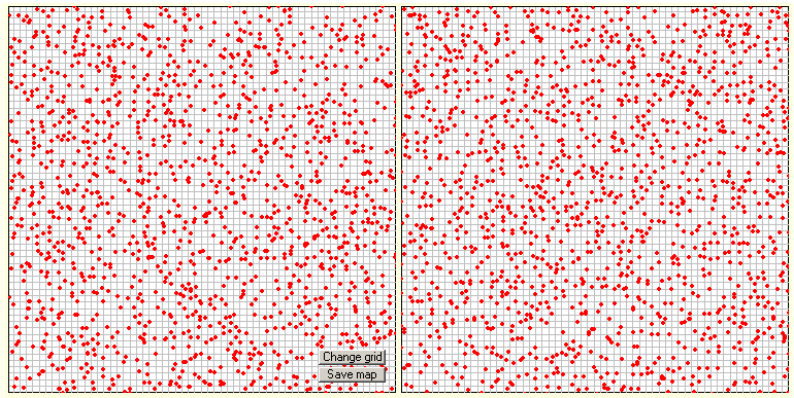


8. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
9. Select null model "Pattern 1 and 2 CSR" to start with the basic CSR algorithm.
10. To fit the soft core process (manually) to the data click checkbox "**Hard core**" and go to the window "Hard core null model" to define details of the RSA null model. Click "**Radius of pattern 1**" because you have a univariate pattern. The pair correlation function and the distribution function of the distances to the nearest neighbor shown above suggest a maximum diameter of the disks of 10m, thus use as first estimate of the radius of pattern 1 a value of 5m. Because the pattern is quite soft, start with an exponent of p = 1.



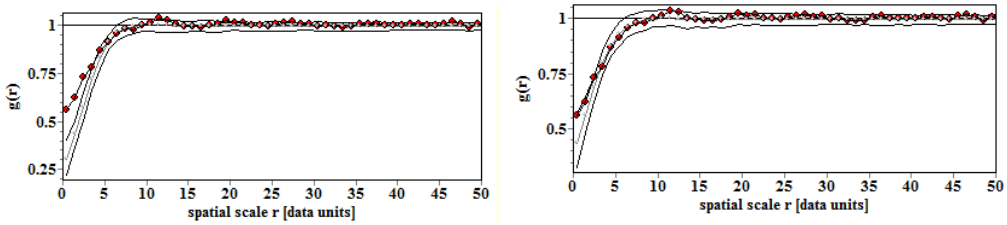
To confirm settings click small **"ok"** button.

11. To simulate the point process press "**Calculate Index**". The simulated patterns look very similar to the observed pattern:



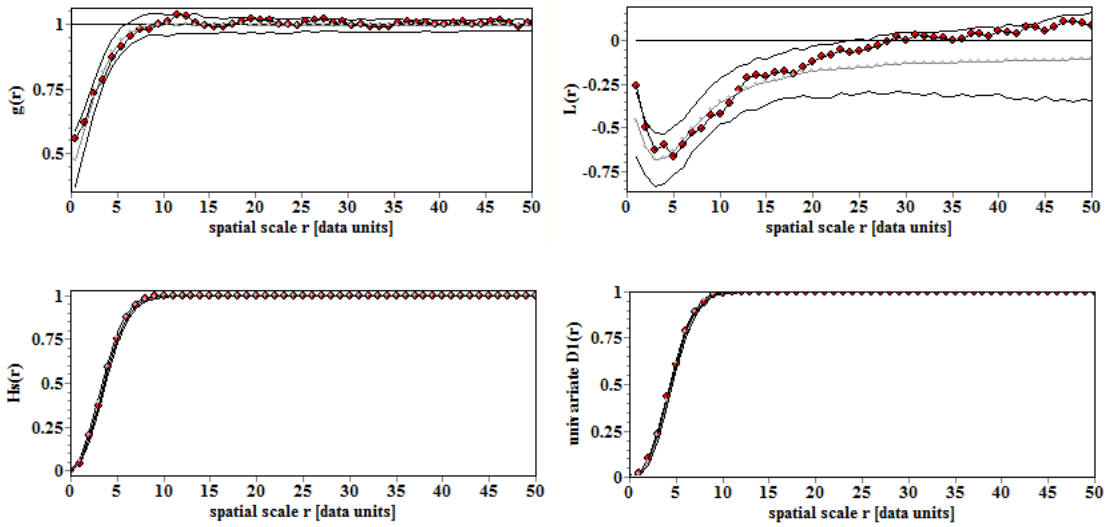
12. However, the pair correlation function is not well fitted at small distances, the observed pattern is softer. Therefore, use an exponent of say $p = 1.5$:

$r_0 = 5$m, $p = 1$ $\qquad$ $r_0 = 5$m, $p = 1.5$



13. An exponent p= 1.5 still yields a small underestimation of the pair correlation function at small distances. Select now $p = 1.7$

14. The soft core RSA point process with parameters r_0 = 5m and p = 1.7 provides a good fit for all of the important summary statistics:



The simulation confirmed that the trees with diameter larger than 20cm at BCI have a type of "zone of influence" of 5m (the radius r_0 = 5m) and the probability that a tree is inside this zone of influence of another tree declines almost with the square root of the distance to the focal tree.

4 Bivariate analysis in the standard mode

Bivariate analysis deals with a data type that comprises two types of points and is usually concerned with the characterization of the small-scale interaction structure between the two types of points although it can also be used to explore (larger-scale) co-occurrence patterns that may be influenced by habitat effects . A bivariate pattern is composed of two univariate component patterns which were created *a priori* by a different set of processes (e.g., two different species of trees in a forest). Chapter 4.2 in Wiegand and Moloney (2014) provides examples for the different analyses of bivariate patterns that are useful in ecology.

4.1 Getting started

4.1.1 Data preparation

Bivariate patterns comprise the coordinates of the two component point patterns. The data files for bivariate standard analysis must be an ASCII file with the *.dat extension and the following format (the example are the first lines of the file Book_Fig4_21a.dat):

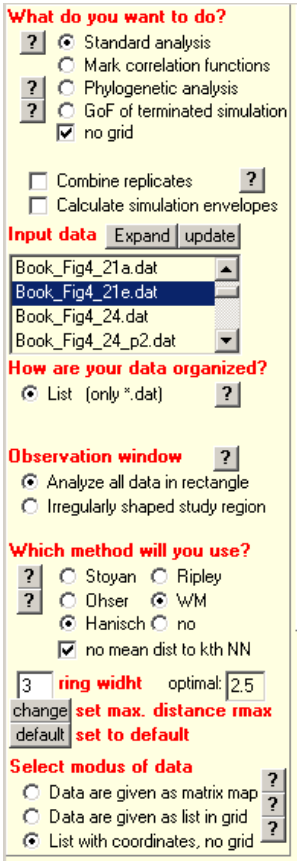
```
0  200  0  200   500
 0.2   56.4  1  0
 0.4  133.6  0  1
 0.4  144.6  0  1
 0.8   19.4  1  0
 1.0   49.6  0  1
 2.4   52.2  1  0
 2.6  177.4  0  1
 3.0  123.8  1  0
 3.8   37.0  0  1
 4.6  196.0  1  0
 5.0   83.2  1  0
 5.0  146.2  0  1
 5.2    4.6  1  0
…
```

where the first line gives the size of the observation window (200 × 200 units in the example) and the number of points in the pattern (= number of lines following the header). The first two columns are the coordinates, an entry "1" in the third column indicates that the point is of pattern 1 (i.e., a type 1 focal point) and an entry "1" in the fourth column indicates that the point is of pattern 2 (i.e., a type 2 point). The value of the third and the forth columns must be for the standard analysis mode "0 1" or "1 0", no larger numbers or "1 1" are allowed.

The data file must be a space or tab delimited ASCII file with the *.dat extension. If you use Excel, there is a simple, but obviously generally unknown, way of saving files of a given type with a given extension:

1. Prepare the data file in Excel following the instructions above
2. Then save as a tab delimited text file, but write "name.dat" for the name (usually you would only write name and end up with a file named name.txt). The quotation marks are important because they force Excel to save the comma delimited file under the name name.dat.

4.1.2 Steps of bivariate analysis in standard mode



Programita estimates for data files of this type several summary statistics based on estimators detailed in Illian et al. (2008) and Chapter 3 of Wiegand and Moloney (2014). The window **Which method will you use** allows you to specify the estimators.

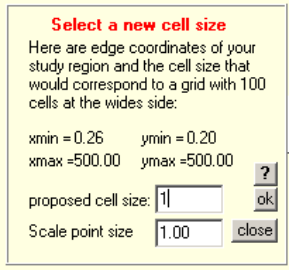
The standard analysis mode can be accessed with the following sequence of actions:

1. Select "**Standard analysis**" in window **What do you want to do?**
2. Highlight a data file in **Input data file** ("Book_4_21e.dat" in the example)
3. Select "**no grid**" in **What do you want to do?**

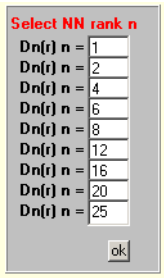
   The window **Select a new cell size** opens and allows you to provide a bin for your analysis given in units of your data. For example, if your data are in meter units and your observation window is 200 × 200m in size, an appropriate bin would be 1m. Press "**ok**" to confirm selection of the bin.

4. After selection of the bin ***Programita*** **suggests a ring width *dr*** based on equation 4.3.43 in Illian et al. (2008) [$dr = 0.2/\lambda^{0.5}$]. This equation provides a rough starting point for deciding on the ring width.

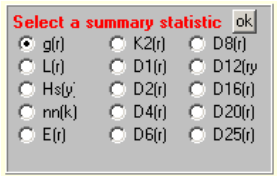
   The estimators of the pair correlation function implemented in the standard mode of *Programita* use a **default ring width of one bin** to obtain non-overlapping concentric rings. For reasons of computational efficiency you can then select only ring widths adding one bin in each direction, i.e., ring widths of 1, 3, 5, 7, … bins. You can change the ring width at the menu "**Which method will you use**". In the example file "Book_Fig4_21a.dat" with 250 type 2 points within a 200 × 200m observation window and a bin of 1m this yields a ring width of $dr = 2.5$. Thus select a ring width of 3.



5. Selecting the option "**no grid**" opens also a small window where you can select the desired rank k of the distribution functions $D^k(r)$ of the distances to the kth neighbor. Default is $k = 1, 2, 4, 6, 8, 12, 16, 20$, and 25. You can thus select the rank k of nine different functions $D^k(r)$. To confirm press the small **ok** button.

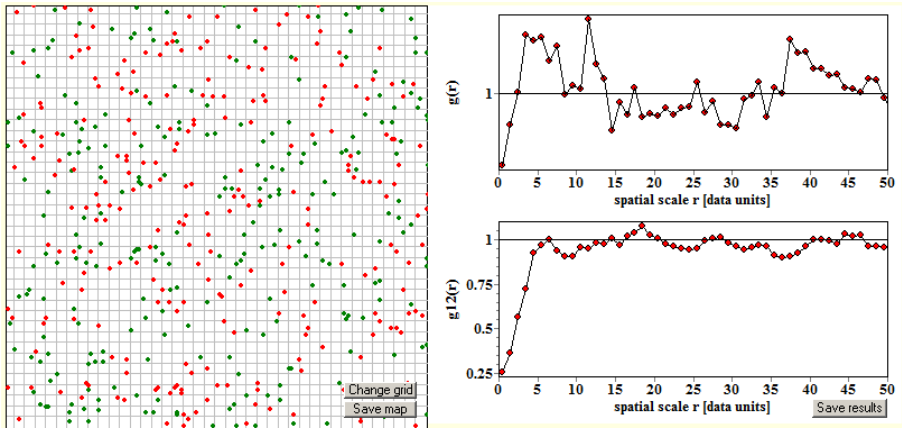


6. Press button "**Calculate Index**" and *Programita* estimates a variety of summary statistics of the uni- and bivariate data:



   - $g(r)$: pair correlation function
   - $L(r)$: L-function,
   - $H_s(r)$: the spherical contact distribution (only univariate)
   - $nn(k)$ the mean distance to the kth neighbor
   - $E(r)$ the probability that a point has no neighbor at distance within distances (r - 0.5, r + 0.5)
   - $K2(r)$ the $K2$ function
   - $D^k(r)$, the kth nn distribution functions, here with k = 1, 2, 4, 6, 8, 12, 16, 20, and 25

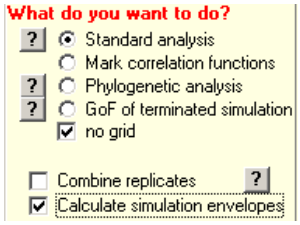
   The screen shows on the left the bivariate pattern Book_4_21e.dat with type 1 points (red) and type 2 points (green). On the right shown are the selected summary statistics. The graph for the (partial) univariate summary statistics (i.e., the analysis of only type 1 points) is shown on the top, and the bivariate summary statistics on the bottom:



   Note that the bivariate summary statistics count type 2 points (green) around type 1 points (red). Thus, points of pattern 1 are the focal points.

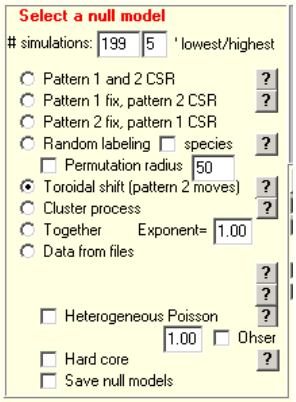
   To view the different summary statistics select the respective radio button and then the small "**ok**" button.

7. The next step is to select a **null model or point process** model implemented in *Programita*. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.



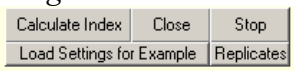
   A window will open that allows you to select a null model. In the example, we select "**Toroidal shift**". In this case pattern 2 is shifted as a whole a random vector and points falling outside the observation window are wrapped following torus geometry.

Here you can specify the **number of simulations of the null model** (199 in the example) and the rule for the estimation of **simulation envelopes** (here the 5th lowest and highest values of the summary statistic of the 199 null model data sets).

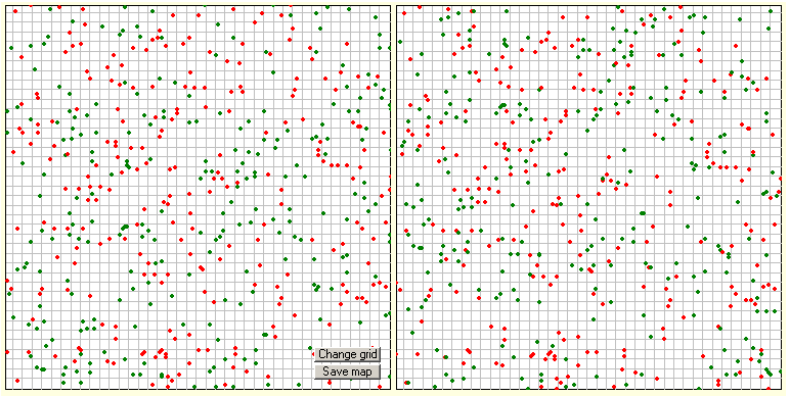


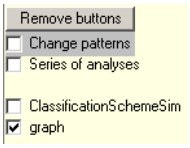
The checkbox "**Save null models**" allows you to save the patterns generated by the null model as "name_n.dat"

If all settings are specified; press "**Calculate Index**" and *Programita* conducts the simulations of the null model.



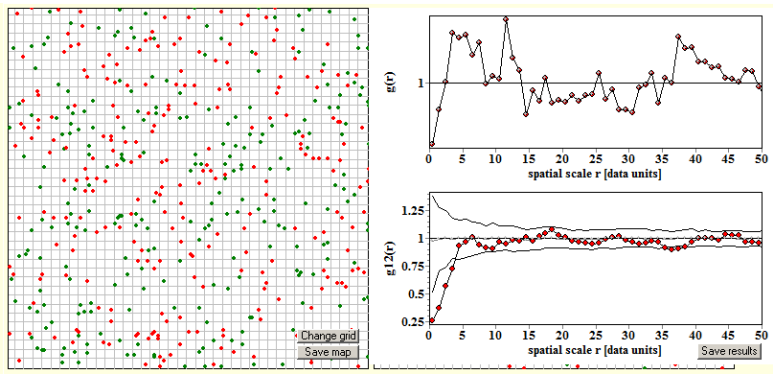
8. *Programita* shows the original point pattern being analyzed (left or top plot), and patterns of the Monte Carlo simulations of the null model (on the right or bottom) used for constructing the simulation envelopes and the GoF test.





The simulation is quicker if *Programita* does not show the plots of all simulated data. You can not show the graphs by disabling the checkbox "**graph**" at the bottom right.

9. After the simulations of the null model the figure with the simulated patterns of the null model disappears, and a figure with the result of the analysis appears:



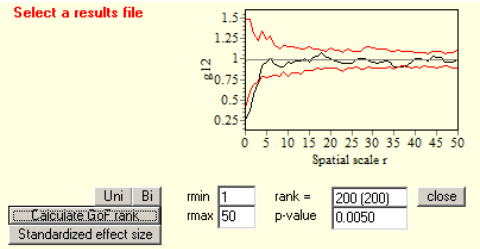
10. The top (or left) figure shows generally the results of the univariate analysis and the bottom (right) figure shows the results of the bivariate analysis. The null model did only displace the type 2 points, therefore no simulation envelopes appear for the univariate results.

11. To save **the results of the analysis for a particular summary statistics** press the button Save results in the result graph for the bivariate analysis. *Programita* then generates a *.res file [e.g., "**name.res**" where "name" is a name] with the summary of the results and the settings of the analysis, and a *.env file with the detailed results of the summary statistic for the data and the simulations of the null model (the *.env file is named for example for the bivariate pair correlation function "g12(r)_name.env"). The *.env file can be used for the GoF test.

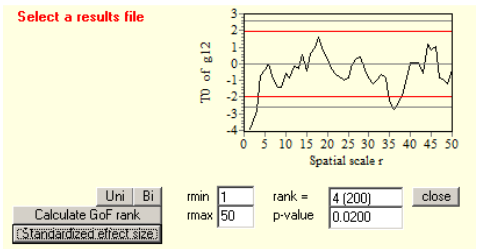
12. To conduct the GoF test check the small checkbox "**GoF**" that appears top right on the window "Select a null model". After enabling the check box a window appears where you need to click "**Uni**" or "**Bi**", depending if the analysis of interest is uni- or bivariate, respectively. Select "**Bi**" since the analysis was bivariate.

13. A small graph with the observed summary statistic and the lowest and highest values of the null model appears. Provide now the distance interval (rmin, rmax) to be tested and click "**Calculate GoF rank**" for the GoF test based on the Cramer-von Mises type test statistic u_i (Loosemore and Ford 2006). The rank and the associated P-value are then provided (left):

GoF test based on test statistic u_i



GoF test based on T test with test statistic T_0:



If you click "**Standardized effect size**" the GoF test based on a T-test is conducted.

The graph shows the standardized effect size $T_0(r)$ for different distances r and the critical bands for $P = 0.05$ (red) and $P = 0.01$ (grey). Basically, the $T_0(r)$ shown in the small graph is a transformation of the observed pair correlation function $g_{12}(r)$ with 5% simulation envelopes having constant values of -1.96 and 1.96. The latter assumes that the values of the summary statistics at distance r, taken from different simulations of the null model, are normally distributed.

The P-value of the two-sided test with test statistic T_0 over the distance interval (r_{min}, r_{max}) is also shown (see equation 2.10 in Wiegand and Moloney 2014).

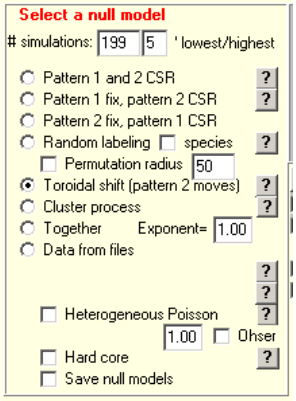
4.2 Methods for bivariate standard analysis

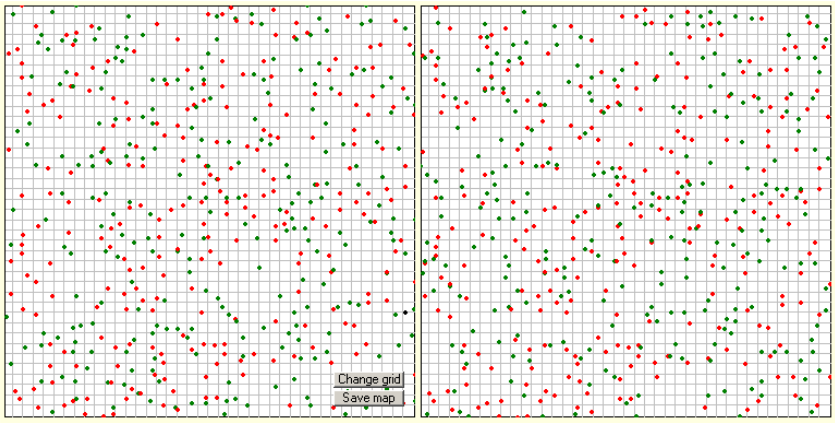
The following examples present step-by-step instructions for the most important bivariate analyses. We start with null models for independence. Devising a null model for independence for bivariate patterns is a highly non-trivial issue because a test of independence must maintain the univariate spatial structures in the two component patterns. *Programita* contains several bivariate null models that can be used as approximation of the independence null model.

4.2.1 Toroidal shift (Book_Fig4_21a.res)

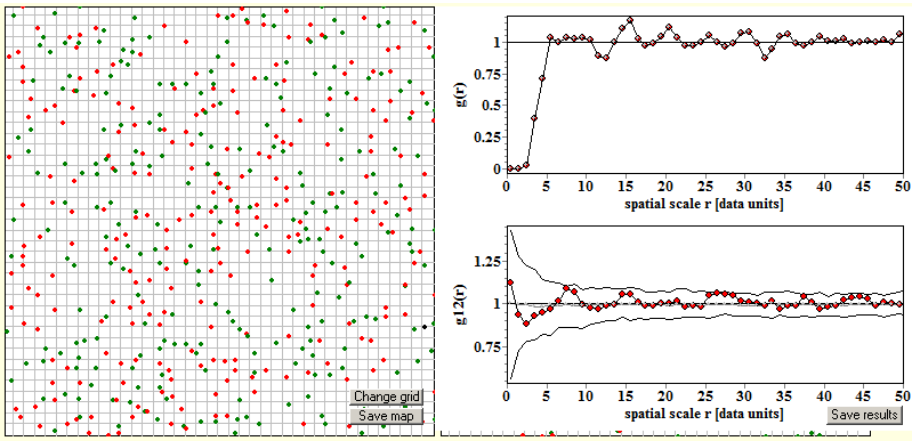
Section 4.2.1 in Wiegand and Moloney (2014) deals with different approaches to test for independence between the two component patterns of a bivariate pattern. An early non-parametric solution is the toroidal shift null model where pattern 2 is shifted as a whole a random vector against pattern 1 which is fixed (see section 4.2.1.1 "The Toroidal Shift Null Model" in Wiegand and Moloney 2014). The parts of the pattern that are shifted outside the observation window re-appear following torus geometry. If one pattern is antecedent (e.g., the pattern of saplings relative to adult trees) the antecedent pattern (e.g., adults) should be selected as the fixed pattern 1 and the other randomized. If no pattern is antecedent, two tests should be conducted switching the role of type 1 and 2.

1. Execute *Programita*.
2. Highlight data file Book_Fig4_21a.dat you want to analyze in Input data file.
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 3 in the menu "Which method will you use"
6. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
7. Press button "**Calculate Index**".
8. Click the checkbox "**Calculate simulation envelopes**" in the menu "What do you want to do?" on the top left of the interface.
9. Specify the **number of simulations of the null model** (199 in the example) and the rule for the estimation of **simulation envelopes** (here the 5th lowest and highest values of the summary statistic of the 199 null model data sets).
10. Select "**Toroidal shift**".
11. Press button "**Calculate Index**" and *Programita* shows the observed pattern (left) and the null model pattern (right):



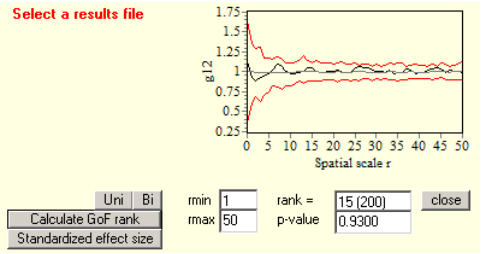


14. After the simulations of the null model the figure with the simulated patterns of the null model disappears, and a figure with the result of the analysis appears:

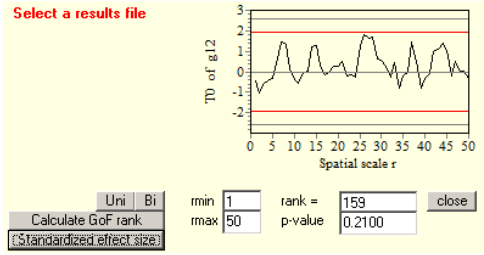
15. 

The pair correlation function is fully within the simulation envelopes. This is also confirmed by the GoF tests over the distance interval 1 - 50m:

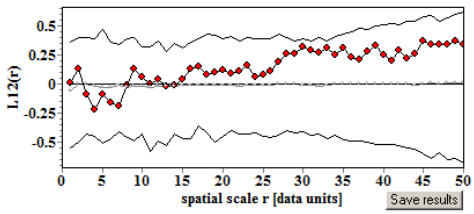
GoF test based on test statistic u_i

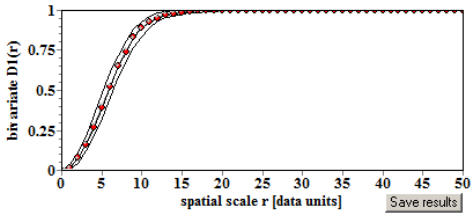


GoF test based on T test with test statistic T_0:



16. The other summary statistics are also well inside the simulation envelopes:

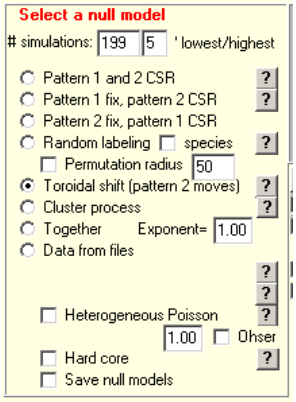




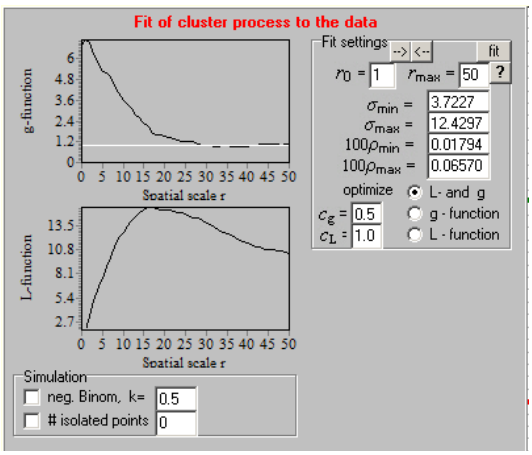
4.2.2 Pattern 2 Thomas process (Book_Fig4_24.res)

You can also fit a parametric point process to one of the component patterns and use the realizations of the fitted point process as null model patterns for independence (see section "4.2.1.2 Parametric Point-Process Models" in Wiegand and Moloney 2014). In this case you define this pattern as pattern 2, fix the other pattern, and randomize pattern 2 following this point process. For a simple Thomas process this procedure is directly implemented in *Programita.* Otherwise, conduct first a univariate analysis and save the patterns generated by the fitted univariate Thomas sprocess. They can then be used as null model patterns for pattern 2 using the "from file" option. Here we show the null model where pattern 1 is fixed and a Thomas process fitted to pattern 2.

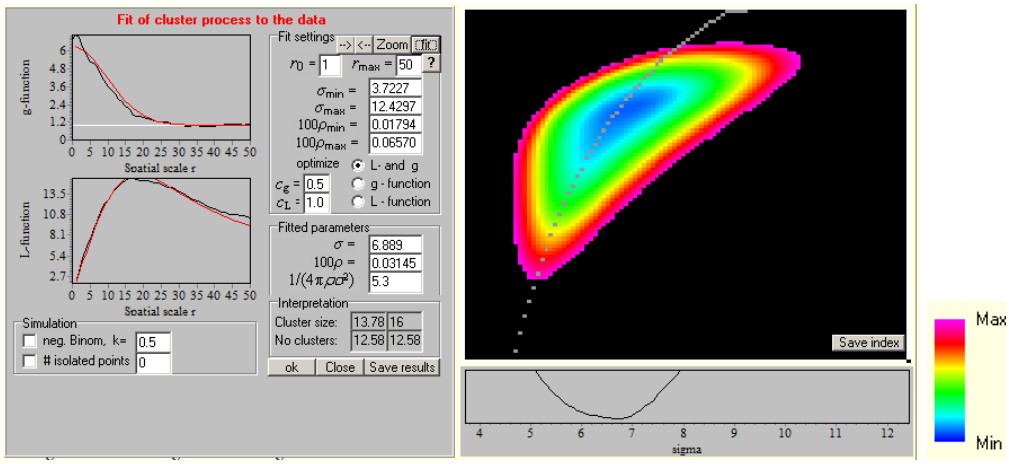
1. Execute *Programita*.
2. Highlight data file Fig4_24a.dat you want to analyze in Input data file.
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 3 in the menu "Which method will you use"
6. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
7. Press button "**Calculate Index**".
8. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?".
9. A window will open that allows you to select a null model. Here you can specify the **number of simulations of the null model** (199 in the example) and the rule for the estimation of **simulation envelopes** (here the 5th lowest and highest values of the summary statistic of the 199 null model data sets).
10. In the example, select "**Cluster process**" and then select "**Pattern 1 fixed, patter 2 cluster**" in the window "Null models":



11. Now the interface for fitting appears:



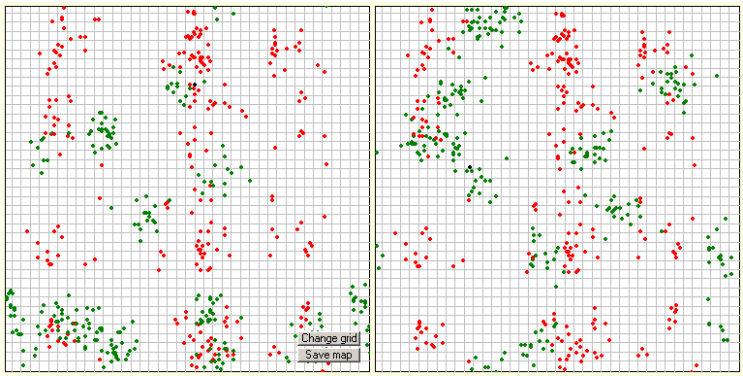
12. Select the radio button "**L- and g**" to use the *g*- and *L*-function for fitting. The default settings over distance interval 1 to 50m (r_0 = 1 r_{max} = 50).
13. Click the button "**fit**" and *Programita* fits the two parameters ρ and σ of the Thomas process to the pattern. Note that ρA yields the number of clusters and 2σ the approximate radius of the "typical cluster". To iteratively encircle the parameter space around the minimum in the σ-ρ parameter space click "**Zoom**" and "**Fit**":



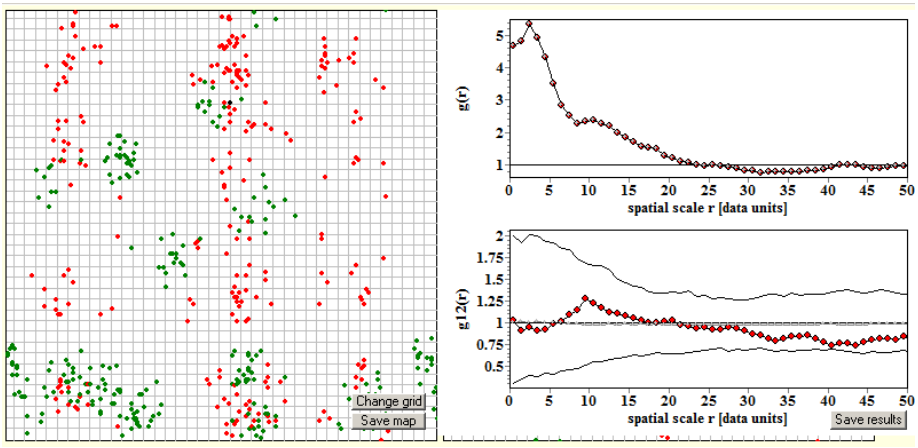
The graph on the right shows the deviation between observed summary statistic (here the pair correlation and the *L*-function) and that predicted by Thomas process over the σ-ρ parameter space indicated by σ_{min}, σ_{max}, $100\rho_{min}$, and $100\rho_{min}$. There is a clear minimum at $\sigma = 6.9$ and $\rho A = 12.6$ clusters. The pattern was generated with $\sigma = 6.2$ and $\rho A = 12.5$ clusters.

If you are satisfied with the fit, press the small "**ok**" button in the "Fitted parameter" section of the fitting window.

14. Press button "**Calculate Index**" and *Programita* shows the observed pattern (left) and the null model pattern (right):



17. After the simulations of the null model the figure with the simulated patterns of the null model disappears, and a figure with the result of the analysis appears:



As expected, the bivariate pair correlation function and the other summary statistics are fully within the simulation envelopes.

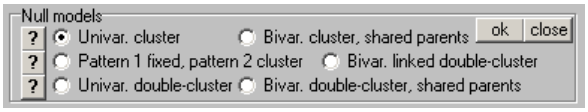
4.2.3 Pattern 2 from file (Book_Fig4_24file.res)

In this example we use null model patterns for pattern 2 that were generated previously with *Programita* fitting a Thomas process to the data of pattern 1. Thus, first fit a Thomas process to pattern 2 and save the patterns generated by the Thomas process. Second, read these files into *Programita* as null model for pattern 2.

1) Fit a Thomas process to pattern 2 (data Book_Fig4_24_p2.dat).

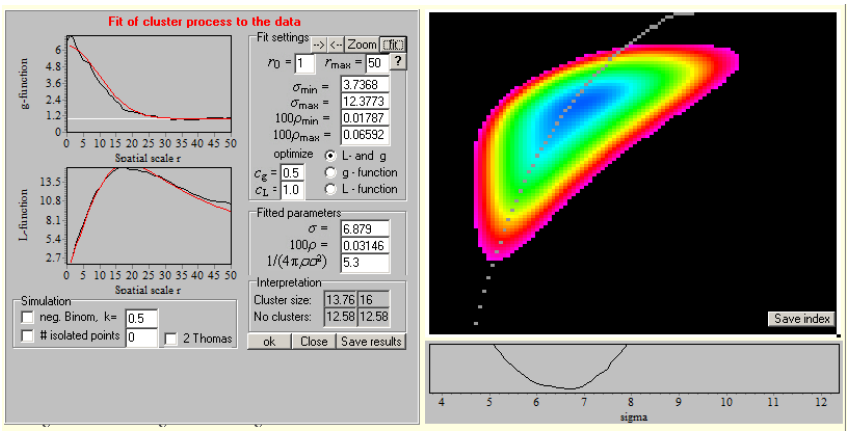
1. Execute *Programita*.
2. Highlight data file Book_Fig4_24_p2.dat you want to analyze in Input data file.
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 3 in the menu "Which method will you use"
6. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
7. Press button "**Calculate Index**".

15. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
16. A window will open that allows you to select a null model. Here you can specify the **number of simulations of the null model** (19 in the example).
17. In the example, select "**Cluster process**" and then select "**Univar. cluster**" in the window "Null models":



8. Now the interface for fitting appears.

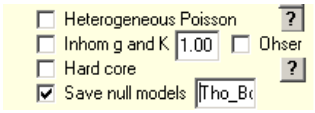
To iteratively encircle the parameter space around the minimum in the σ-ρ parameter space click "**Zoom**" and "**Fit**". As expected, the best fit is similar to that in the previous example:



There is a clear minimum at $\sigma = 6.9$ and $\rho A = 12.6$ clusters. The pattern was generated with $\sigma = 6.2$ and $\rho A = 12.5$ clusters.

If you are satisfied with the fit, press the small "**ok**" button in the "Fitted parameter" section of the fitting window.

9. To save the patterns generated by the Thomas process click "Save null model" and provide name of null model files (Tho_Book_Fig4_24_p2)

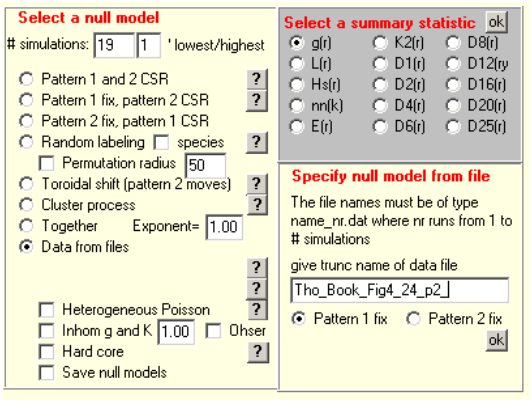


10. Press button "**Calculate Index**" and *Programita* generates the null model patterns Tho_Book_Fig4_24_p2_1.dat, Tho_Book_Fig4_24_p2_2_dat,….

2) Conduct analysis with the null model from file (Book_Fig4_24file.res)

11. Execute *Programita*.
12. Highlight data file Book_Fig4_24.dat you want to analyze in Input data file
13. Select "**no grid**" in What do you want to do?
14. Select bin of 1m window Select a new cell size
15. Select a ring width of 3 in the menu "Which method will you use"
16. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
17. Press button "**Calculate Index**"
18. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.

19. Select "**Data from file**" in the window "Select a null model".

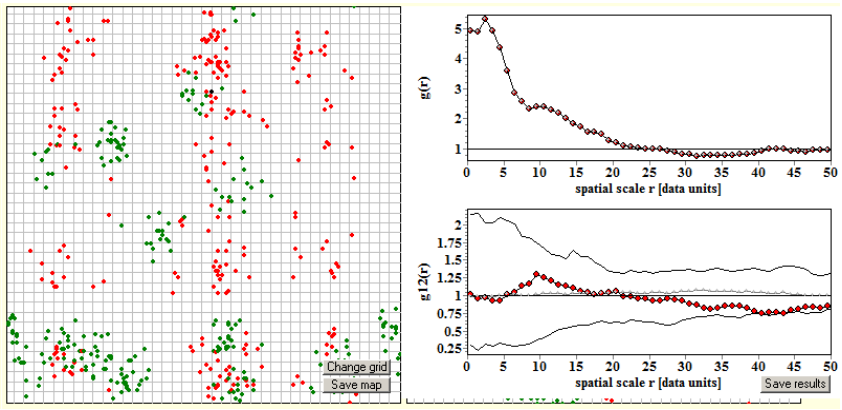


20. Insert the name trunk of the null model files (Tho_Book_Fig4_24_p2_) in the window "Specify null model files from file" that opens to read the null model files Tho_Book_Fig4_24_p2_1.dat, Tho_Book_Fig4_24_p2_2.dat, …

    Click also the radio button "**Pattern 1 fix**". This means that the null model files are used for pattern 2.

    To finish click the small **ok** button in the window "Specify null model files from file". Specify the number of simulations of the null model (19 in the example) and the rule for the estimation of simulation envelopes (here the 1th lowest and highest values of the summary statistic of the 19 simulated null model data sets).

21. If all settings are specified, press the button "**Calculate Index**" and *Programita* conducts the simulations of the null model.
22. After the simulations of the null model the figure with the simulated patterns of the null model disappears, and a figure with the result of the analysis appears:

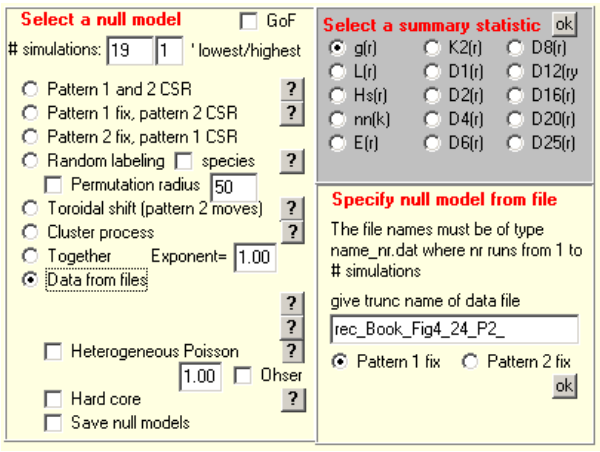


    As in the previous example, the bivariate pair correlation function and the other summary statistics are fully within the simulation envelopes.

4.2.4 Pattern reconstruction (Book_Fig4_24rec.res)

In this example we use null model patterns for pattern 2 that were generated previously with the pattern reconstruction software (Wiegand et al. 2013). *Programita* can read these files and use it for the null model.

1. Execute *Programita*.
2. Highlight data file Book_Fig4_24.dat you want to analyze in Input data file
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 3 in the menu "Which method will you use"
6. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
7. Press button "**Calculate Index**"
8. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
9. Select "**Data from file**" in the window "Select a null model".



10. Insert the name trunk of the null model files (rec_Book_Fig4_24_p2_) in the window "Specify null model files from file" that opens to read the null model files rec_Book_Fig4_24_p2_1.dat, rec_Book_Fig4_24_p2_2.dat, …

    Click also the radio button "**Pattern 1 fix**". This means that the null model files are used for pattern 2.

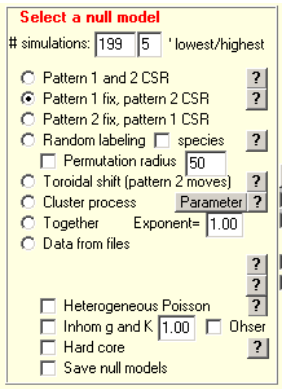
    To finish click the small **ok** button in the window "Specify null model files from file". Specify the number of simulations of the null model (19 in the example) and the rule for the estimation of simulation envelopes (here the 1th lowest and highest values of the summary statistic of the 19 simulated null model data sets).

11. If all settings are specified, press the button "**Calculate Index**" and *Programita* conducts the simulations of the null model.
12. After the simulations of the null model the figure with the simulated patterns of the null model disappears, and a figure with the result of the analysis appears.

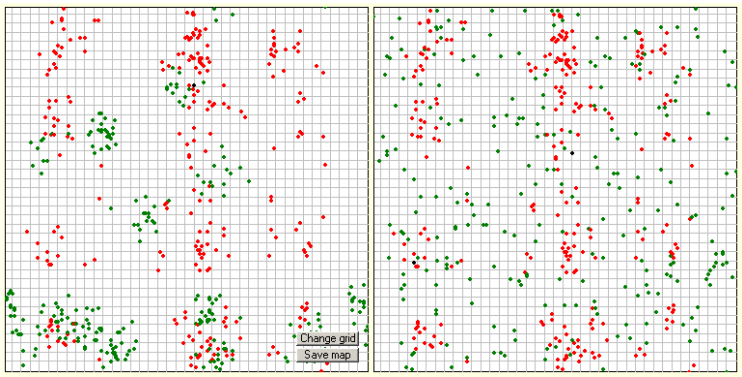
4.2.5 Pattern 2 CSR (Book_Fig4_24CSR2.res)

In this example we use the CSR null model for pattern 2. Note that this is not a suitable null model for testing for independence if pattern 2 shows clustering or hyperdispersion.

1. Execute *Programita*.
2. Highlight data file Book_Fig4_24.dat you want to analyze in Input data file
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 3 in the menu "Which method will you use"
6. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
7. Press button "**Calculate Index**"
8. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
9. Select "**Pattern 1 fix, pattern 2 CSR**" in the window "Select a null model".
10. Specify the **number of simulations of the null model** (199 in the example) and the rule for the estimation of **simulation envelopes** (here the 5th lowest and highest values of the summary statistic of the 199 null model data sets).



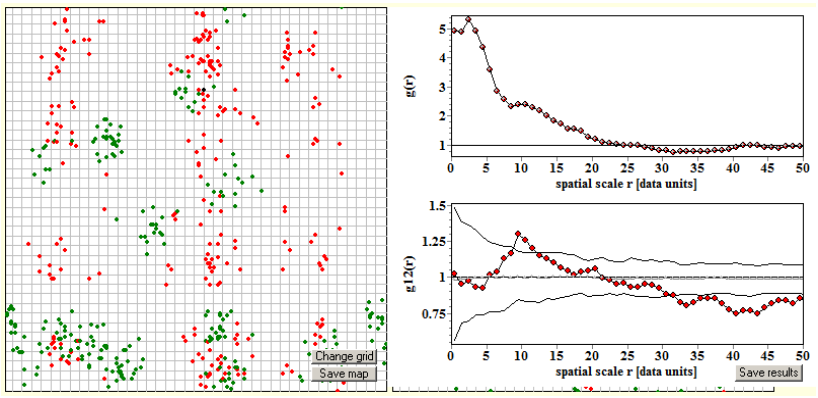
11. If all settings are specified, press the button "**Calculate Index**" and *Programita* conducts the simulations of the null model. The null model patterns are shown on the right, the observed pattern on the left:



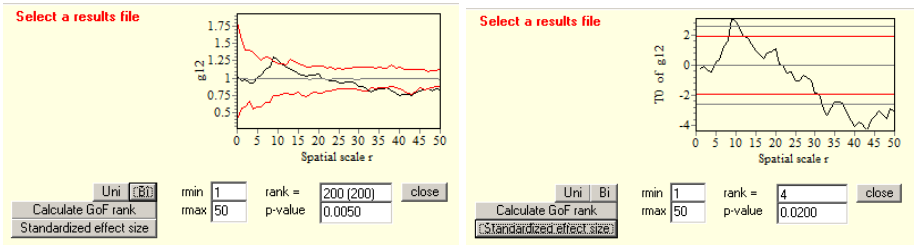
It is clear that the null model (i.e., the green points) does not conserve the observed cluster structure.

12. After the simulations of the null model the figure with the simulated patterns of the null model disappears, and a figure with the result of the analysis appears. Note that no simulation envelopes are shown for the univariate case because pattern 1 is fixed.

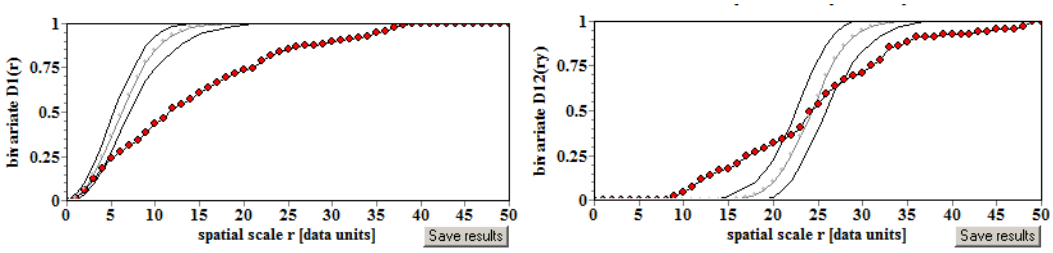
    The simulation envelopes for the bivariate summary statistics are substantially narrower than that of the diverse null models that conserved the observed spatial structure of pattern 2.



13. The GoF tests show that the departures of the bivariate pair correlation function are significant:



14. Especially the nearest neighbor distribution functions show strong departures from the null model:



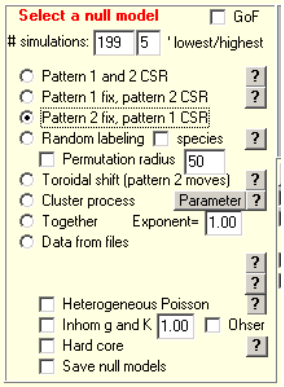
    The nearest type 2 neighbor of type 1 points in the null model is usually much closer than in the observed data. This is because pattern 2 was clustered in the data.

    The observed bivariate pattern contains non-random spatial structure, but this structure is caused by the univariate clustering of the component patterns which accidently created a point configuration which is difficult to reproduce with the null model where pattern 2 is CSR. If we conserve the observed univariate structures we see that such point configurations can arise just by chance.

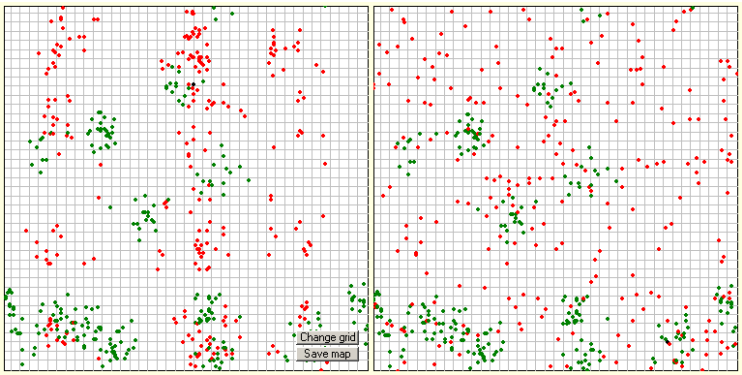
4.2.6 Pattern 1 CSR (Book_Fig4_24CSR1.res)

In this example we use the CSR null model for pattern 1 whereas pattern 2 is unchanged. Note that this is not a suitable null model for testing for independence if pattern 1 shows clustering or hyperdispersion.

1. Execute *Programita*.
2. Highlight data file Book_Fig4_24.dat you want to analyze in Input data file
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 3 in the menu "Which method will you use"
6. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
7. Press button "**Calculate Index**"
8. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
9. Select "**Pattern 2 fix, pattern 1 CSR**" in the window "Select a null model".
10. Specify the **number of simulations of the null model** (199 in the example) and the rule for the estimation of **simulation envelopes** (here the 5th lowest and highest values of the summary statistic of the 199 null model data sets).

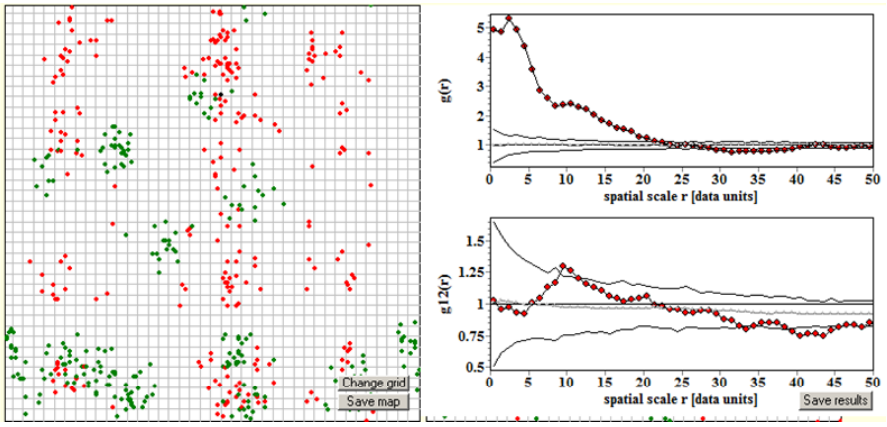


11. If all settings are specified, press the button "**Calculate Index**" and *Programita* conducts the simulations of the null model. The null model patterns are shown on the right, the observed pattern on the left:

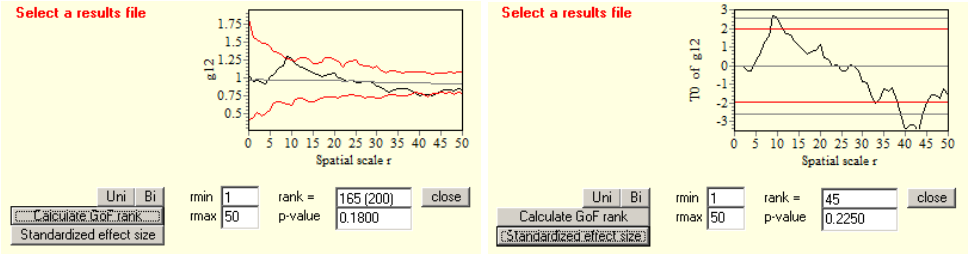


It is clear that the null model (i.e., the red points) does not conserve the observed cluster structure.

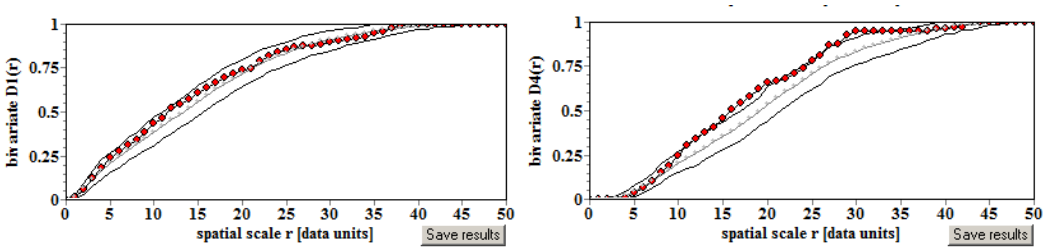
12. After the simulations of the null model the figure with the simulated patterns of the null model disappears, and a figure with the result of the analysis appears. Note that there is now a result for the univariate analysis because pattern 1 was randomized.



13. The simulation envelopes are narrower than for the diverse null models that conserved the observed spatial structure of pattern 2, but the GoF tests show that the departures of the bivariate pair correlation function are not significant:



14. Interestingly, the nearest neighbor distribution functions show no departures from the null model, but those with higher neighborhood ranks such as the 4th nearest neighbor:

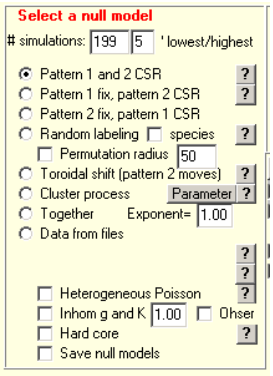


However, as expected there is a strong departure from the CSR null model in the univariate analysis.

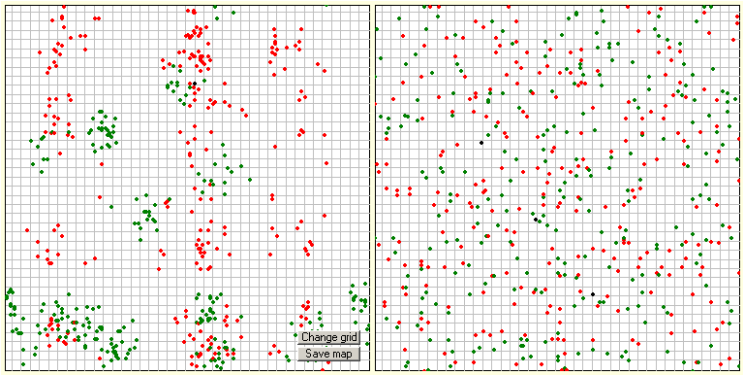
4.2.7 Pattern 1 and 2 CSR (Book_Fig4_24CSR12.res)

In this example we use the CSR null model for both component patterns. Note that this is not a suitable null model for testing for independence if pattern 1 and 2 shows clustering or hyperdispersion.

1. Execute *Programita*.
2. Highlight data file Book_Fig4_24.dat you want to analyze in Input data file
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 3 in the menu "Which method will you use"
6. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
7. Press button "**Calculate Index**"
8. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
9. Select "**Pattern 1 fix, pattern 2 CSR**" in the window "Select a null model".
10. Specify the **number of simulations of the null model** (199 in the example) and the rule for the estimation of **simulation envelopes** (here the 5th lowest and highest values of the summary statistic of the 199 null model data sets).

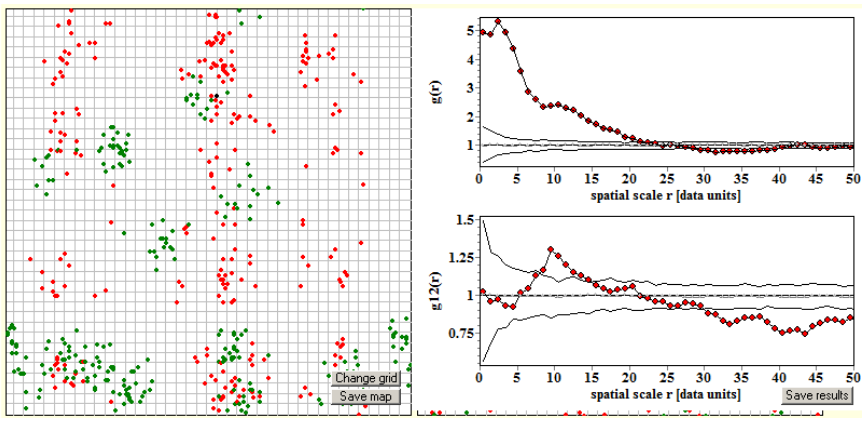


11. If all settings are specified, press the button "**Calculate Index**" and *Programita* conducts the simulations of the null model. The null model patterns are shown on the right, the observed pattern on the left:

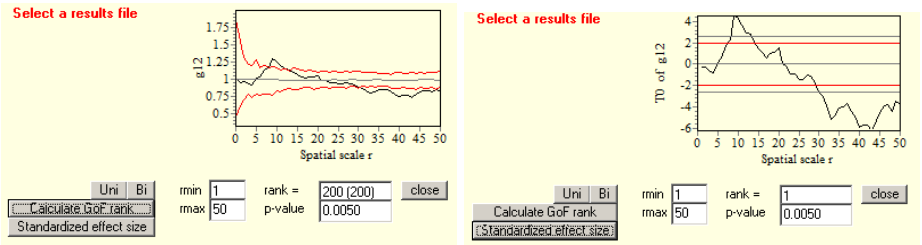


It is clear that the null model (i.e., the red and green points) does not conserve the observed cluster structure.

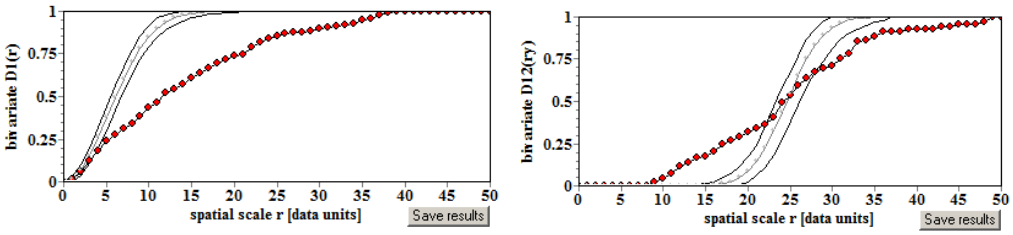
12. After the simulations of the null model the figure with the simulated patterns of the null model disappears, and a figure with the result of the analysis appears. The simulation envelopes are substantially narrower than for the diverse null models that conserved the observed spatial structure of pattern 2.



13. As a consequence, the GoF tests show that the departures of the bivariate pair correlation function are highly significant:



Especially the nearest neighbor distribution functions show strong departures from the null model:



The nearest neighbor in the null model is usually much closer than in the observed data. This is because the observed patterns 1 and 2 were clustered.

The observed bivariate pattern contains non-random spatial structure, but this structure is caused by the univariate clustering of the component patterns which created a point configuration which is impossible to reproduce with the null model where patterns 1 and 2 are CSR. However, if we conserve the observed univariate structures we see that such point configurations can arise just by chance.

4.2.8 Classification scheme

The classification scheme is used for exploring the overall spatial association structure of species pairs in a multivariate pattern. The goal of this analysis is to determine how the individuals of a species j were distributed within local neighborhoods of individuals of a species i, and this analysis is repeated for all pairs i-j of the community. The scheme is basically a summary of all pairwise bivariate analyses at a given neighborhood scale *r* which are possible for a community. Sections 4.2.2.2- 4.2.2.4 and section 4.3.1.1 in Wiegand and Moloney (2014) provide details on the scheme.

The bivariate analysis of the scheme can use two different null models; first, one pattern is kept fixed and the other pattern is randomized following CSR, and second, one pattern is kept fixed and the other pattern is randomized using pattern reconstruction. The analysis using the CSR null model reveals how frequently the two different species were in close contact and therefore have the opportunity to interact whereas the second case explores how frequent different departures from independence occur for the pairs of species at a given neighborhood *r*.

The scheme relies on two summary statistics, the bivariate *K* function $K_{ij}(r)$ and the bivariate distribution function $D_{ij}(r)$ of the distances to the nearest neighbor and their expectations under the two null models [i.e., $D_{ij}(r) = 1- \exp(-\lambda_j \pi r^2)$ and $K_{ij}(r)=\pi r^2$].

In case of homogeneous patterns departures in $D_{ij}(r)$ and $K_{ij}(r)$ would be correlated. However, for real world heterogeneous patterns the spatial configuration of individuals of species *j* around individuals of species i can widely vary at different locations in the plot. For example, at some locations some individuals of species *i* may have many neighbors of species *j* and at other locations some individuals of species *i* may have only few neighbors of species *j*. To describe the different types of spatial configurations which may arise we classify the bivariate pattern of a species pair at neighborhoods *r* into a two-dimensional space spanned by the two axis

$$\hat{P}(r) = \hat{D}_{ij}(r) - (1 - \exp(-\lambda_2 \pi r^2)$$

$$\hat{M}(r) = \ln(\hat{K}_{ij}(r)) - \ln(\pi r^2)$$

where the hat symbol indicates the observed value of each species pair. The $D_{ij}(r)$ and $K_{ij}(r)$ axes were normalized and transformed in a way that the expectation of the independence (or CSR) null model is located at the origin and that positive or negative departures of $K(r)$ from the null model are weighted in the same way (Wiegand et al. 2012).

This approach classifies the species pairs that departed significantly from the null model into four fundamental types of interspecific association patterns:

- **Mixing:** Species pairs located in the upper-right quadrant show a high degree of spatial association (mixing) because here individuals of species *j* occur more often within neighborhoods of radius *r* around individuals of species *i* [$M(r) > 0$], and individuals of species *i* have more neighbors of species *j* [$P(r) > 0$], than expected under the null model.

- **Segregation:** Conversely, species pairs located in the lower-left quadrant show segregation because there are fewer individuals of species *j* within neighborhoods of radius *r* around individuals of species *i* than expected under the null model [$\hat{P}(r) < 0$ and $\hat{M}(r) < 0$].

- **Partial overlap:** Species pairs located in the upper-left quadrants show partial overlap because individuals of species *j* occur more often within neighborhoods of radius *r* around individuals of species *i* [$M(r) > 0$], but a notable proportion of individuals of species *i* have fewer neighbors of species *j* [$P(r) < 0$] than expected under the null model.

- **Type IV:** For species pairs located in the lower-right quadrant, species *i* individuals are highly clustered and some species *j* individuals occur in these clusters [$\hat{P}(r) > 0$ and $\hat{M}(r) < 0$]. This type rarely occurs.

Instructions for the scheme based on univariate patterns

1. Prepare a data set with a univariate *.dat data file for each species. No specific name conventions are required, however, it is better to code the name with the species acronyms (e.g., ADE1TR.dat, ALSEBL.dat, BEILPE.dat) or numbers (e.g., BCI_C1_sp1.dat, BCI_C1_sp2.dat, BCI_C1_sp3.dat,…). In the example data we have sp_1.dat, sp_2.dat, sp_3.dat.

   Do not include species with less than 50 individuals.

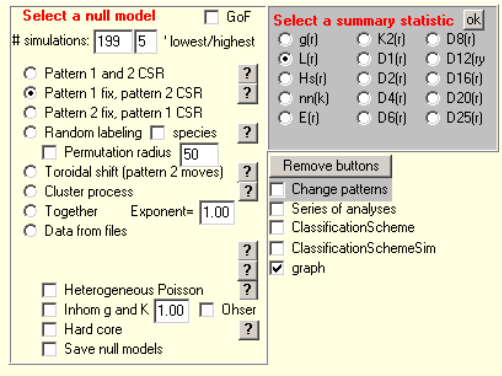
2. Prepare one bivariate data file. In the example this is sp_1_3.dat.

3. Prepare an *.txt ASCII file with a list of the species names (without extension) you want to analyze. In our case the list is "sp.txt":

   ```
   sp_1
   sp_2
   sp_3
   ```

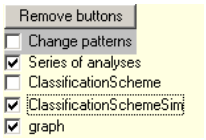
4. Now prepare the example analysis for the bivariate data file. First, execute *Programita*.
5. Highlight data file sp_1_3.dat you want to analyze in Input data file

6. Select "**no grid**" in What do you want to do?
1. Select bin of 1m window Select a new cell size
2. Select a ring width of 3 in the menu "Which method will you use"
3. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
4. Press button "**Calculate Index**"
5. Select L(r) in window "Select a summary statistic" and click the small "**ok**" button
6. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface. Select "**Pattern 1 fix, pattern 2 CSR**" in the window "Select a null model".
7. Specify the **number of simulations of the null model** (199 in the example) and the rule for the estimation of **simulation envelopes** (here the 5th lowest and highest values of the summary statistic of the 199 null model data sets).

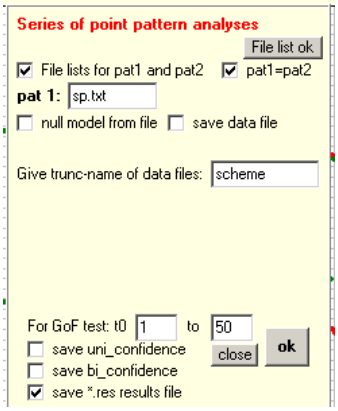


If all settings are specified, press the button "**Calculate Index**" and *Programita* conducts the simulations of the null model. Save the results with the "**Save results**" button.

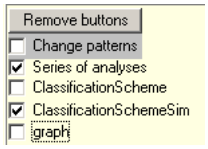
8. Enable check box "**ClassificationSchemeSim**" on the right hand side of the "Select a null model" window. A window opens with settings of the Series of analyses.



9. To select files for pairwise analyses based on the univariate data files select "**File list for pat1 and pat2**".
10. If pattern 1 and 2 should be selected from the same list select "**Pat1=pat2**" (click the check box "**pat1=pat2**" two times). This is necessary to omit that the same file is selected as pattern 1 and pattern 2.
11. To save the bivariate data files that are automatically assembled by *Programita* enable "**save data file**"
12. Insert the name of the file list for **pat 1** (and if appropriate, for pat 2). In our case it is "sp.txt".
13. The trunk-name is used to name the summary output file. We use "scheme"
14. Provide the distance interval for the GoF test (1-50m in our case)



15. Once all settings are specified, click the fat **ok** bottom ok, and then "**Calculate Index**" to start the series of analyses.
16. Disable the checkbox "graph" to simulate quicker.



17. The important output file is then named "Summary_SchemeSim_scheme.txt" ("scheme" is the name you selected). It is a comma delimited ASCII file. It gives you a summary of the result of all analyses and can be used to construct the scheme.

    The first part of the data file:

	A	B	C	D	E	F	G	H	I	J	K
1	Dataname	nr	r0	r1	ln(K12)-ln(K12th) 1	ln(K12)-ln(K12th) 2	ln(K12)-ln(K12th) 3	*	D12-D12th 1	D12-D12th 2	D12-D12th 3
2	rectemp.dat	1	0	50	1.2014438	0.5101046	0.1052974	*	0.0073539	-0.0021827	0.0034796
3	rectemp.dat	2	0	50	-111	-111	-111	*	-0.0014943	-0.0059637	-0.0133684
4	rectemp.dat	3	0	50	1.199928	0.5075179	0.1033702	*	0.002695	-0.0008382	0.0011429
5	rectemp.dat	4	0	50	-111	-111	-111	*	-0.0014943	-0.0059637	-0.0133684
6	rectemp.dat	5	0	50	-111	-111	-111	*	-0.0011805	-0.0047138	-0.0105748
7	rectemp.dat	6	0	50	-111	-111	-111	*	-0.0031993	-0.0127359	-0.0284279

- Dataname: the datafile of the given species pair. It is always rectemp.dat if the bivariate data were composed from the univariate data. The columns "name 1" and "name 2" provided later give the names of the univariate files.
- nr: the number of the species pair analyzed
- r0 and r1: the interval of the GoF test
- the values of the M-axis of the scheme for distances 1, 2, 3 (ln(K12)-ln(K12th) 1, ln(K12)-ln(K12th) 2, ..., ln(K12)-ln(K12th) 50). We show here only distances *r* up to 3. A value of -111 indicates a large negative value (> -4)
- the values of the P-axis of the scheme for distances 1, 2, 3 (-D12-D12th 1, - D12-D12th 2, ..., - D12-D12th 50). We show here only distances *r* up to 3.

The second part of the file:

M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA
rank11	rank12	anzp1	anzp2	name 1	name 2	rankL12 0- 50	rankD12 0- 50	Rank M 1	Rank M 2	Rank M 3	*	Rank P 1	Rank P 2	Rank P 3
1	187	94	255	sp_1	sp_2	187	124	197	109	87	*	163	24	124
1	50	94	119	sp_1	sp_3	50	33	1	173	185	*	1	77	167
1	101	255	94	sp_2	sp_1	101	142	189	110	1	*	153	31	135
1	200	255	119	sp_2	sp_3	200	1	1	189	200	*	26	88	94
1	36	119	94	sp_3	sp_1	36	35	1	177	191	*	1	76	156
1	200	119	255	sp_3	sp_2	200	2	1	200	200	*	62	93	86

- rank11: the rank of the GoF test for the univariate pair correlation function (not used here)
- rank12: the rank of the GoF test for the bivariate pair correlation function (additional info)
- anzp1: number of points of pattern 1
- anzp2: number of points of pattern 2
- name 1: name of focal pattern
- name 2: name of second pattern
- rankL12 0- 50: the rank of the GoF test for the $L_{ij}(r)$ over the entire range of distances selected (to check if the M-axis of scheme is significant)
- rankD12 0- 50: the rank of the GoF test for the $D_{ij}(r)$ over the entire range of distances 1-50 selected (to check if P-axis of the scheme is significant)
- Rank M1, Rank M2, …, Rank M 50: the rank of the M-axis of the scheme at all individual distances r = 1, 2, … 50m analyzed
- Rank P1, Rank P 2, …, Rank P50: the rank of the P-axis of the scheme at all individual distances r = 1, 2, … 50m analyzed

18. Based on the values of the *M* and *P* axes and the rank of the *M* and *P* axes at distance *r* you can determine the association type of a given species pair.

    Because we use two test statistics at the same time [$L_{ij}(r)$, $D_{ij}(r)$] we need to use a P-value of 0.025 for each summary statistics to yield an overall error rate of 5%. If you selected 199 simulation of the null model, the rank must be therefore larger than 195 to be significant.

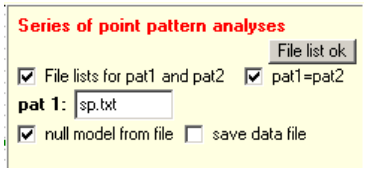
    If (Rank P r) > 195 or (Rank M r) > 195 the species pair belongs at distance *r* to a significant class:

    - P < 0 and M < 0: type 1 (**segregation**)
    - P < 0 and M > 0: type 2 (**partial overlap**)
    - P > 0 and M > 0: type 3 (**mixing**)
    - P > 0 and M < 0: type 4 (does only rarely occur)

    - otherwise the species pair belongs to the no "**significant patterning class"**

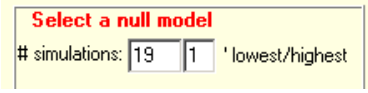
Instructions for the scheme based on univariate component patterns and null models from file

You can do the same procedure as above also for cases where you previously saved the null model files for the different patterns listed in the file lists, for example generated with **pattern reconstruction**.



In this case you need to click "null model from file". The null model patterns corresponding to your data files must follow the name conventions:

data file: `name.dat`
null model file: `rec_name_n.dat`



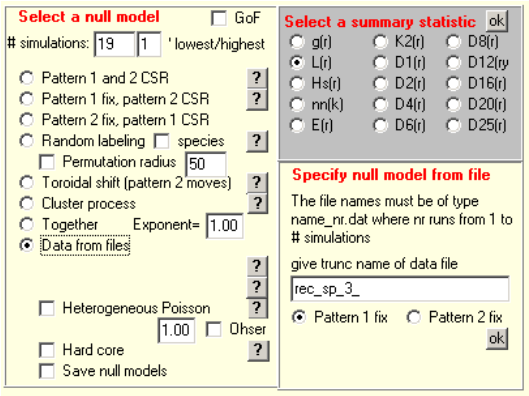
where `name` is the data file (e.g., sp_3 in the example from the file list above) and `n` the number that should run from 1 to the number of # simulations of the null model specified in the window "**Select a null model**".

Derive scheme with pattern reconstruction null model (Example scheme_rec.res)

For each univariate data file you need to generate as much reconstructions as you select for the number of simulations of the null model.

In the first step you have to prepare one example analysis with the correct settings and then save the corresponding *.res file.

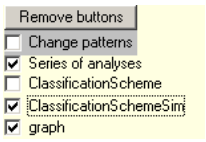
1. First, execute *Programita*.
2. Highlight data file sp_1_3.dat you want to analyze in Input data file
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 3 in the menu "Which method will you use"
6. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
7. Press button "**Calculate Index**"
8. Select L(r) in window "Select a summary statistic"
9. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface. Select "**Data from file**" in the window "Select a null model".
10. Specify the **number of simulations of the null model** (19 in the example) and the rule for the estimation of **simulation envelopes** (here the 1th lowest and highest values of the summary statistic of the 19 null model data sets).
11. Provide trunk name of null model files in window "Specify null model from file". It is "rec_sp_3_" because the data file used sp_1.dat as focal pattern and sp_3.dat as second pattern. Select also "**Pattern 1 fix**" because you randomize pattern 2 and click small "**ok**" button.



If all settings are specified, press the button "**Calculate Index**" and *Programita* conducts the simulations of the null model.

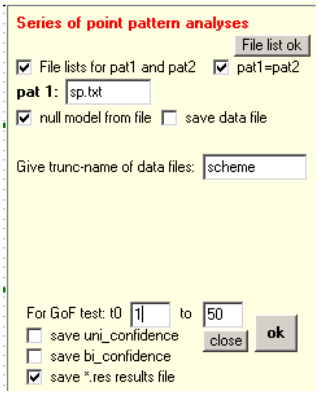
12. Save results file as "scheme_rec.res"

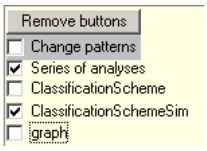
19. Enable check box "**ClassificationSchemeSim**" on the right hand side of the "Select a null model" window. A window opens with settings of the Series of analyses.



20. To select files for pairwise analyses based on the univariate data files select "**File list for pat1 and pat2**".
21. If pattern 1 and 2 should be selected from the same list select "**Pat1=pat2**" (click the check box "**pat1=pat2**" two times). This is necessary to omit that the same file is selected as pattern 1 and pattern 2.
22. To save the bivariate data files enable "**save data file**"
23. Insert the name of the file list for **pat 1** (and if appropriate, for pat 2). In our case it is "sp.txt".

24. The trunk-name is used to name the summary output file. We use here the name "scheme"
25. Provide the distance interval for the GoF test (1-50m in our case)
26. Once all settings are specified, click the fat **ok** bottom ok, and then "**Calculate Index**" to start the series of analyses.
27. Disable the checkbox "graph" to simulate quicker.





4.2.9 Bivariate Thomas process with shared parents

The simple univariate Thomas process can be generalized to a bivariate point process that includes an explicit mechanism of attraction between the two component patterns.

While in a simple Thomas process the points of one pattern are distributed around the cluster centers, the bivariate Thomas process distributes the points of two patterns around the cluster centers. Each component pattern has an own parameter ρ determining the number of cluster centers for this pattern and a parameter σ determining the approximate size of the clusters. See section 4.2.3.1 "Bivariate Thomas Process with Shared Parents" in Wiegand and Moloney (2014).

If $\rho_1 = \rho_2$ all cluster centers are shared and the attraction between pattern 1 and 2 will be maximal. However, if $\rho_1 < \rho_2$ or $\rho_1 > \rho_2$ not all clusters host points of both types. In this case some parents are not shared. The fewer parents are shared, the less attraction exists between pattern 1 and pattern 2. Note that the degree of attraction is also determined by the size of the clusters.

The pair correlation function of the simple Thomas process yields:

$$g(r,\rho_1,\sigma_1) = 1 + \frac{1}{\rho_1} h_2(r,\sigma_1) = + \frac{1}{\rho_1} \frac{\exp(-r^2/4\sigma_1^2)}{4\pi\sigma_1^2}$$

and the pair correlation function of the simple bivariate Thomas process with shared parents (i.e., $\rho_1 = \rho_2 = \rho$) yields

$$g_{12}(r,\rho_1,\sigma_1,\sigma_2) = 1 + \frac{1}{\rho} \frac{\exp(-r^2/[2(\sigma_1^2+\sigma_2^2)]}{2\pi(\sigma_1^2+\sigma_2^2)} = 1 + \frac{1}{\rho} \frac{\exp(-r^2/[4(\sigma_1^2+\sigma_2^2)/2]}{4\pi(\sigma_1^2+\sigma_2^2)/2}$$

Thus, *Programita* can fit basically the pair correlation function of a simple Thomas process to the observed bivariate pair correlation function but with parameters $\sigma_t^2 = (\sigma_1^2+\sigma_2^2)/2$ and ρ:

$$g_{12}(r,\rho,\sigma_t) = 1 + \frac{1}{\rho} \frac{\exp(-r^2/4\sigma_t^2)}{4\pi\sigma_t^2}$$

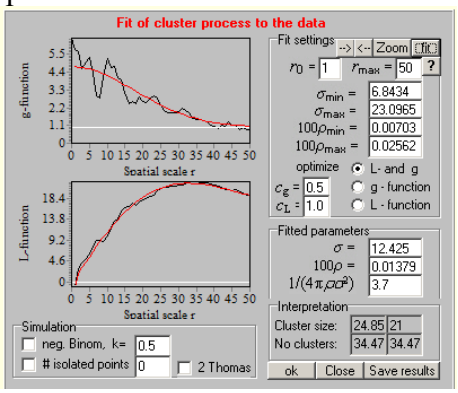
The fitted point process is therefore only appropriate if the fitted value of σ_t yields in good approximation $\sigma_t^2 = (\sigma_1^2+\sigma_2^2)/2$ and if $\rho_1 \approx \rho_2 \approx \rho$. To test this, you need first to determine the parameters ρ_1, ρ_2, σ_1 and σ_2 of the univariate patterns using univariate analysis.

This example is based on the example Book_Fig4_13_bi.res and the two univariate patterns are taken from two different realizations of the univariate parent-offspring Thomas process. That means that the cluster centers are the same (i.e., shared parents) and the cluster sizes are also the same. The parameters of the original point process were $\sigma = 12.54$ and $A\rho = 34.3$.

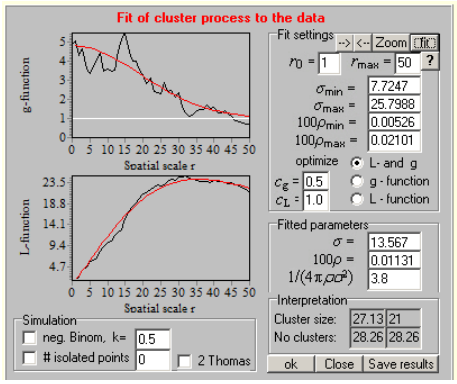
Example BiThomasShared.res

1. In the first step fit a simple Thomas process to the two univariate component patterns to verify that the assumption of this point process (i.e., both patterns follow a simple Thomas process) holds.
2. The fitted parameters of pattern 1 (file BiThomasShared_p1.dat) yield $\sigma_1 = 12.43$ and $A\rho_1 = 34.5$
3. The fitted parameters of pattern 2 (file BiThomasShared_p2.dat) yield $\sigma_2 = 13.57$ and $A\rho_2 = 28.3$

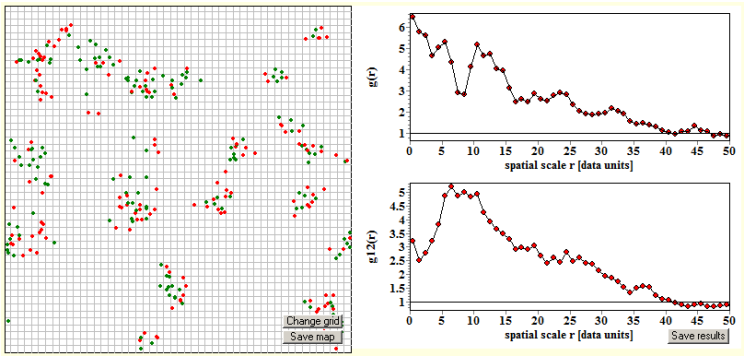
pattern 1:



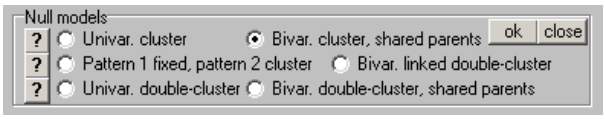
pattern 2:



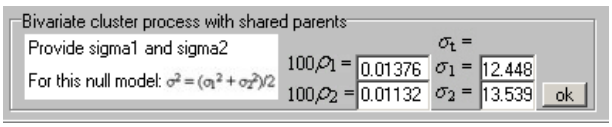
4. Using the information from the univariate analysis, now fit in a second step the bivariate Thomas process with shared parents to the data
5. Execute *Programita*.
6. Highlight data file BiThomasShared.dat you want to analyze in Input data file.
7. Select "**no grid**" in What do you want to do?
8. Select bin of 1m window Select a new cell size
9. Select a ring width of 3 in the menu "Which method will you use"
10. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
11. Press button "**Calculate Index**"
12. *Programita* then shows the bivariate pattern. It is clear that the points of the two patterns are merged within the same clusters:



13. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
14. Select "**Cluster process**" in the window "Select a null model".
15. A window "Fit of cluster process to data" opens. Select in the section "Null models" at the bottom the button "**Bivar. cluster, shared parents**".

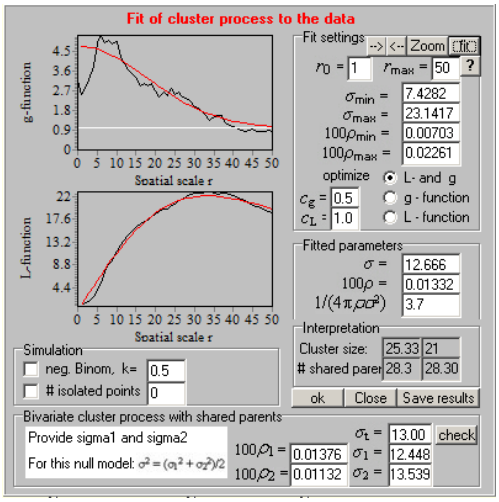


Now this window appears that asks you to provide the parameters from the univariate analysis of patterns 1 and 2. Copy-paste the results from the univariate analyses and press the small "**ok**" buttons:



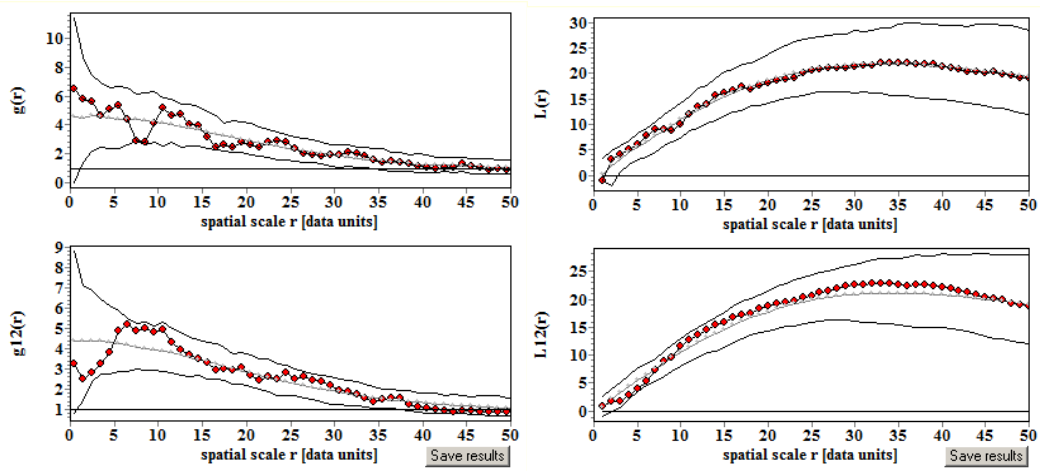
16. Now you are at the fitting window. Use the "**fit**" and "**zoom**" buttons to find a good fit. *Programita* fits the bivariate pair correlation function to the data, estimating

$$g_{12}(r,\rho,\sigma_t) = 1 + \frac{1}{\rho}\frac{\exp(-r^2/4\sigma_t^2)}{4\pi\sigma_t^2} \text{ where } \sigma_t^2 = (\sigma_1^2+\sigma_2^2)/2\,.$$

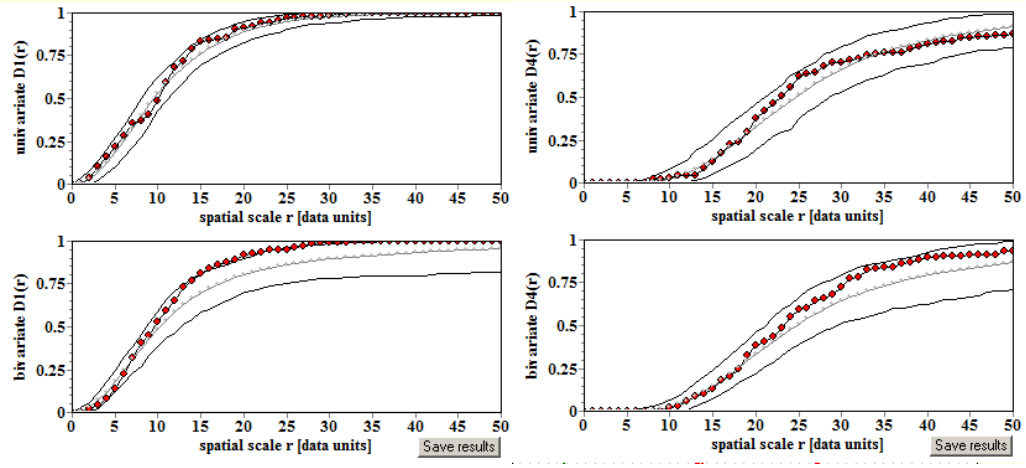


The best fit yields a parameter $A\rho = 28.3$ shared parents and the fitted value of $\sigma_t = 12.66$ is in good agreement with $\sigma_t = ((\sigma_1^2+\sigma_2^2)/2)^{0.5} = 13.0$.

17. As expected, the different summary statistics are in good agreement with the fitted point process for both, the univariate analysis of pattern 1 and the bivariate analysis:



The small (non-significant) departure in the nearest neighbor distribution function is caused by the underestimation of the number of clusters in pattern 2. A few cluster centers were not shared and therefore for the non-shared clusters of pattern 1 the nearest type 2 neighbors were farther away than expected from the data:



4.2.10 Bivariate Thomas process with partly shared parents

Not all clusters of the bivariate Thomas process with shared clusters need to be shared. If fewer parents are shared, the attraction of the two patterns will decline and in the extreme case where no cluster is shared they will be independent. Section 4.2.3.2 "Bivariate Thomas Process with Partly Shared Parents" in Wiegand and Moloney (2014) estimate the pair correlation function for the more general cluster process where not all clusters are shared. The pair correlation function of the simple bivariate Thomas process with partly shared parents yields

$$g_{12}(r,\rho^*,\sigma_1,\sigma_2) = 1 + \frac{1}{\rho^*} h_2(r,\sigma^*) \text{ where } \sigma^* = \sqrt{(\sigma_1^2 + \sigma_2^2)/2} \text{ and } \rho^* = \frac{\rho_1 \rho_2}{\rho_s}$$

The ρ_1 and ρ_2 are the fitted parameter of the univariate analyses of pattern 1 and 2, respectively, and the number of shared clusters yields $A\rho_s$. Thus, this point process has an "effective" number of clusters ρ^* which agrees with the number of clusters if all clusters are shared (i.e., $\rho_1 = \rho_2 = \rho_s$), and which becomes very large if no clusters are shared (i.e., ρ_s is small).

Thus, as before *Programita* fits basically the pair correlation function of a simple Thomas process to the observed bivariate pair correlation function and estimates ρ^* and σ^*.

Comparing the values of ρ^* and σ^* with the values of the parameters from the univariate analysis σ_1, $A\rho_1$, σ_2, and $A\rho_2$ allows to find out if the point process yields consistent parameters. First, the fitted value for σ^* should yield in approximation $\sigma^* = ((\sigma_1^2 + \sigma_2^2)/2)^{0.5}$. Second, we also expect $\rho_1 > \rho_s$ and $\rho_2 > \rho_s$, because the number of shared parents cannot be greater than the number of parents of pattern 1 or 2.

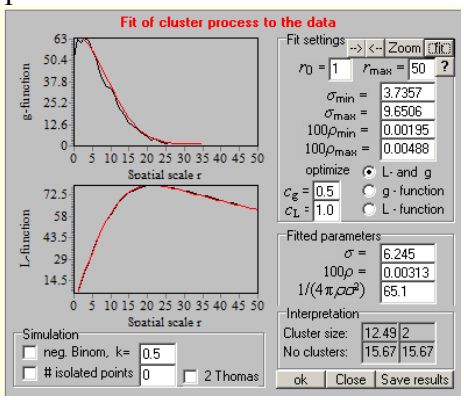
This example is based on a realization of the bivariate Thomas process with partly shared parents fitted in example of Figure 4.29 in Wiegand and Moloney (2014).

The parameters of the original point process were $\sigma_1 = 6.65$, and $A\rho_1 = 14.85$, $\sigma_2 = 4.42$, $A\rho_2 = 64.75$, and the fitted parameter of the shared parents was $A\rho = 10.3$.

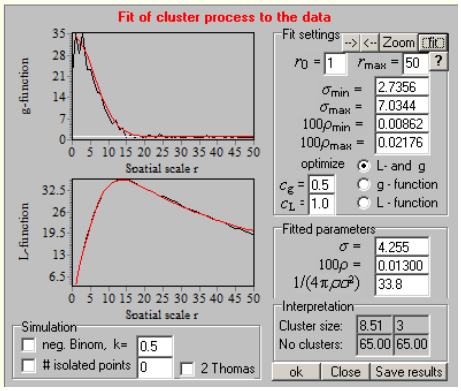
Example BiThomasShared.res

1. In the first step fit a simple Thomas process to the two univariate component patterns to verify that the assumption of this point process (i.e., both patterns follow a simple Thomas process) holds.
2. The fitted parameters of pattern 1 (file Book_Fig4_29_p1.dat) yield $\sigma_1 = 6.25$ and $A\rho_1 = 15.67$
3. The fitted parameters of pattern 2 (file Book_Fig4_29_p2.dat) yield $\sigma_2 = 4.26$ and $A\rho_2 = 65.0$

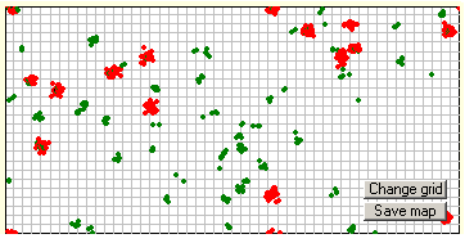
pattern 1:

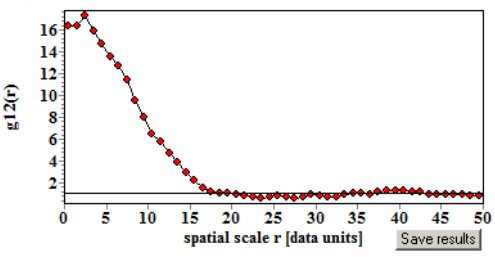


pattern 2:

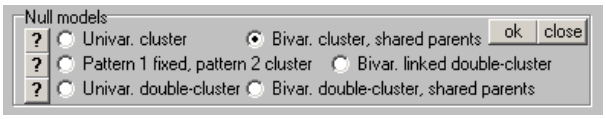


4. Using the information from the univariate analysis, now fit in a second step the bivariate Thomas process with shared parents to the data
5. Execute *Programita*.
6. Highlight data file Book_Fig4_29.dat you want to analyze in Input data file.
7. Select "**no grid**" in What do you want to do?
8. Select bin of 1m window Select a new cell size
9. Select a ring width of 3 in the menu "Which method will you use"
10. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
11. Press button "**Calculate Index**"
12. *Programita* then shows the bivariate pattern. It is clear that not all clusters are shared but that there is a strong attraction between the two patterns:

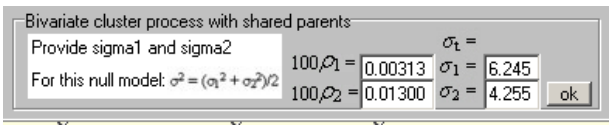




13. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
14. Select "**Cluster process**" in the window "Select a null model".
15. A window "Fit of cluster process to data" opens. Select in the section "Null models" at the bottom the button "**Bivar. cluster, shared parents**".

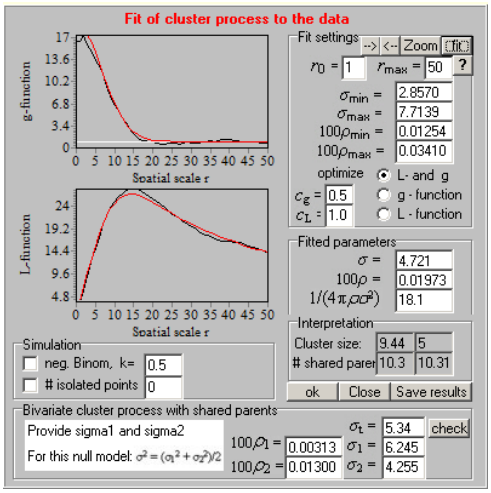


Now this window appears that asks you to provide the parameters from the univariate analyses of patterns 1 and 2. Copy-paste the results from the univariate analyses and press the small "**ok**" buttons:



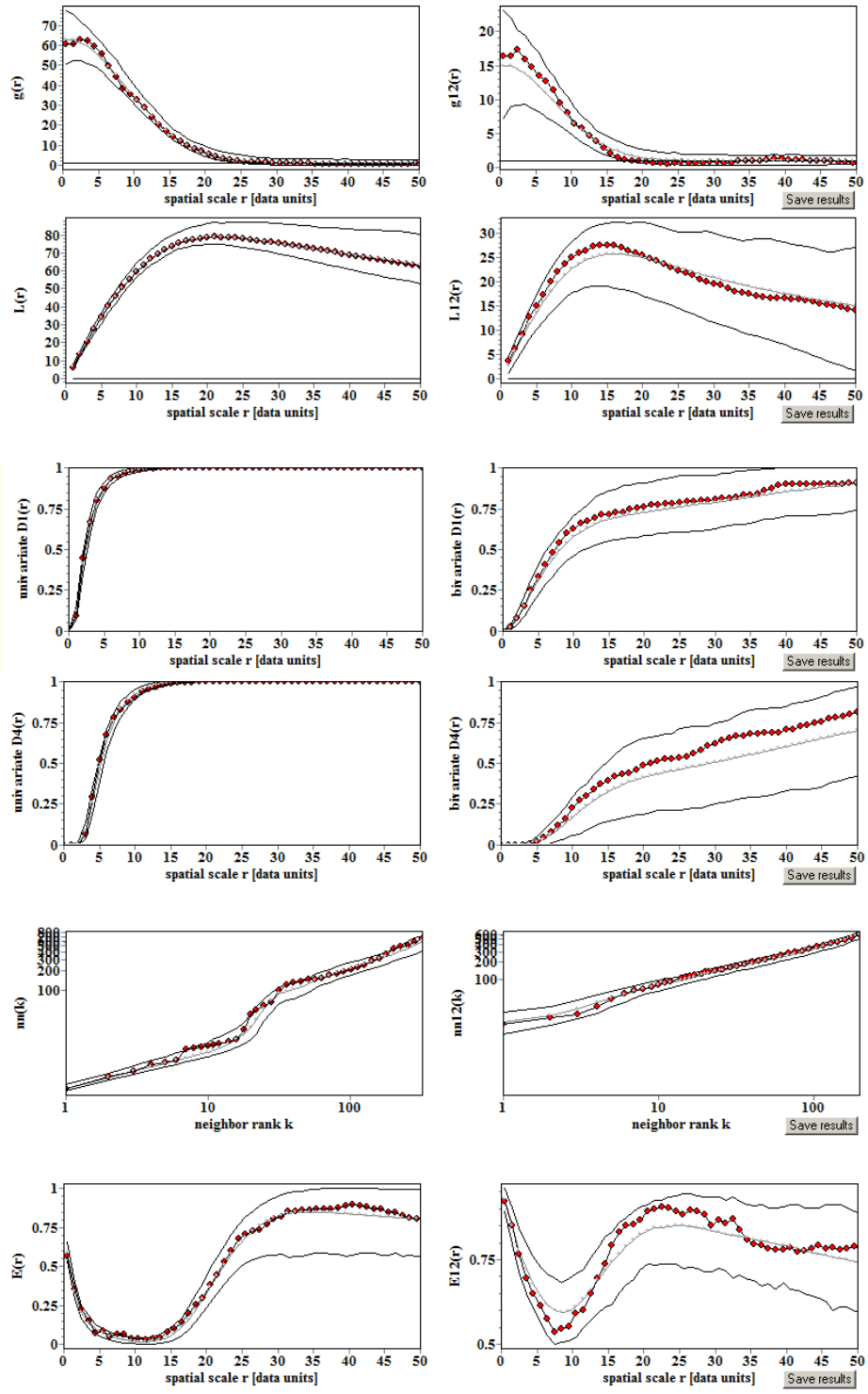
16. Now you are at the fitting window. Use the "fit" and "zoom" buttons to find a good fit. *Programita* fits the bivariate pair correlation function to the data, estimating

$$g_{12}(r,\rho,\sigma_t) = 1 + \frac{1}{\rho}\frac{\exp(-r^2/4\sigma_t^2)}{4\pi\sigma_t^2} \quad \text{where} \quad \sigma_t^2 = (\sigma_1^2+\sigma_2^2)/2\,.$$



The best fit yields a parameter $A\rho = 10.3$ shared parents and the fitted value of $\sigma_t = 4.7$ is in good agreement with $\sigma_t = ((\sigma_1^2+\sigma_2^2)/2)^{0.5} = 5.34$.

17. As expected, the different summary statistics are in good agreement with the fitted point process for both, the univariate analysis of pattern 1 and the bivariate analysis. However, note the relatively wide simulation envelopes of the bivariate analysis which indicates a strong stochastic variability among the realizations of this point process.


g(r)
g12(r)
L(r)
L12(r)
univariate D1(r)
bivariate D1(r)
univariate D4(r)
bivariate D4(r)
nn(k)
nn12(k)
E(r)
E12(r)
spatial scale r [data units]
neighbor rank k
Save results

4.2.11 Bivariate hard and soft core processes

Gibbs (or Markov) point processes can be used to consider interaction among points of one type and between points of different type (see section 4.2.3.4 "Gibbs Processes to Model Interactions between Species" in Wiegand and Moloney 2014). They are natural generalizations of the univariate Gibbs processes.

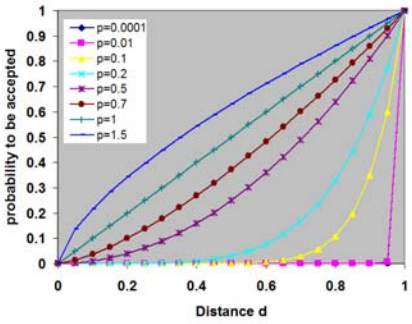
Simulation of bi (or multivariate) Gibbs processes requires optimization techniques where points of an initial pattern are deleted and replaced by randomly drawn points, which are accepted if the new point configuration becomes more likely, given the location density function. Such "birth and death" simulation algorithms closely resemble aspects of spatial population or community dynamics and are structurally very similar to spatially explicit, individual-based simulation models (Grimm and Railsback 2005) which are used by ecologists to study the spatiotemporal dynamics of plant populations and communities.

Programita has not implemented Gibbs processes. However, to provide you the possibility to simulate simple point patterns with repulsion or segregation, *Programita* includes a simple algorithm based on "random sequential absorption" (RSA) processes to produce bivariate hard and soft core patterns.

The RSA algorithm implemented in *Programita* is simple. It is constructed by placing iteratively and randomly points within an observation window W which are thought to be the centers of disks with radius r_0. For bivariate patterns *Programita* first places the points of pattern 1 and then in a second step, the points of pattern 2. Thus, points of pattern 2 do not influence the placement of points of pattern 1.

Five parameters govern this point process, two radiuses r_1 and r_2 of pattern 1 and 2, respectively, that give the "zone of influence" of the two patterns, and three interaction coefficients p_1, p_2, and p_{21} that determine together with the distance d to the nearest already placed point the probability that the new point is accepted. As in the univariate case, this probability yields

$$p_{\mathrm{HC}}(d) = \begin{cases} d^{1/p} & \text{for } d \le r_{12} \\ 1 & \text{for } d > r_{12} \end{cases}$$



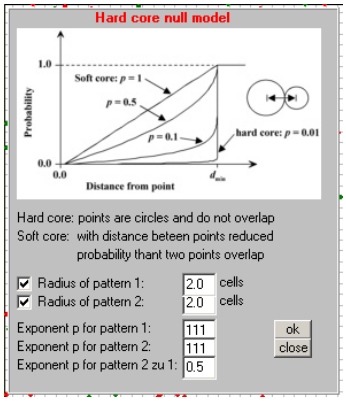
where p is p_1, p_2, and p_{12} depending on the types of the tentatively placed point and the nearest neighbor, $r_{12} = 2\ r_1$ if the tentatively placed point and the nearest neighbor are of type 1, $r_{12} = 2\ r_2$ if the tentatively placed point and the nearest neighbor are of type 2, and $r_{12} = r_1 + r_2$ if the tentatively placed point and the nearest neighbor are of different type.

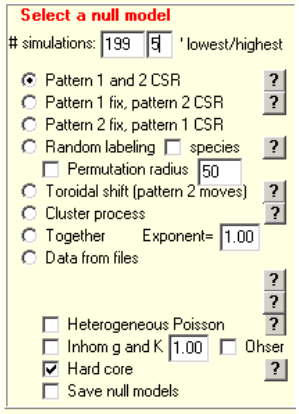
This setting allows for different types of patterns with interactions only between points of type 1 and type 2, only between points of type 1, between points of type 2, and so on.

Example Book_Fig4_21e_bi.res (bivariate RSA inhibition process)

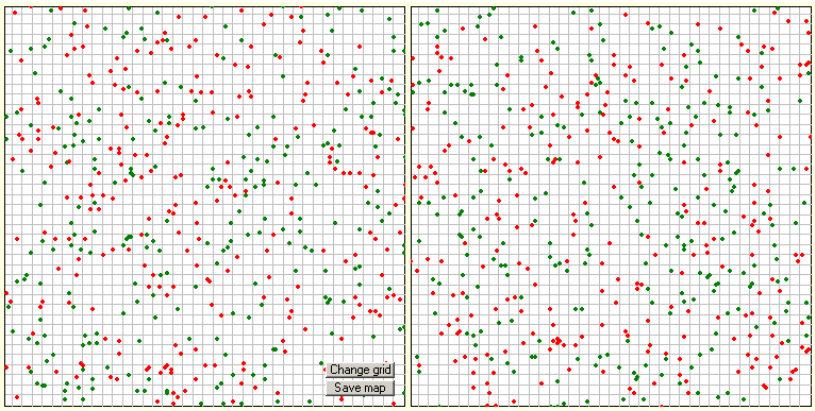
This pattern has been generated with a RSA algorithm to simulate non-overlapping disks with radius r_0 = 2m, but overlap was only restricted between disks of different type.

1. Execute *Programita*.
2. Highlight data file Book_Fig4_21e.dat you want to analyze in Input data file.
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 3 in the menu "Which method will you use"
6. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
7. Press button "**Calculate Index**"
8. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
9. Select null model "Pattern 1 and 2 CSR" to start with the basic CSR algorithm.
10. Click checkbox "Hard core" and go to the window "Hard core null model" to define details of the RSA null model. Click "**Radius of pattern 1**" and "**Radius of pattern 2**"because you have a bivariate pattern and provide the radius (2.0) in our case.

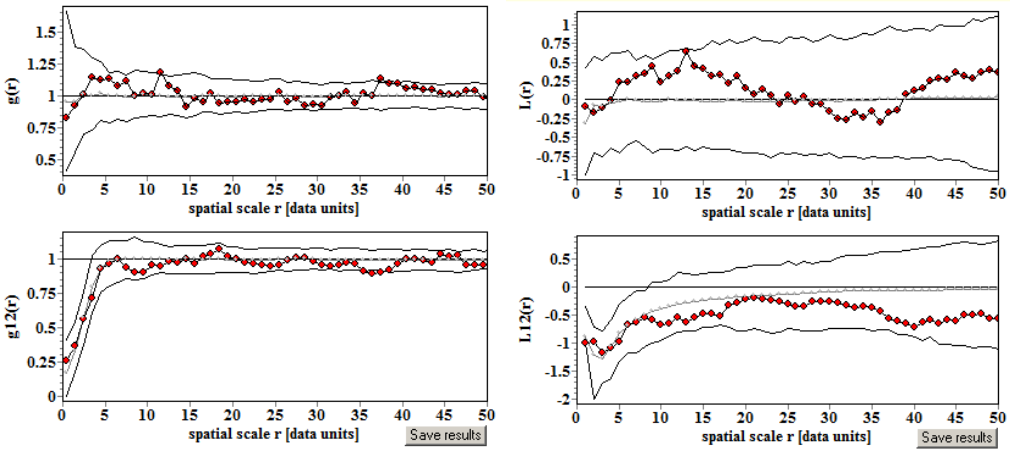




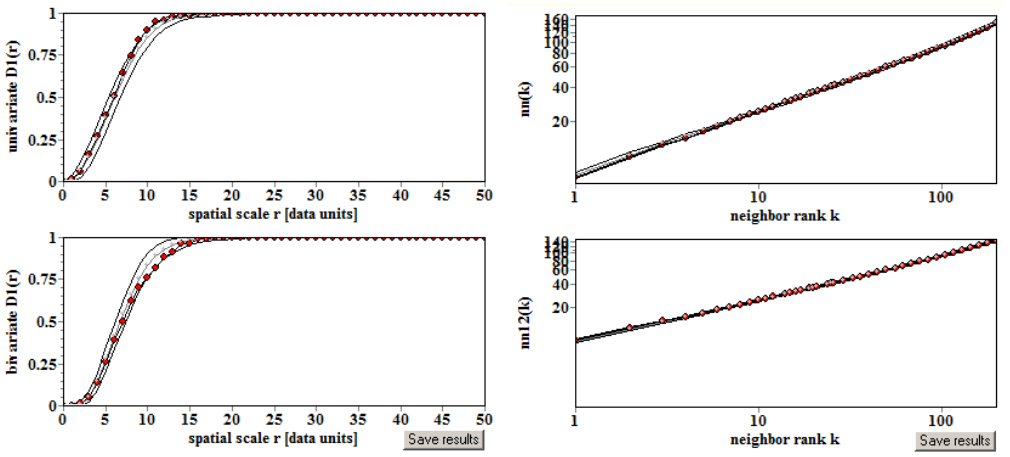
11. Because the two univariate component patterns should be CSR patterns, select large values of the exponents p_1 and p_2 e.g., 111. However, because of a negative interaction between type 1 and type 2 points select p_{21} = 0.5. Finally, click the small "**ok**" button.
12. To simulate the point process press "**Calculate Index**". As expected, the simulated patterns look very similar to the observed pattern:



13. The uni- and bivariate pair correlation functions and the *L*-functions agree well with the simulated point process and the bivariate summary statistics show the typical "soft-core" shape:



14. The same is true for the nearest neighbor distribution functions:

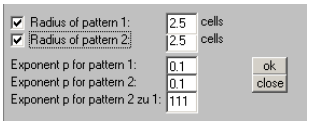


Example Book_Fig4_21a_bi.res (Bivariate inhibition process)

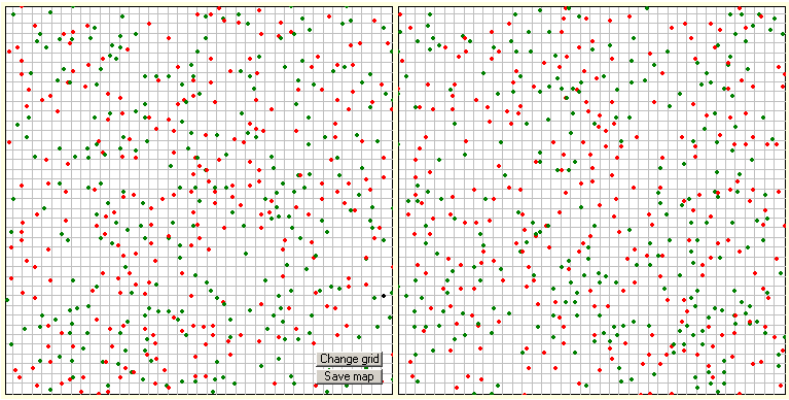
This pattern has been generated with a RSA algorithm to simulate non-overlapping disks with radius r_0 = 2m where "interactions" occurred only among points of the same type.

1. Execute *Programita*.
2. Highlight data file Book_Fig4_21a.dat you want to analyze in Input data file.
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 3 in the menu "Which method will you use"
6. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
7. Press button "**Calculate Index**"

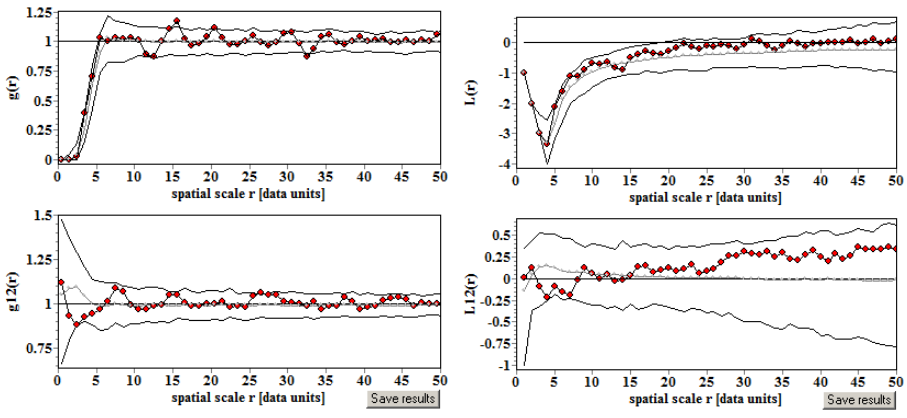
8. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
9. Select null model "Pattern 1 and 2 CSR" to start with the basic CSR algorithm.
10. Click checkbox "Hard core" and go to the window "Hard core null model" to define details of the RSA null model. Click "**Radius of pattern 1**" and "**Radius of pattern 2**"because you have a bivariate pattern and provide the radius (2.5) in our case.
11. Because the two univariate component patterns should be hard core patterns, select small values of the exponents p_1 and p_2 e.g., 0.1. However, because there should be no interaction between type 1 and type 2 points select p_{21} = 111. Finally, click the small "**ok**" button.



12. To simulate the point process press "**Calculate Index**". As expected, the simulated patterns look very similar to the observed pattern:



13. The uni- and bivariate pair correlation functions and the *L*-functions agree well with the simulated point process, and the univariate summary statistics show the typical "hard-core" shape:



The distribution functions of the distances to the nearest neighbor are also very well matched.

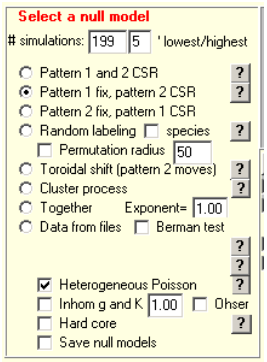
4.2.12 Heterogeneous Poisson processes for bivariate patterns

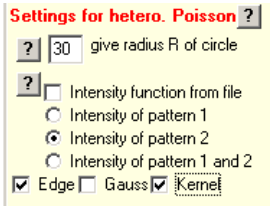
The heterogeneous Poisson process can be used to account for first-order heterogeneity [i.e., the pattern has a non-constant intensity function $\lambda(\mathbf{x})$] in one of the two component patterns of a bivariate pattern. The summary statistics are still impacted by the heterogeneity but the null model shows the same heterogeneity than the original pattern. The shape of the observed summary statistics, relative to that of the simulated null model patterns, allows revealing potential effects of species interactions.

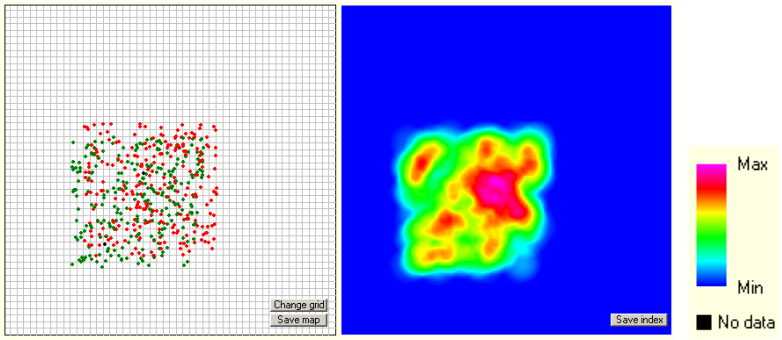
The following examples show simple cases of heterogeneity which are nevertheless often close to real-world patterns of tropical forests. In the examples the two component patterns are CSR and independent of each other, but only distributed within subareas A_1 and A_2 of the observation window W. Depending on the size and the overlap of the two subareas, the different association types mixing, partial overlap and segregation already introduced above in the classification scheme can occur.

Example Book_Fig4_31a.res
The following example presents the analysis of Figure 4.31 using a heterogeneous Poisson process with non-parametric kernel estimate for pattern 2 whereas pattern 1 remains unchanged. You can repeat the analyses with the data files Book_Fig4_31b.dat and Book_Fig4_31c.dat

1. Execute *Programita*.
2. Highlight data file Book_Fig4_31a.dat you want to analyze in Input data file
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 3 in the menu "Which method will you use"
6. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
7. Press button "**Calculate Index**"
8. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
9. Select "**Pattern 1 fix, pattern 2 CSR**" in the window "Select a null model".
10. Specify the number of simulations of the null model (199 in the example) and the rule for the estimation of simulation envelopes (here the 5th lowest and highest values of the summary statistic of the 199 simulated null model data sets).
11. Click checkbox "**Heterogeneous Poisson**"
12. Go to window "Settings for hetero. Poisson" on the left and insert the bandwidth R (30m in the example), enable "**Kernel**" for the Epanechnikov kernel and select "**Intensity of pattern 2**" (because this null model randomizes pattern 2). Edge correction "**Edge**" is enabled by default.
13. Click "**Calculate Index**" and *Programita* estimates the intensity function and shows the pattern and the corresponding intensity function.

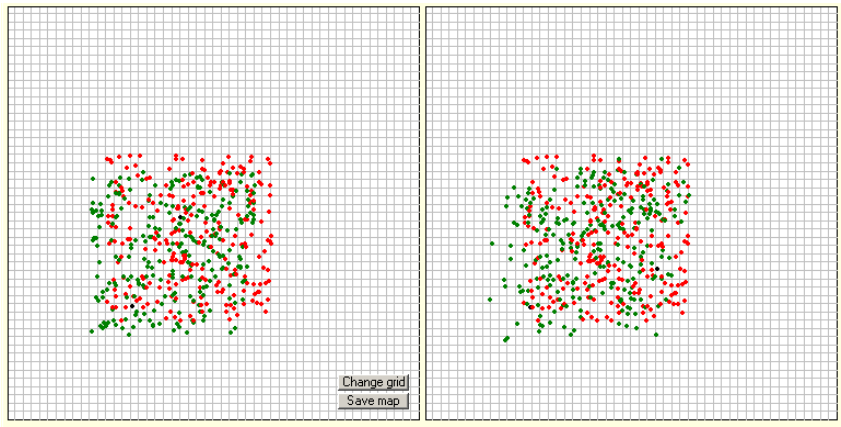






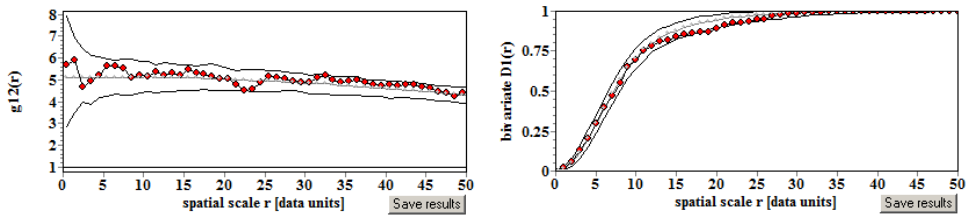
Click OK at the message box to save the intensity file. The file is saved with name int_E_Book_ Book_Fig4_31a_R2_30.int where the "int_E" indicates Epanechnikov kernel, Book_Fig4_31a.dat was the data file, "_R2_30" means that the intensity was estimated with pattern 2 and bandwidth 30.

14. Now *Programita* conducts the analysis. You can observe during the simulations that the null model distributes the points with probability proportionally to the intensity. Here an example:



The strong heterogeneity of the second pattern (green points) is conserved, although pattern 2 is somewhat "smeared" because of the kernel function.

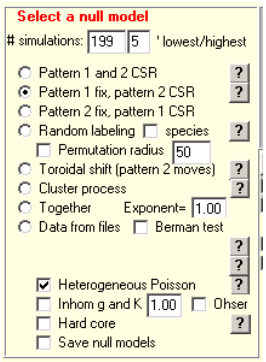
15. The result resembles that in Figure 4.31d, g well:

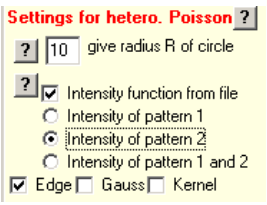


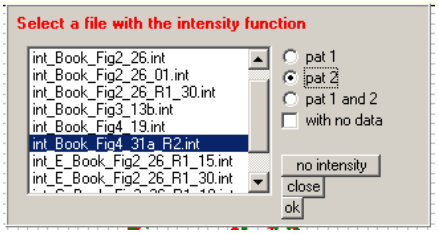
Example Book_Fig4_31a_int2.res

The following example presents the analysis above, but uses instead of the approximation of the intensity function based on a kernel estimate the "real" intensity function which has a value of λ inside the (100, 300) × (100, 300) subarea occupied by pattern 2. The null model again is a heterogeneous Poisson process for pattern 2 whereas pattern 1 remains unchanged.

1. Execute *Programita*.
2. Highlight data file Book_Fig4_31a.dat you want to analyze in Input data file
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 3 in the menu "Which method will you use"
6. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
7. Press button "**Calculate Index**"
8. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
9. Select "**Pattern 1 fix, pattern 2 CSR**" in the window "Select a null model".
10. Specify the number of simulations of the null model (199 in the example) and the rule for the estimation of simulation envelopes (here the 5th lowest and highest values of the summary statistic of the 199 simulated null model data sets).
11. Click checkbox "**Heterogeneous Poisson**"
12. Go to window "Settings for hetero. Poisson" on the left and select "**Intensity of pattern 2**" (because this null model randomizes pattern 2).
13. Click checkbox "**Intensity function from file**". The window "Select a file with the intensity function" opens.
14. Highlight in this window the file int_Book_Fig4_31a_R2.int that contains the intensity function,

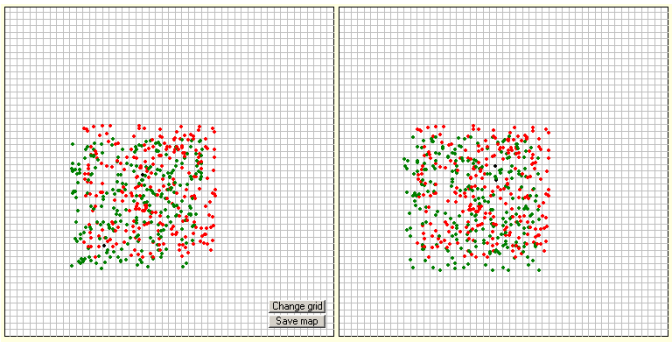




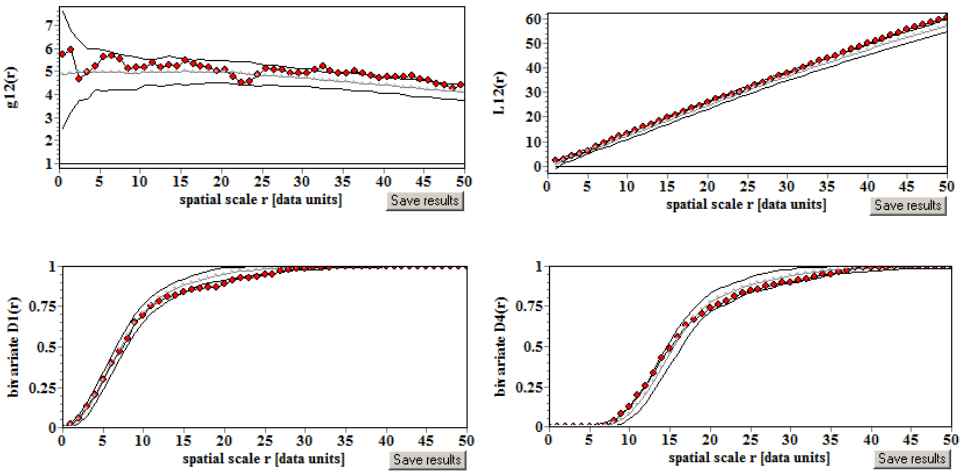


select also "pat 2" because it is the intensity of pattern2 and then the small "**ok**" button. Be sure that "**Intensity of pattern 2**" is selected in the window "Settings for hetero. Poisson".

15. Click "**Calculate Index**" and *Programita* simulates the heterogeneous Poisson process. Because now the real intensity function was used, the null model patterns are not smeared as in the heterogeneous Poisson process with a kernel estimate of the intensity function:



16. The results are similar to the previous case, as expected, there is no significant departure from the null model although a slight departure is visible at larger scales in the *g*- and *L*-function:



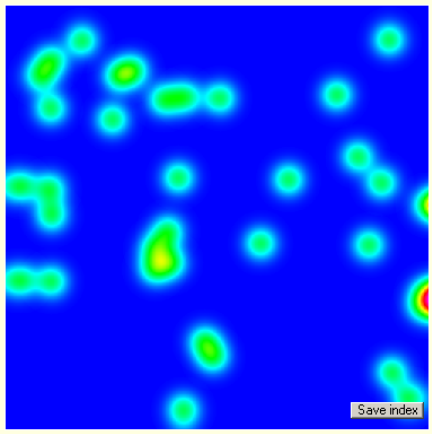
Heterogeneous Poisson process with dispersal kernel (Example Book_Fig4_18.res)

The example Book_Fig4_18.res was used in the univariate section to illustrate the duality between a Cox process and a parent-offspring Thomas process where the cluster centers (i.e., parents) are known and are the same in each simulation of the point process. In this case we assume a "dispersal kernel" around the points of type 1 (which are the cluster centers) and the points of type 2 are distributed in accordance with this dispersal kernel.

The procedure for this null model is the same as in the example above, but now you select in the window "Settings for hetero. Poisson "**Intensity of pattern 1**" (because this null model uses the intensity function of pattern 1 to randomize pattern 2).

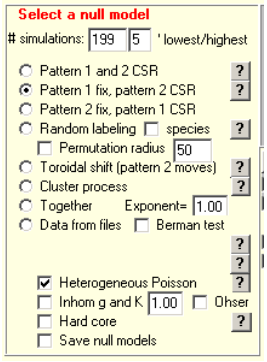
1. Execute *Programita*.
2. Highlight data file Book_Fig4_18.dat you want to analyze in Input data file
3. Select "**no grid**" in What do you want to do?

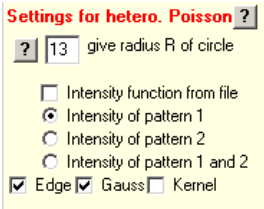
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 3 in the menu "Which method will you use"
6. Accept selection of neighborhood ranks for estimation of $D^k(r)$.
7. Press button "**Calculate Index**"
8. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
9. Select "**Pattern 1 fix, pattern 2 CSR**" in the window "Select a null model".
10. Specify the number of simulations of the null model (199 in the example) and the rule for the estimation of simulation envelopes (here the 5th lowest and highest values of the summary statistic of the 199 simulated null model data sets).
11. Click checkbox "**Heterogeneous Poisson**"
12. Go to window "Settings for hetero. Poisson" on the left and insert the bandwidth R (13m in the example), enable "**Gauss**" for the kernel function being 2-dimensional normal distribution and select "**Intensity of pattern 1**" (because in this null model the intensity is based on pattern 1). Edge correction "**Edge**" is enabled by default.
13. Click "**Calculate Index**" and *Programita* estimates the intensity function and shows the pattern and the corresponding intensity function:

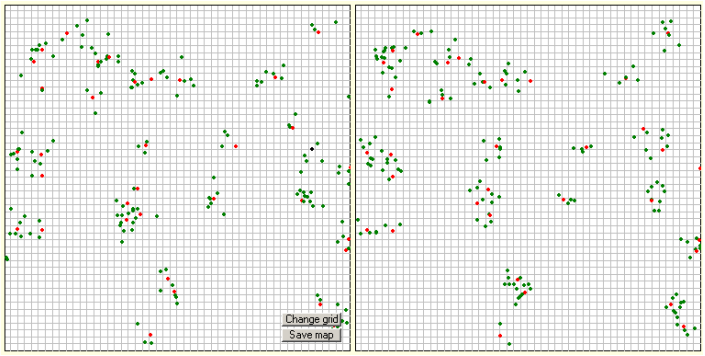


14. Click OK at the message box if you want to save the intensity file. The file is saved with int_G_Book_Fig4_18_R1_13.int where the "int_G" indicates Gaussian kernel, Book_Fig4_18.dat was the data file, "_R1_13" means that the intensity was estimated with pattern 1 and bandwidth 13.

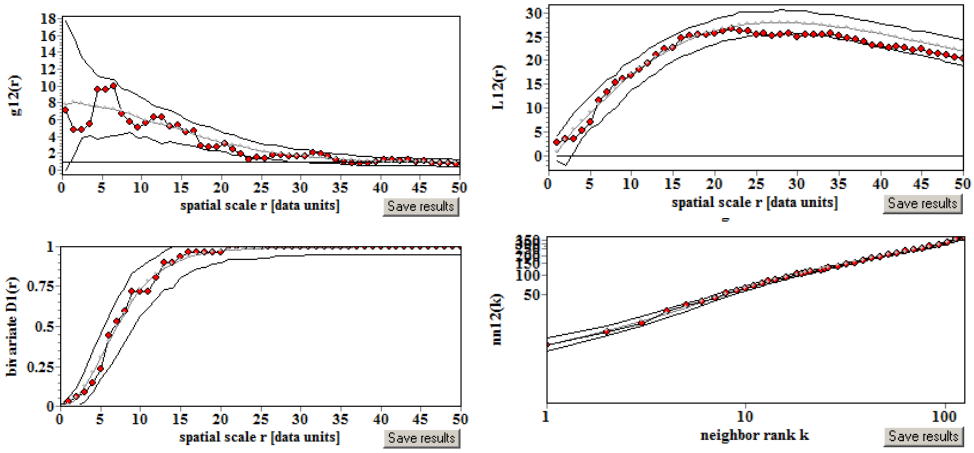
15. Now *Programita* conducts the analysis. You can observe during the simulations that the null model distributes the points with probability proportionally to the intensity function. Here an example:







16. As expected, the summary statistics of the point process agree well with that of the observed data and resemble that of Figure 4.18:



4.2.13 Inhomogeneous *g*- and *K* functions

Inhomogeneous second-order statistics are not yet implemented for bivariate patterns.

5 Analysis of qualitatively marked patterns

Qualitatively marked patterns usually comprise the coordinates of a univariate pattern, but each point carries an additional (*a posteriori*) qualitative mark that distinguishes two types of points. The qualitative mark is a binary property of the point such as surviving vs. dead, infected vs. non-infected, occupied vs. non-occupied, etc. which was created by a given process *a posteriori* on the existing points of the univariate pattern. This distinguishes (*a posteriori*) qualitative marked patterns from bivariate patterns where the difference between the two types of points is *a priori* (e.g., two different species).

The major interest in analysis of qualitatively marked patterns is in revealing the spatial correlation structure of the marking process, conditional on the given univariate pattern. For example, are dead saplings clustered within all saplings, are dead saplings surrounded by a higher density of saplings than surviving saplings, etc.? The basic null model for this data type is the so-called **random labeling null model** which randomly shuffles the marks over the points, thus removing all potential spatial structure in the marks. Chapter 4.4 in Wiegand and Moloney (2014) provides examples for the different analyses of qualitatively marked patterns that are useful in ecology.

5.1.1 Data preparation

Qualitatively marked patterns comprise the coordinates of the underlying univariate point patterns and the mark. The data files for standard analysis must be an ASCII file with the *.dat extension and the following format (the example shows the first lines of the file Book_Fig_2_15.dat):

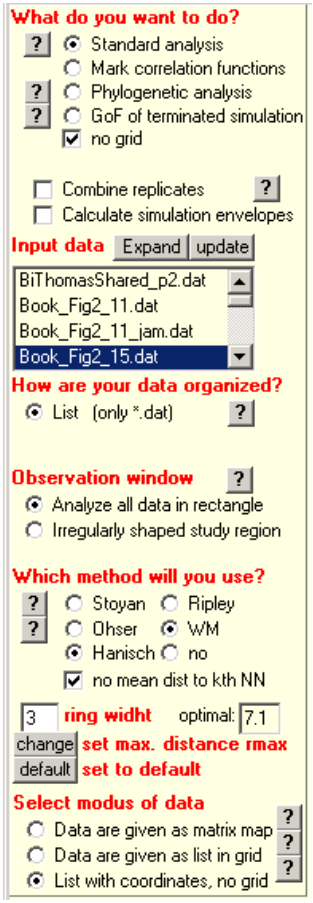
```
0   500   0   500   600
0.60    35.35   0   1
0.70   274.90   0   1
1.10   385.85   1   0
1.15   342.20   1   0
1.80   274.60   1   0
2.02   385.30   0   1
2.50   230.25   1   0
2.60   383.05   1   0
2.85    40.15   0   1
3.25   322.25   0   1
3.65    37.45   0   1
…
```

where the first line gives the size of the observation window (500 × 500 units in the example) and the number of points in the underlying univariate pattern (= number of lines following the header). The first two columns are the coordinates, an entry "1" in the third column indicates that the point is of type 1 (i.e., dead in the example) and an entry "1" in the fourth column indicates that the point is of type 2 (i.e., surviving in the example). The value of the third and the forth columns must be for the standard analysis mode "0 1" or "1 0", no larger numbers or "1 1" are allowed.

The data file must be a space or tab delimited ASCII file with the *.dat extension. If you use Excel, there is a simple, but obviously generally unknown, way of saving files of a given type with a given extension:

1. Prepare the data file in Excel following the instructions above.
2. Then save as a tab delimited text file, but write "name.dat" for the name (usually you would only write name and end up with a file named name.txt. The quotation marks are important because they force Excel to save the comma delimited file under the name name.dat.

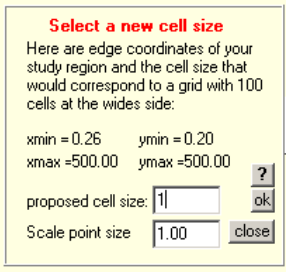
5.1.2 Steps of random labeling analysis in standard mode



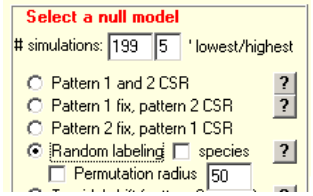
Programita estimates for data files of this type several adapted test statistics based on pair correlation functions (or *L*- and *K*-functions) presented initially in Jacquemyn et al. (2010) and detailed in section 4.4.1 in Wiegand and Moloney (2014). The standard analysis mode can be accessed with the following sequence of actions:

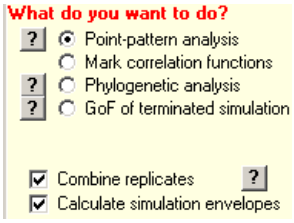
1. Select "**Standard analysis**" in window What do you want to do?
2. Highlight a data file in Input data file ("Book_2_15.dat" in the example).
3. Select "**no grid**" in What do you want to do?

   The window Select a new cell size opens and allows you to provide a bin for your analysis given in units of your data. For example, if your data are in meter units and your observation window is 500 × 500m in size, an appropriate bin would be 1m. Press "**ok**" to confirm selection of the bin.



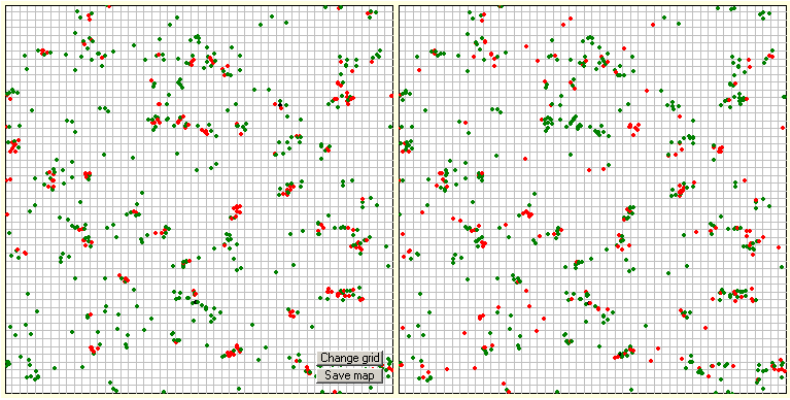
4. Select a ring width of 3 in the menu "Which method will you use"
5. Accept selection of neighborhood ranks for estimation of $D^k(r)$ (will not be used in random labeling analyses).
6. To access the standard analysis mode for **qualitatively marked patterns** (i.e., random labeling analysis; **data type 4**) you must enable the checkbox "Calculate simulation envelopes" in the window What do you want to do? and select the random labeling null model in the Select a null model window.





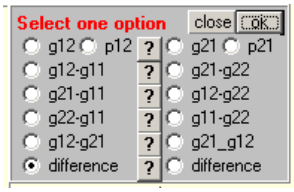
7. If you enable the check box "**Combine replicates**" before running the analysis and save the results of the analysis with Save results under name.res, *Programita* saves additionally two files (name_1.rep and name_2.rep) which allow you to view and save the results for all different test statistics. You can access the procedure for loading the results with button "**Replicates**".

8. Press button "**Calculate Index**" and *Programita* runs the simulations of the random labeling null model:

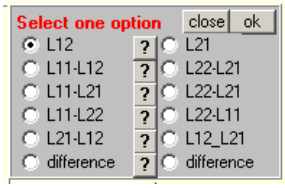


As you can see, the randomization procedure keeps the locations of the points unchanged, but randomly shuffles the mark, indicated here by red (dead) and green (surviving).

9. After running the Monte Carlo simulations of the random labeling null model you can select among several test statistics that allow you to assess potential departures from random labeling. The test statistics are described in detail in Jacquemyn et al. (2010) and section 4.4 on Wiegand and Moloney (2014). Here the ones on the left (the ones on the right results from exchanging labels 1 and 2):



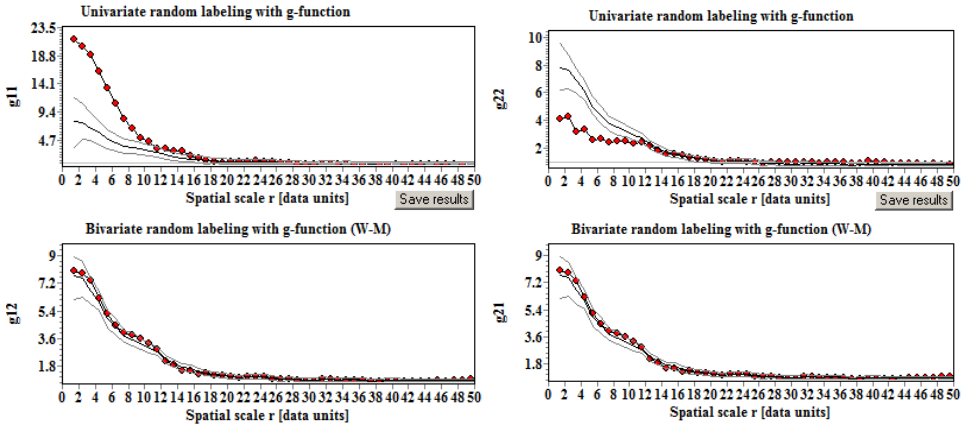
- $g_{12}(r)$: shows the pair correlation functions $g_{11}(r)$ and $g_{12}(r)$
- $p_{12}(r)$: shows the mark connection functions $p_{11}(r)$ and $p_{12}(r)$
- $g_{12}(r)$ - $g_{11}(r)$ shows the $g_{11}(r)$ and $g_{12}(r)$ - $g_{11}(r)$
- $g_{21}(r)$ - $g_{11}(r)$ shows the $g_{11}(r)$ and $g_{21}(r)$ - $g_{11}(r)$
- $g_{22}(r)$ - $g_{11}(r)$ shows the $g_{11}(r)$ and $g_{22}(r)$ - $g_{11}(r)$
- $g_{12}(r)$ - $g_{21}(r)$ shows the $g_{11}(r)$ and $g_{12}(r)$ - $g_{21}(r)$
- difference shows the $g_{11}(r)$ and $g_{1,1+2}(r)$ - $g_{2,1+2}(r)$



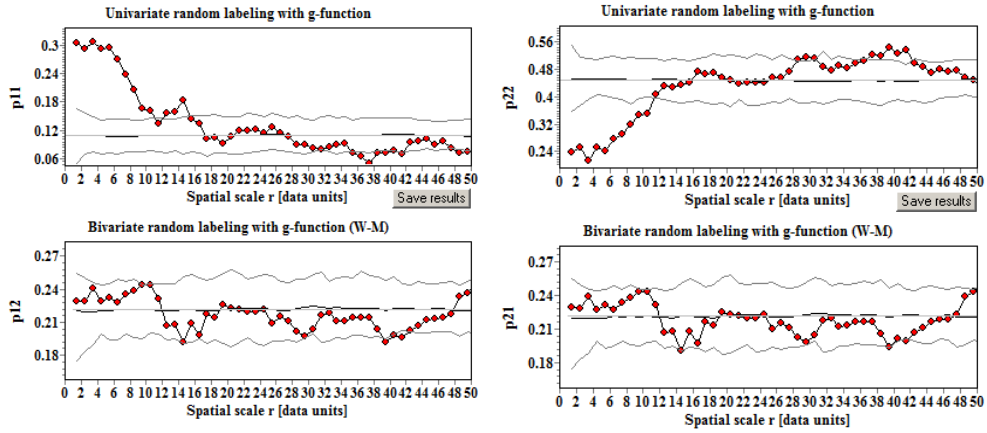
12. If you select the *L*-function before selecting the random labeling null model you can also view the analogous test statistic based on the *L*- or *K*-functions.

15. The marks of the 600 points of data Book_Fig2_15.dat are constructed; 200 dead individuals and 400 surviving individuals. The probability of an individuals dying was proportional to the number of individuals occurring within a 10 m neighborhood; more isolated individuals thus had a higher probability of surviving.

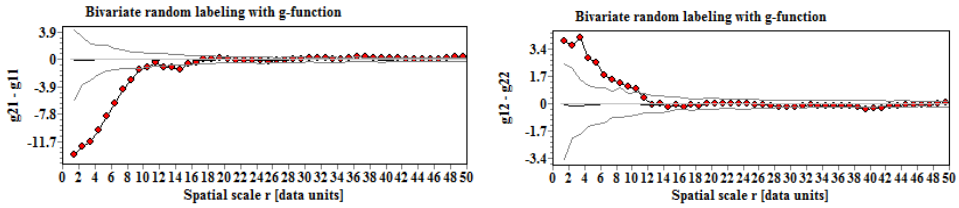
16. The results of the univariate and bivariate pair correlation functions show that the dead individuals are strongly clustered inside all individuals (g11), that the surviving individuals are regularly distributed within all individuals (g22), but that the bivariate relationship between surviving and dead (g12, g21) does not show departures from random labeling:



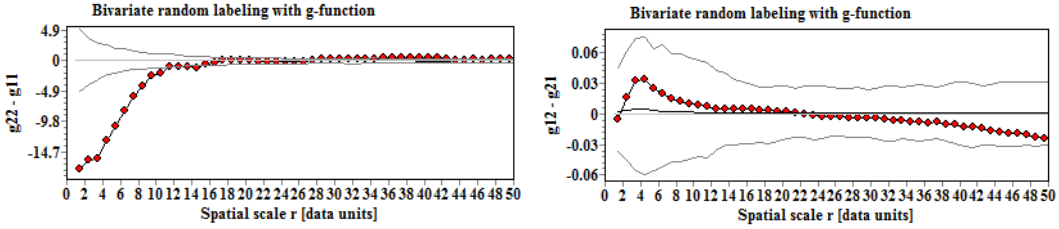
17. When removing the signal of the clustering of all individuals (i.e., the pattern of the underlying univariate pattern) by using the corresponding mark connection functions we can see these results much clearer:



18. Instructive are also the summary statistics $g_{21}(r)$ - $g_{11}(r)$ and $g_{12}(r)$ - $g_{22}(r)$. The $g_{21}(r)$ - $g_{11}(r)$ shows that the neighborhood density of dead individuals is much lower around surviving individuals (g21) than dead individuals (g11). The $g_{12}(r)$ - $g_{22}(r)$ shows that the neighborhood density of surviving individuals is somewhat higher around dead individuals (g12) than surviving individuals (g22):

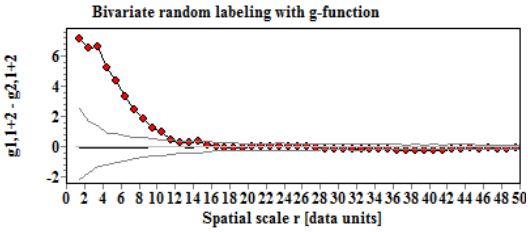


19. The summary statistics $g_{22}(r)$ - $g_{11}(r)$ shows that the dead individuals are much more clustered than the surviving ones



and the summary statistics $g_{12}(r)$ - $g_{21}(r)$ shows that edge effects are unimportant. Note that the $g_{12}(r)$ and $g_{21}(r)$ are the same except small edge effects that arise for example if many dead individuals are close to an edge of the observation window, but not surviving individuals.

20. Finally, the summary statistic $g_{1,1+2}(r)$ - $g_{2,1+2}(r)$ which was especially designed to detect density dependent effects in mortality shows that the neighborhood density of surviving and dead individuals (indicated by subscript 1+2) is much higher in neighborhoods around dead individuals than around surviving individuals. Thus, the more pre-mortality individuals in the neighborhood of an individual, the higher the risk of mortality. As expected this effect is strong up to 10m.



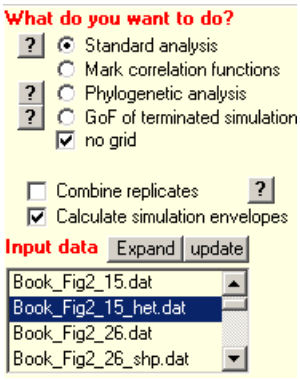
5.1.3 Local random labeling

Random labeling analysis may also be impacted by heterogeneity. In this case the value of the marks may be influenced by environmental covariates and we may observe systematic spatial trends in the marks. For example, the proportion of dead individuals may be larger at the eastern part of an observation window than at the western part.

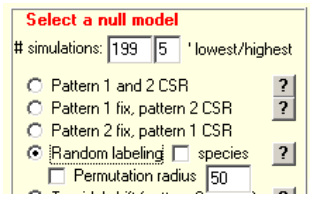
The effect of a large-scale heterogeneity in the marking can be factored out in a similar way as using the heterogeneous Poisson process with kernel intensity estimate for factoring out large scale heterogeneity in univariate patterns. While the marks in standard random labeling are shuffled in a way that the mark of each point can be exchanged with the mark of each other point in the entire observation window, localized random labeling exchanges only marks of points which are located closer than a given distance *R*. This removes any small-scale correlation structure of the marks, but maintains their observed large-scale correlation structure. Technically, all n_{1+2} points i of the marked pattern are numbered and the entries of the array *nr*[i] that runs from 1 to n_{1+2} are randomly permutated in a way that the coordinates of all points i and j = *nr*[i] are not farther away than distance *R*.

Local random labeling, example Book_Fig2_15_local.res

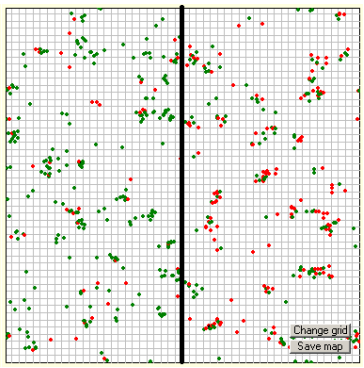
The example pattern Book_Fig2_15_het.dat is the same univariate pattern as used in the previous example, but the marking is different. The points in the left part of the observation window (x-coordinate <250) have a mortality rate of 0.08 and the points of the right part of the observation window (x-coordinate > 250) have a mortality rate of 0.2533.



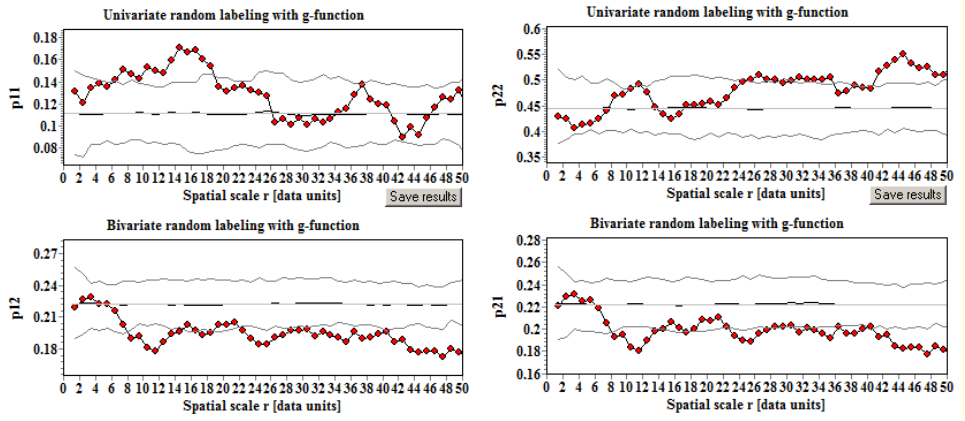
1. Select "**Standard analysis**" in window What do you want to do?
2. Highlight a data file in Input data file ("Book_2_15het.dat" in the example).
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 5 in the menu "Which method will you use"
6. Accept selection of neighborhood ranks for estimation of $D^k(r)$ (will not be used in random labeling analyses).
7. To access the standard analysis mode for **qualitatively marked patterns** (i.e., random labeling analysis; **data type 4**) you must enable the checkbox "Calculate simulation envelopes" in the window What do you want to do? and select the random labeling null model in the Select a null model window.
8. Press button "**Calculate Index**" and *Programita* runs the simulations of the random labeling null model.



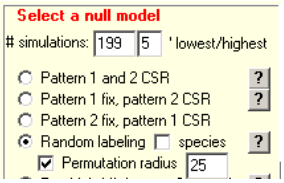
9. You can observe the lower mortality rate at the left side of the observation window:



10. Consequently, the test statistics show departures from random labeling which are difficult to interpret:

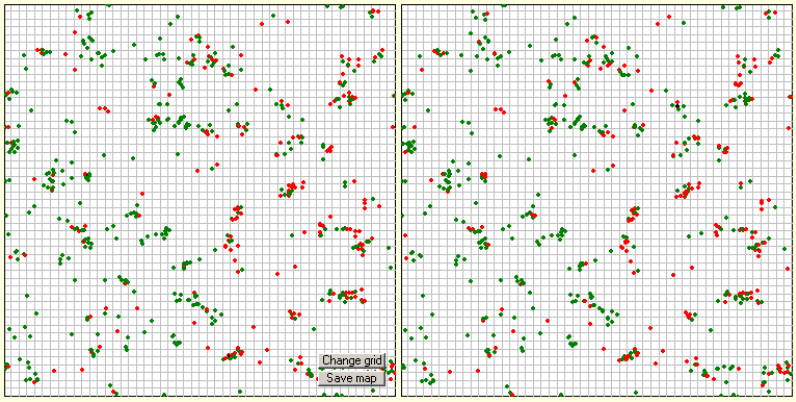


11. To conduct local random labeling click the checkbox "**Permutation radius**" and insert an appropriate radius *R* (25m in our case). Only the marks of points that are less than 25m apart are shuffled. This null model therefore maintains approximately the large-scale structure of the marks (i.e., the higher mortality rate on the right side of the observation window).

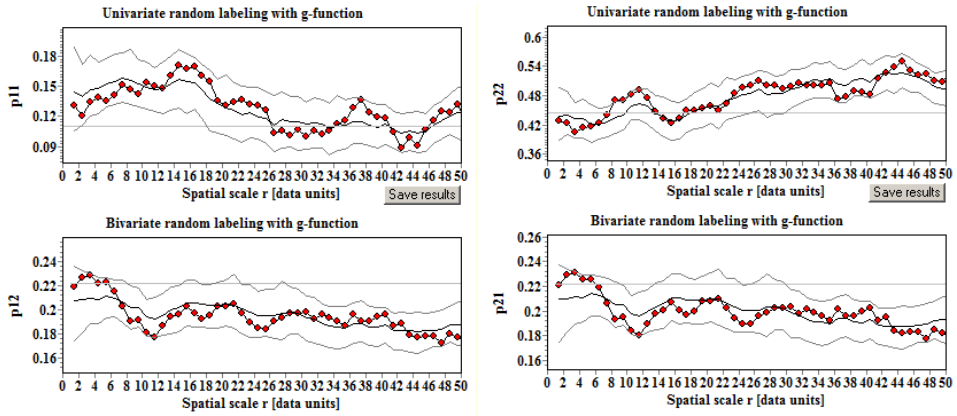


12. Press button "**Calculate Index**" and *Programita* runs the simulations of the random labeling null model:

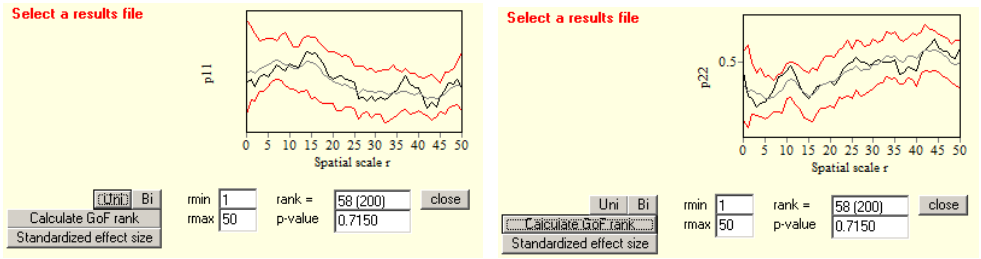
You can observe that the randomization now maintains the large-scale structure of the marks (i.e., the higher mortality rate on the right side of the observation window):



13. The summary statistics do now not show significant departures from the local random labeling null model, but the expectation under this null model are not centered on the expectations for random labeling (which are indicated by a grey horizontal line):



You can verify with the GoF test that there is no significant departure from the null model over distance interval 1-50m:



14. It is clear that the radius R should not be too large compared with the scale of the heterogeneity. However, isolated points (with distances to the nearest neighbor < R) maintain under this null model their mark.

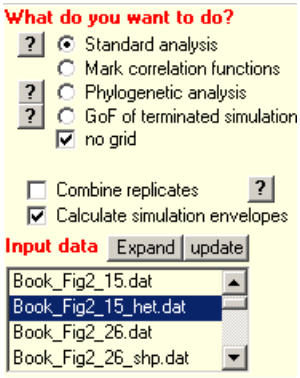
5.1.4 Random labeling with covariate

Random labeling analysis may also be impacted by heterogeneity. In this case the value of the mark may be influenced by environmental covariates and we may observe systematic trends in the marks. For example, the proportion of dead individuals may be larger at the eastern part of an observation window than at the western part.

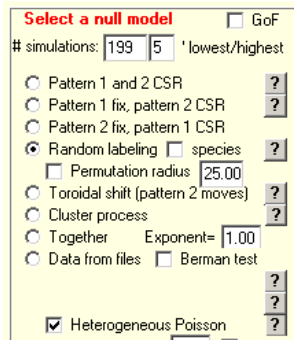
A second option to consider such first-order heterogeneity in the marks is to use a covariate that describes how the probability of mortality changes in dependence on the location **x**. *Programita* offers you the possibility to read an intensity function $\lambda(\mathbf{x})$ that governs the probability of occurrence of one of the two types of points (e.g., the probability of mortality).

Random labeling with covariate, example Book_Fig_2_15_cov.res

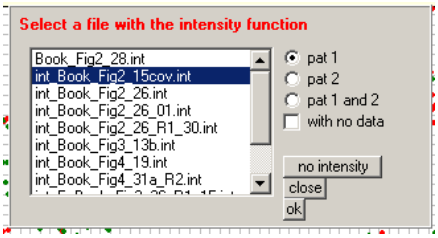
We use the pattern Book_Fig_2_15_het.dat as example. The points in the left part of the observation window (x-coordinate <250) have a mortality rate of 0.08 and the points of the right part of the observation window (x-coordinate > 250) have a mortality rate of 0.25.



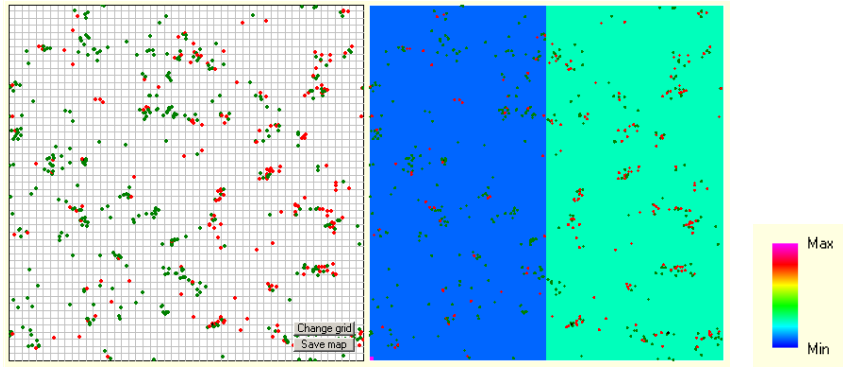
1. Select "**Standard analysis**" in window What do you want to do?
2. Highlight a data file in Input data file ("Book_2_15het.dat" in the example).
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 5m in window "Which method will you use"



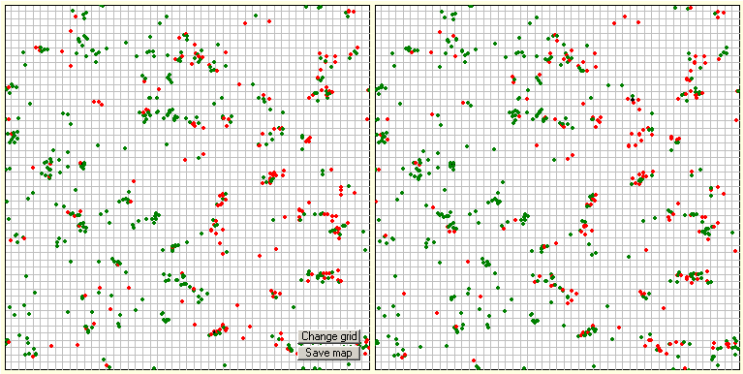
6. Accept selection of neighborhood ranks for estimation of $D^k(r)$ (will not be used in random labeling analyses).
7. To access the standard analysis mode for **qualitatively marked patterns** (i.e., random labeling analysis; **data type 4**) you must enable the checkbox "**Calculate simulation envelopes**" in the window What do you want to do? and select the "Random labeling" in the Select a null model window.
8. To use the covariate press button "**Heterogeneous Poisson**" in the Select a null model window and highlight the intensity file (int_Book_Fog2_15cov.int in the example).
9. Select "**pat 1**" because the dead individuals are pattern 1 (i.e., the type 1 points). If you select "pat2" the event will be surviving and the probability of the intensity function will be applied to survival and not to mortality:



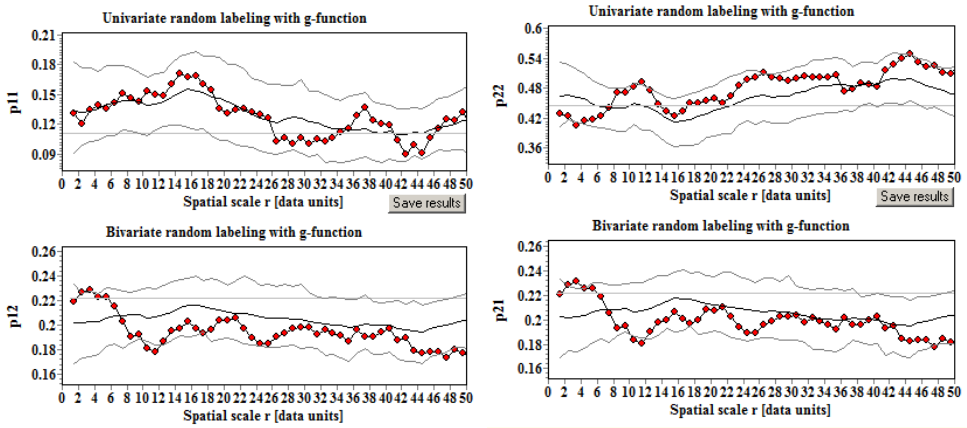
10. *Programita* then shows the intensity function together with the points. The intensity function yields the probability that the event of pattern 1 (here dead) occurs (0.08 at the western part and 0.2533 on the eastern part of the observation window):



11. Now click "**Calculate Index**" to run the simulations of the heterogeneous random labeling null model. You can observe that the randomization maintains the large-scale structure of the marks (i.e., the higher mortality rate on the right side of the observation window):



12. Consequently, the test statistics show no departure from random labeling and similarly to localized random labeling, the expectation under the null model (black line) differs from that under random labeling (grey horizontal line):



5.1.5 Trivariate random labeling

The standard random labeling explores the statistical properties of the marking process. In many cases, the marking of a given pattern may be dependent on the presence of a third, antecedent pattern. For example, mortality of saplings in tropical forests may depend on the proximity of large trees. Trivariate random labeling explores the influence of an antecedent pattern on the marking of a qualitatively marked pattern.

I included trivariate random labeling into the 2004 *Programita* manual using the name "Random labeling under antecedent condition". Trivariate random labeling based on the *K*-function was first published by Marcelino de la Cruz in Spanish [de la Cruz Rot M. 2006. Introducción al análisis de datos mapeados o algunas de las (muchas) cosas que puedo hacer si tengo coordenadas. Ecosistemas. 2006/3], later in De la Cruz et al. (2008) using the name "independent labeling" and the *Programita* implementations based on the mark connection functions were published in Biganzoli et al. (2009), Jacquemyn et al. (2010) and Raventos et al. (2010).

The null model is again random labeling, but the summary statistic considers the impact of individuals of the third antecedent pattern on the marking of the qualitatively marked pattern. In this case, we have individuals of the antecedent pattern (subscript a), dead individuals (subscript 1) and surviving individuals (subscript 2) of the marked pattern. The summary statistic estimates the probability of mortality (or survival) of the individuals of the marked pattern as a function of distance *r* from the individuals of the antecedent pattern:

$$p_{a,1}(r) = \frac{\lambda_1}{(\lambda_1 + \lambda_2)} \frac{g_{a,1}(r)}{g_{a,1+2}(r)}$$

The quantities λ_1 and λ_2 are the partial intensities of the dead and surviving individuals, respectively, and $g_{a,1+2}(r)$ and $g_{a,1}(r)$ are the bivariate pair correlation functions measuring the intensity normalized neighborhood density around antecedent individuals (a) of the combined pattern of surviving and dead individuals (1+2) and dead individuals (1), respectively. More details on trivariate random labeling are provided in sections 3.1.6.2 "Trivariate Random Labeling (Data Type 5)" and 4.4.1.6 "Trivariate Perspective" in Wiegand and Moloney (2014). Note that the summary statistic used for trivariate random labeling can be considered a mark connection function (equation 3.82) in Wiegand and Moloney (2014) or a mark correlation function (equation 3.83) in Wiegand and Moloney (2014).

Programita offers two equivalent implementations of trivariate random labeling, one based on the grid-based mode and one based on the mark correlation function.

Grid-based trivariate random labeling

You can run the trivariate random labeling using the grid-based implementation of *Programita*. The continuous summary statistic can be very well approximated with an underlying grid for distances larger than the grid size. The coordinates are given here as coordinates of a grid that runs from 1 to 1000:

```
 1  1000  1  1000  1001
543  952  1  0
...
383  333  1  0
  0    0  1  0
  2  550  0  0
…
993  766  0  0
491  235  0  1
  3  772  0  1
…
```

where the first line gives the size of the observation window (1000 × 1000 units in the example) and the total number of points (= number of lines following the header).

The points of the antecedent pattern are coded as

```
543  952  1  0
```

with the coordinates (giving the number of the grid cell in x- and y-direction) and as “1 0” in the third and forth column, respectively.

The points of the event (e.g., dead) in the qualitatively marked pattern are codes as:

```
491  235  0  1
```

with the coordinates (giving the number of the grid cell in x- and y-direction) and as “0 1” in the third and forth column, respectively.

The points of the no-event (e.g., surviving) in the qualitatively marked pattern are codes as:

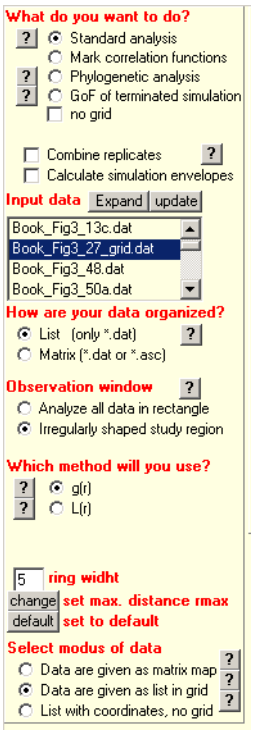
```
  2  550  0  0
```

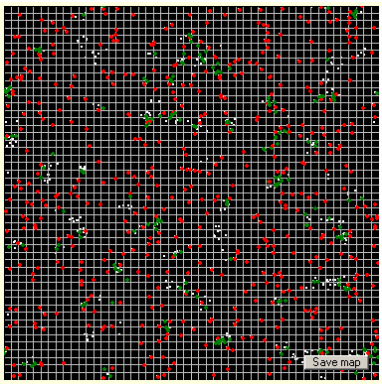
with the coordinates (giving the number of the grid cell in x- and y-direction) and as “0 0” in the third and forth column, respectively.

If there are two of more individuals of the antecedent or qualitatively marked pattern within one cell, each individual must appear individually in the list. *Programita* then reads the list of individuals and applies random labeling over the qualitatively marked pattern.

In the grid-based version of *Programita* I managed to handle 3 patterns at the same time with a trick based on irregularly shaped observation windows. The observation window was reduced to cells that contained points, and the random labeling was conducted between cells coded with “0 1” and “0 0”. With small cell sizes the continuos functions were well approximated.

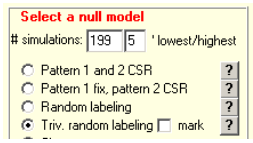
1. Execute *Programita*.
2. Highlight data file Book_Fig3_27_grid.dat you want to analyze in Input data file.
3. Select "**Data are given as list in grid**" in Select modus of data
4. Select a ring width of 5 in the menu "Which method will you use"
5. Click button "change" below to set maximal distance *r* to be analyzed. Insert 100 in small box that opens and then the small **ok** button.
6. Press button "**Calculate Index**"
7. Select "**Irregularly shaped study region**" in window Observation window
8. Press button "**Calculate Index**". *Programita* now shows a somewhat unusual plot of the data:

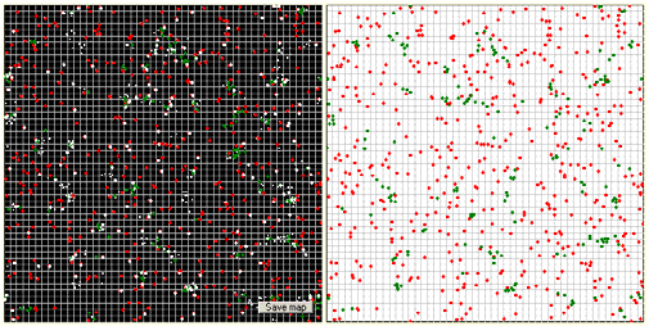




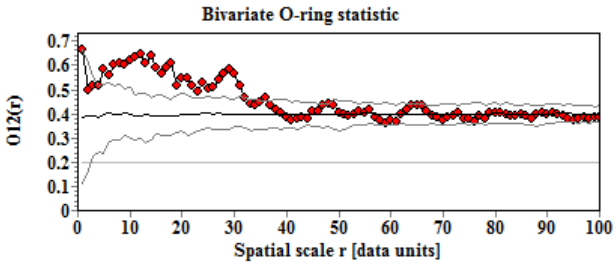
where the points of the antecedent pattern are shown in red, the "event" of the qualitatively marked pattern is shown in green, and the no-event in white. All cells which have no point are black. Thus, basically, the observation window is reduced to the cells which contain points.

9. Click the checkbox "**Calculate simulation envelopes**" to be found in the menu "What do you want to do?" on the top left of the interface.
10. Select null model "Triv. random labeling".
11. Press button "**Calculate Index**". *Programita* now conducts the random labeling of the qualitatively marked pattern, shown are the re-locations of the event:





12. *Programita* now shows the results of the trivariate random labeling analysis:



The label mortality is not random. The antecedent pattern indeed increases the risk of mortality within 40m distance of the points of the antecedent pattern.

Trivariate random labeling based on mark correlation functions

The summary statistic of trivariate random labeling can be interpreted as a bivariate mark correlation function (equation 3.83 in Wiegand and Moloney (2014):

$$\hat{p}_{al}(r) = \frac{\hat{\rho}_{al}(r)}{\hat{\rho}_{a,l+m}(r)} = \frac{\sum_{i=1}^{n_a}\sum_{j=1}^{n_{l+m}} \mathbf{C}_l(\mathbf{x}_j) \times k(\|\mathbf{x}_i - \mathbf{x}_j\| - r)}{\sum_{i=1}^{n_a}\sum_{j=1}^{n_{l+m}} k(\|\mathbf{x}_i - \mathbf{x}_j\| - r)}$$

where the $\mathbf{x}_i$ are the points of the antecedent pattern *a* and the $\mathbf{x}_j$ are the points of the qualitatively marked pattern. The estimator basically visits all pairs of points $\mathbf{x}_i$ of the antecedent pattern and $\mathbf{x}_j$ of the qualitatively marked pattern which are located at distance *r* [(selected by the kernel function *k*()] and estimates the mean value of the test function $\mathbf{C}_l(\mathbf{x}_j)$ over these point pairs. The test function is only a function of the mark of the qualitatively marked pattern and yields 1 if the event occurred for point $\mathbf{x}_j$ and zero otherwise. If the event is dead, this summary statistic thus estimates the mean proportion of dead individuals of the qualitatively marked pattern at distance *r* of the individuals of the antecedent pattern *a*. This is a so-called *r*-mark correlation function.

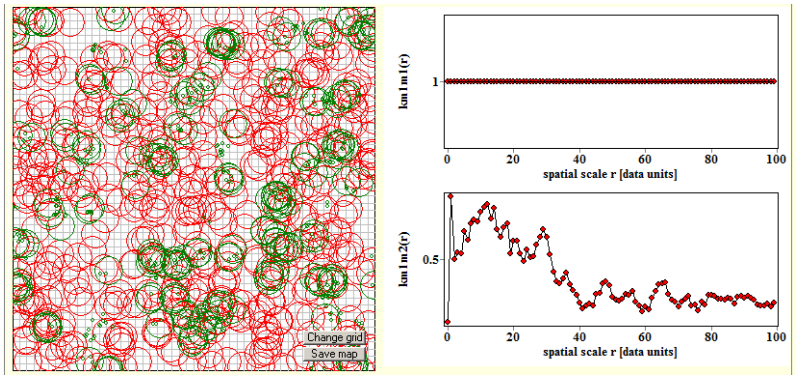
The coding of the data file follows that of bivariate mark correlation functions. It is **data type 9** (a bivariate pattern with one quantitative mark), the data files must be an ASCII file with the *. mcf extension and have always the following format:

```
0  1000    0  1000  1001
543   952   1   1   0
...
383   333   1   1   0
  0     0   1   1   0
  2   550   2   0   0
…
993   766   2   0   0
491   235   2   0   1
  3   772   2   0   1
…
```

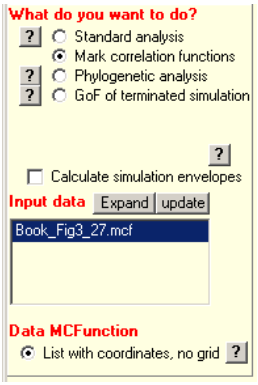
where the third column must be "1" for all points of the antecedent pattern 1 and "2" for all points of the qualitatively marked pattern 2, the forth column gives the value of the mark attached to the type 1 point (which is not used in our case, but got a value of 1), and the fifth column the value of the mark attached to the type 2 point. The fifth column yields "1" for the event (e.g., dead) and zero otherwise.

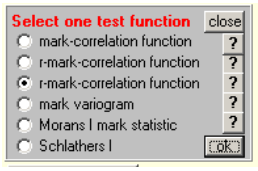
Trivariate random labeling based on mark correlation functions example Book_Fig_3_27_mcf.res

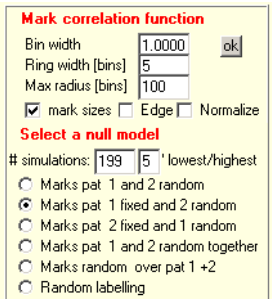
1. Execute *Programita*.
2. Select "**Mark correlation functions**" in window What do you want to do?
3. Highlight data file (Book_Fig3_27.mcf) you want to analyze in Input data file
4. Click "**List with coordinates, no grid**" in MCFunction
5. Provide in the window mark correlation functions the bin width in data units (1), an appropriate ring width (5), and a maximal distance *r* of the analysis (100).
6. Disable "**Normalize**" to get non-normalized mark correlation functions.
7. Click the small **"ok"** button the window mark correlation functions
8. Press button "**Calculate Index**" and *Programita* shows you the pattern and the summary statistics:



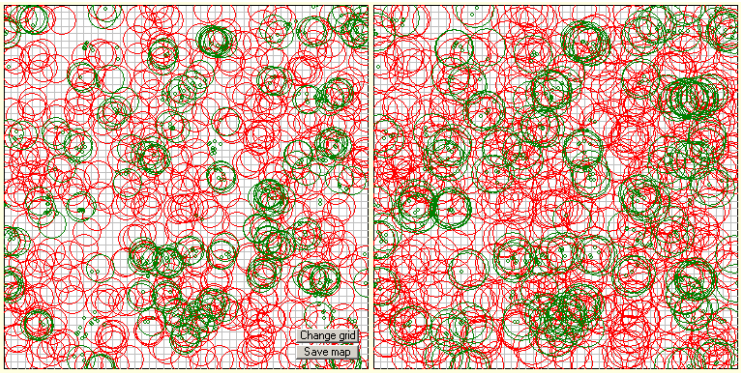
9. Select the third test function "r-mark correlation functions"
10. Check the checkbox "**Calculate simulation envelopes**", select the number of simulations of the null model (199) and the rule for the simulation envelopes (5), and again "100" for the maximal radius
11. Select the null model "**Marks pat 1 fixed and 2 random**" which randomizes only the marks of the qualitatively marked pattern.
12. Press button "**Calculate Index**".





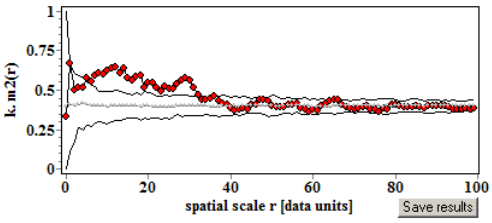


13. *Programita* now shows the data and the simulations of the null model where the red circles are the points of the antecedent pattern 1 with radius proportionally to the mark (1) and the green circles are the points of the second pattern with a large circle for dead (mark 1) and a small circle for surviving (mark 0):

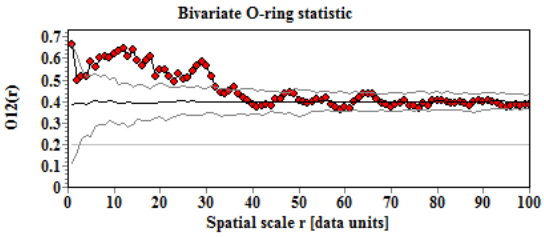


14. After termination of the simulations, *Programita* shows the results of the trivariate random labeling analysis which are virtually identical with that of the grid-based analysis:

mark correlation analysis:



grid-based analysis:



5.1.6 Random labeling for communities

The standard random labeling works for a univariate pattern that carries a qualitative mark. In some cases, however, the data are given by a multivariate pattern (e.g., the locations of all saplings in a tropical forest) and a qualitative mark such as surviving vs. dead. In this case we can run the standard random labeling analysis without regard to the species (i.e., the observed species-specific mortality rates are not conserved). This is called "**community-wide species-blind random labeling** in Wiegand and Moloney (2014)

However, another possibility is to conserve the observed species-specific mortality rates in the random labeling null model. Thus, conventional random labeling is conducted here within each species, but the final test statistics average over all species. This is called "**community-wide species-specific random labeling**" in Wiegand and Moloney (2014). Details can be found in section 4.4.2.3 Community Wide Random Labeling in Wiegand and Moloney (2014).

The coding of the data file is almost the same as for standard random labeling, but an additional fifth column given the species number is required:

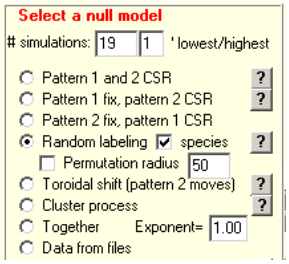
```
0  1000   0  500   2582
 37.76  308.91  0  1  3
 39.92  348.13  1  0  3
144.53   25.32  0  1  4
229.01   64.05  1  0  4
435.43  351.24  0  1  4
506.88  478.12  1  0  5
620.09  488.88  0  1  5
…
```

Community-wide species-specific random labeling, example Book_Fig4_42_species.res

To generate this example we first conducted pattern reconstruction with the underlying univariate pattern of the original data file of Figure 4.42 (the pattern of small saplings of gap species at the BCI forest) and then applied a mortality rate of 35% that depended linearly on the number of neighbors within 10m. Finally, the species labels of the original data where then randomly distributed over the points. Thus, the pattern shows density dependent mortality but there is no difference between species-specific and species-blind random labeling.

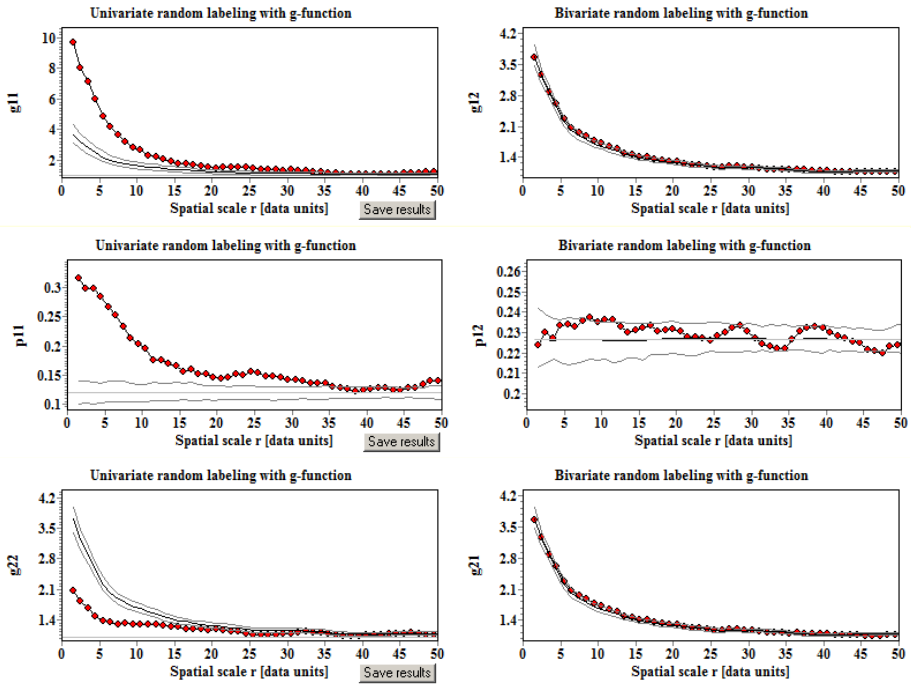
1. Select "**Standard analysis**" in window What do you want to do?
2. Highlight a data file in Input data file ("Book_Fig4_42_rand.dat" in the example).
3. Select "**no grid**" in What do you want to do?
4. Select bin of 1m window Select a new cell size
5. Select a ring width of 5m in window "Which method will you use"
6. Accept selection of neighborhood ranks for estimation of $D^k(r)$ (will not be used in random labeling analyses).
7. Enable the checkbox "Calculate simulation envelopes" in the window What do you want to do? and select the random labeling null model in the Select a null model window.
8. Specify the **number of simulations of the null model** (199 in the example) and the rule for the estimation of **simulation envelopes** (here the 5^{th} lowest and highest values of the summary statistic of the 199 null model data sets).

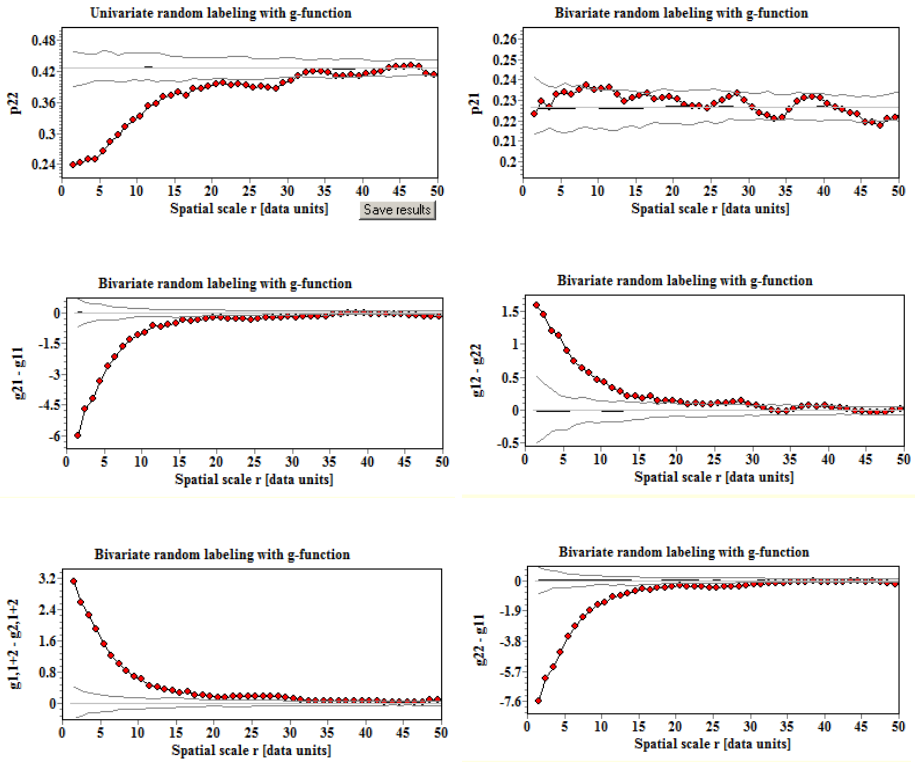
9. Check the checkbox "**species**" beside of "Random labeling":



10. Press button "**Calculate Index**" and *Programita* shows the observed and simulated pattern:

11. The results of the simulations of the species-specific random labeling null model show indeed that effects are not species specific. This is recognized because the expectation of the null model (the black line) is identical to the expectation of community-wide species-blind random labeling (i.e., standard random labeling without regard to the species label; the grey line):




Univariate random labeling with g-function
p22
Spatial scale r [data units]
Save results
Bivariate random labeling with g-function
p21
Spatial scale r [data units]
Bivariate random labeling with g-function
g21 - g11
Spatial scale r [data units]
Bivariate random labeling with g-function
g12 - g22
Spatial scale r [data units]
Bivariate random labeling with g-function
g1,1+2 - g2,1+2
Spatial scale r [data units]
Bivariate random labeling with g-function
g22 - g11
Spatial scale r [data units]

6 Analysis with mark correlation functions

Because the data structure of quantitatively marked patterns is different from that of the standard univariate, bivariate and qualitatively marked patterns I introduced an own *.mcf extension for the data files of mark correlation analysis.

Programita allows for mark correlation analysis of the following data structures:

- univariate patterns with one quantitative mark (data type 6)
- univariate patterns with two quantitative marks (data type 7)
- qualitatively marked patterns with one quantitative mark (data type 8)
- bivariate patterns with one quantitative mark (data type 9)

Data type 9 includes also trivariate random labeling as special case.

The estimators for the mark correlation functions are explained in detail in Section 3.17 "Summary Statistics for Quantitatively Marked Point Patterns" of Wiegand and Moloney (2014). All estimators for mark correlation functions included in *Programita* are based on real distances between pairs of points.

6.1.1 Analysis of univariate patterns with one mark (data type 6)

Univariate quantitatively marked patterns comprise the coordinates of a univariate pattern, but each point carries an additional quantitative mark that characterizes the ecological object that is idealized as point. The quantitative mark is usually a continuous attribute such as the size of a tree, but can also be an integer such as the number of seeds of a tree.

The major interest in analysis of (univariate) quantitatively marked patterns (with one qualitative mark) is in revealing potential spatial correlation structure of the marks, conditional on the underlying univariate pattern. For example, are trees which are closer together usually smaller than the average tree or is the number of seeds of trees which are closer together higher than that of more isolated trees? The basic null model for this data type is the so-called **independent marking null model** which randomly shuffles the marks over the points, thus removing all potential spatial structure in the marks. Section 3.1.7.1 in Wiegand and Moloney (2014) provides examples for the different analyses of qualitatively marked patterns that are useful in ecology.

6.1.2 Data preparation for data type 6

The data files must be an ASCII file with the *.mcf extension with the following format (the example shows the first lines of the data file Book_Fig2_16a.mcf):

```
0   500   0   500   600
249.75   451.80   1   3.725   0
434.30   272.60   1   4.726   0
482.65    35.20   1   1.826   0
100.10   196.60   1   1.983   0
245.00   117.45   1   2.012   0
423.40    38.05   1   2.552   0
222.85   349.05   1   1.651   0
 32.15    21.55   1   3.040   0
208.65   352.10   1   4.177   0
 92.95   214.85   1   2.094   0
112.40   452.65   1   3.256   0
…
```

where first line gives the size of the observation window (500 × 500 units in the example) and the number of points in the pattern.

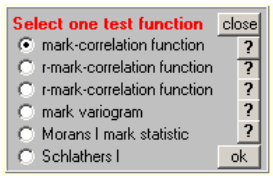
- the first two columns of the following lines are the coordinates of the points, the third column indicates the pattern and must be "1" for all points because this data structure is based on an univariate pattern.
- The forth column carries the mark of the point.
- The fifth column is reserved for a second mark or for the mark of a second pattern and must be therefore "0".

The data file must be a space or tab delimited ASCII file with the *.dat extension. If you use Excel, there is a simple, but obviously generally unknown, way of saving files of a given type with a given extension:

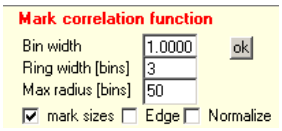
1. Prepare the data file in Excel following the instructions above.
2. Then save as a tab delimited text file, but write "name.mcf" for the name (usually you would only write name and end up with a file named name.txt). The quotation marks are important because they force Excel to save the comma delimited file under the name name.mcf.

6.1.3 Steps data type 6 (Book_Fig2_16.res)

Programita estimates for data files of the *.mcf type several adapted test statistics based on mark correlation functions which are described in detail in section 3.1.7.1 "Univariate Quantitatively Marked Pattern (Data Type 6)" of Wiegand and Moloney (2014):

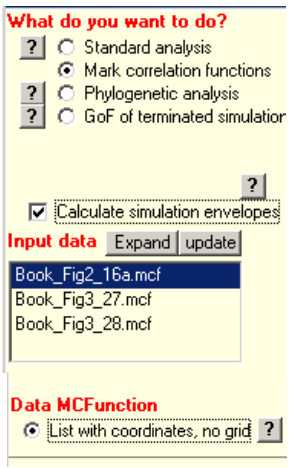


- the mark correlation function $k_{mm}(r)$
- the *r*-mark correlation function $k_{m\,.}(r)$
- the *r*-mark correlation function $k_{.\,m}(r)$
- the mark variogram $\gamma_{mm}(r)$
- a Moran's I type mark statistics $I_{mm}(r)$
- Schlather's correlation function $I_{mm}(r)$

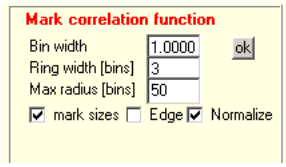


The default for mark correlation functions is to use the normalized functions, i.e., $k_t(r) = c_t(r)/c_t$ where c_t is the normalization constant for a given test function *t*. However, you can select the non-normalized mark correlation functions $c_t(r)$ by disabling the checkbox "**Normalize**".

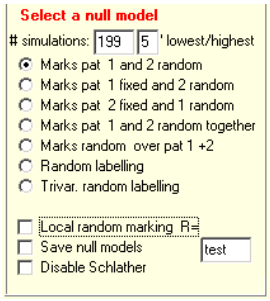
The mark correlation mode can be accessed with the following sequence of actions:



1. Select "**Mark correlation functions**" in window What do you want to do?
2. Highlight data file you want to analyze in Input data file. In the example it is file "Book_Fig2_16a.mcf"
3. Click "**List with coordinates, no grid**" in MCFunction
4. Provide in the window mark correlation functions the bin width in data units, an appropriate ring width, and a maximal distance *r* of the analysis. Select in the example a bin of 1m, and a ring width of 3m.
5. If you use a bin of 1 unit, you can later use the function "**Combine replicates**" to load the results of the analysis, to change the bin, and to use the corresponding cumulative summary statistic (see below "View results of mark correlation analysis").
6. Disable "**Normalize**" if you want to use the non-normalized mark correlation functions. The default is "Normalize"
7. Check "**Edge**" if you want to use the Ripley edge correction. Default is no edge correction. Note that edge correction is not required for mark correlation functions.
8. Check "Calculate simulation envelopes" in window What do you want to do? The subwindow "Select a null model" appears.



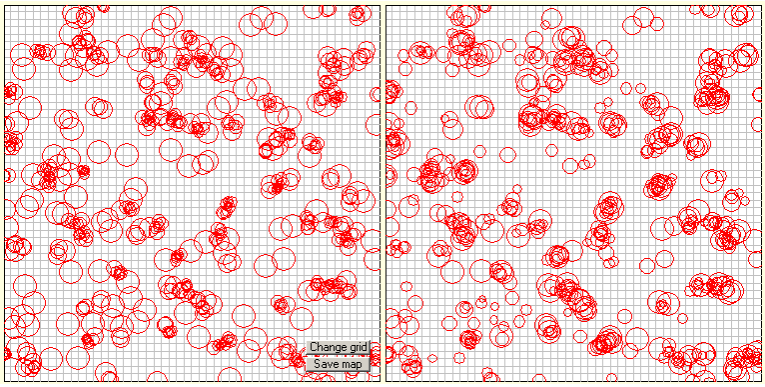
9. Select an appropriate null model, the number of simulations of the null model (199), and the rule for the simulation envelopes (5' lowest and highest). Note that not all null models are appropriate for all mark correlation function data type and that they must be selected with care.



   Select for univariate patterns with one quantitative mark the null model "**Marks pat 1 and 2 random**" that shuffles the mark randomly over the points.

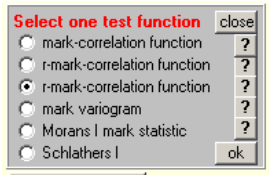
10. The estimation of Schlathers I requires double calculations; if you do not need this summary statistic you can speed up *Programita* by clicking "**Disable Schlather**".

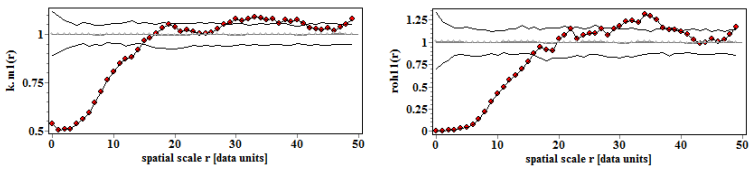
11. Press button "**Calculate Index**" and *Programita* shows the observed and simulated pattern. The area of the disk that represents a point is proportionally to the mark:



   You notice that the points do not change their location in the null model simulation (right), but that the size of the points changes because the mark of all points is randomly shuffled.

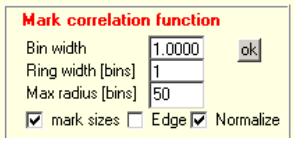
12. Use the radio buttons of the window Select one test function to select a mark correlation function and click the small "**ok**" button to get the result graphic:





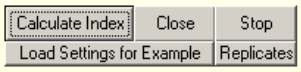
   The graph for the bivariate mark correlation is empty because the data were univariate. The results show that the mark of a point which is located at distance *r* from another point of the pattern is for distances $r < 16$m smaller than expected (*r*-mark correlation function; left) and that two points that are located closer than 16m have marks which are more similar than expected (mark variogram; right).

View results of mark correlation analysis



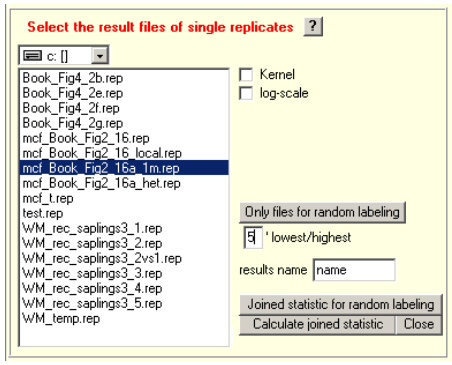
Note that this function works only correctly if you select ring width of one unit (i.e., you obtain non-overlapping rings). Thus, if you want to use it you have to use a ring width of 1 unit.

After conducting a mark correlation analysis you should save the results with button "**Save results**". *Programita* creates a results file names.res that also contains all settings of your analysis and an additional file name.rep which allows you to view and save the results for all mark correlation functions.

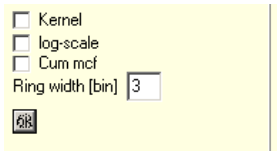


You can access the procedure for loading the results with button "**Replicates**". If you conduct your analysis with a ring width of 1, the **Replicates** option allows you additionally to estimate the mark correlation function with different ring widths and to estimate the analogous cumulative mark correlation functions. To access this option follow the steps below:

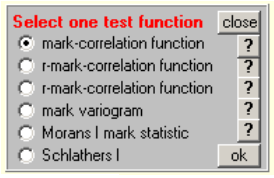
1. select "**Replicates**"
2. highlight the *.rep results file you want to analyze in the window Select the result files of single replicates (mcf_Book_Fig2_16a_1m.rep)
3. Select the rule for the simulation envelopes (insert integer before **'lowest/highest)**. For example, if you conducted 199 simulations of the null model you may select 5 (i.e., the simulating envelopes are the 5th lowest and highest values).



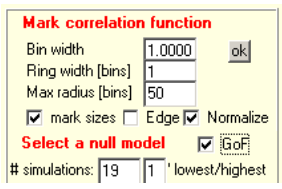
4. Click button "**Calculate joined statistic**" and *Programita* shows you the results of the analysis.
5. To change the ring width or to use the **cumulative mark correlation function** select the ring width after "**Ring width [bin]**" or enable the check box "**Cum mcf**" for the cumulative mark correlation function [see section 3.1.7.2 "Univariate Marked K-Functions (Data Type 6)" in Wiegand and Moloney 2014]. You can also plot the results on a logarithmic x-axis with check box "**log-scale**". Finally, press the small **ok** button and *Programita* shows the results with the modified estimator.



6. In the window Select one test function you can view the results of the different mark correlation functions based on the modified estimator.



7. Using the button "**Save results**" you can save the results for this mark correlation function as *.res file.
8. You can also conduct the GoF test by checking the small box "**GoF**". To this end first select in the window that appears the distance interval of the GoF test (t0 and t1), the button "**Uni**" or "**Bi**" depending if the analysis is uni- or bivariate, and finally the button "**Calculate GoF rank**" to get the rank and the P-value of the test.



6.1.4 Local independent marking, data type 6

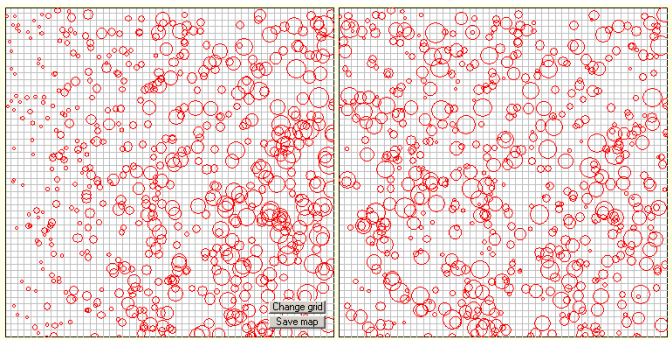
Mark correlation analysis may also be impacted by heterogeneity. In this case the value of the marks may be influenced by environmental covariates and we may observe systematic spatial trends in the values of the marks. For example, the size of the trees may be larger at the eastern part of an observation window than at the western part.

The effect of a large-scale heterogeneity in the marking can be approximately factored out in the same way as for random labeling. While the marks in standard independent marking null model are shuffled in a way that the mark of each point can be exchanged with that of any other point in the entire observation window, localized independent marking exchanges only marks of points which are located closer than a given distance *R*. This removes any small-scale correlation structure in the marks, but maintains their observed large-scale correlation structure. Technically, all n_{1+2} points i of the marked pattern are numbered and the entries of the array *nr*[i] that runs from 1 to n_{1+2} are randomly permutated in a way that the coordinates of the point pair i and j = *nr*[i] are not farther away than distance *R*.

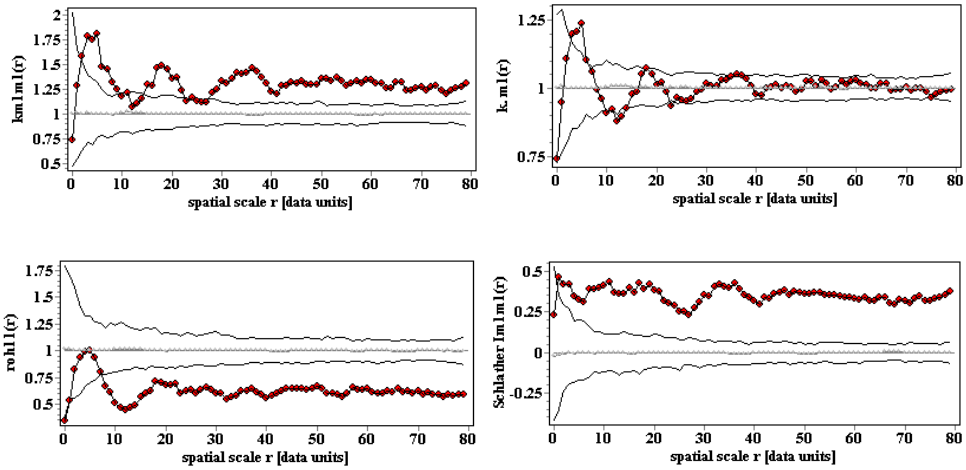
Local independent marking, example CSR_grad_local.res

This example uses the data set CSR_grad.mcf that is based on a random pattern with random marks (and mean mark μ) within a 500m × 500m observation window, but afterwards the marks of the points were multiplied by factor x/500. Thus, the average size of the marks in dependence on the x-value is $m(x) = \mu\, x/500$ Thus, there is a systematic gradient in the marks from west to east where the marks become increasingly larger when moving eastward. However, except this gradient the marks do not show any spatial correlations.

1. Select "**Mark correlation functions**" in window What do you want to do?
2. Highlight data file CSR_grad.mcf in Input data file
3. Click "**List with coordinates, no grid**" in MCFunction
4. Press button "**Calculate Index**"
5. Select in mark correlation functions a bin width of 1, and a ring width of 1 and a maximal radius of 80.
6. Check "Calculate simulation envelopes" in window What do you want to do? and select "**Marks pat 1 and 2 random**". In a first step we use global independent marking. Select the number of simulations of the null model (199), and the rule for the simulation envelopes (5' lowest and highest).
7. Press button "**Calculate Index**" and *Programita* shows the observed and simulated patterns. In the data the marks in the west are smaller than the marks in the east, but not in the null model:

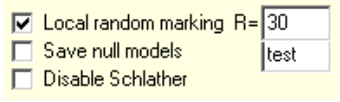


8. Save results with button "**Save results**" and use button "**Replicates**" to view the results of the analysis for a ring width of 5.
9. Use the radio buttons of the window Select one test function to select a mark correlation function and click the small "**ok**" button to get the result graphic:
10. Various of the mark correlation functions show larger scale departures from the independent marking null model (see CSR_grad_5m.res):

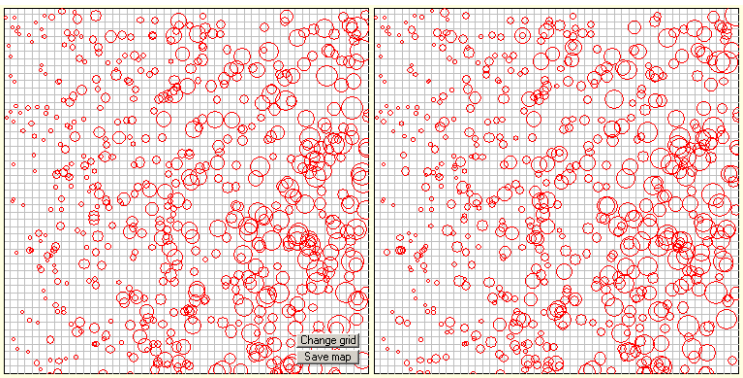


Because the average size of the marks depends linearly on their x-coordinate [i.e., $m(x) = \mu\, x/500$] the mark product of points separated by distance r will be larger than μ^2. Consequently, we observe a positive departure in the mark correlation function $k_{mm}(r)$. However, the x-dependence in the mark is linear in the r-mark correlation functions and therefore averages out. The mark variogram shows negative departures because nearby marks are by construction similar in size, and the Moran's I type correlation coefficient $I_{mm}(r)$ shows for the same reason positive departures. Especially, the marks are strongly correlated in x-direction.

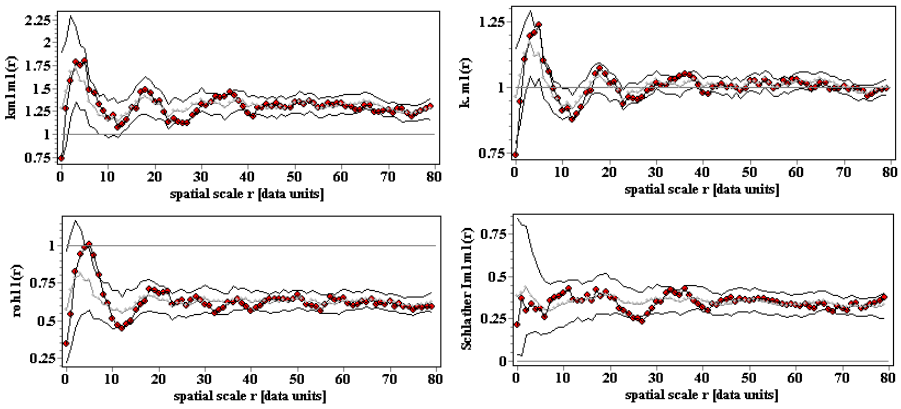
11. To approximately factor out the large-scale heterogeneity click the option "**Local independent marking**" and select an appropriate maximal distance R for points that should switch their marks (select here $R = 30$).



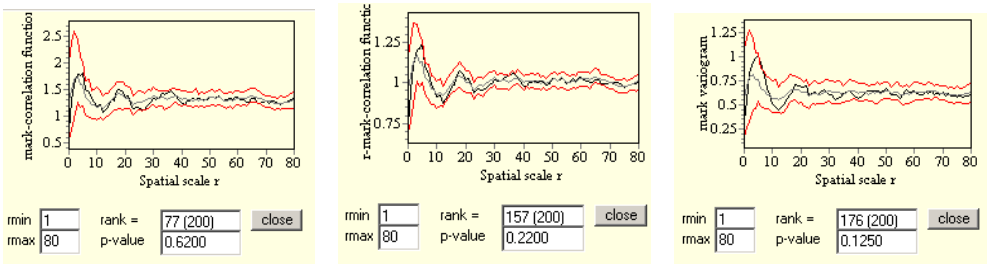
12. Press button "**Calculate Index**" and *Programita* shows the observed and simulated patterns. The null model shuffles the marks only locally and the marks in the east are larger than in the west, now in the data and the null model:



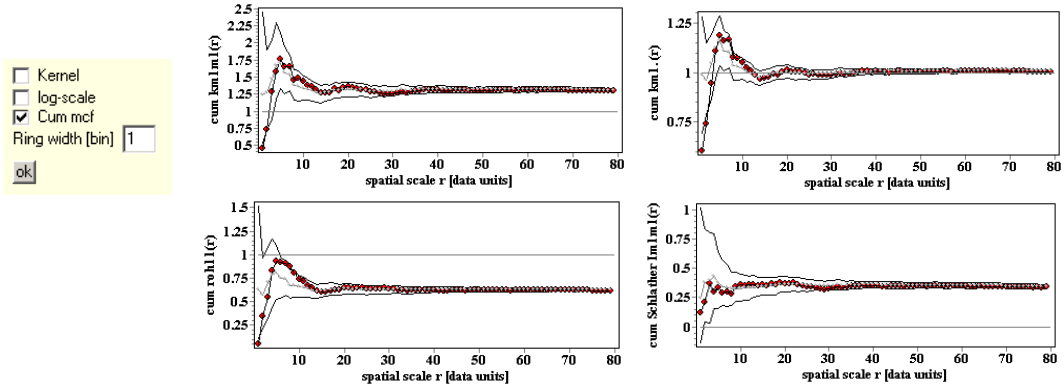
13. Use the combine replicates function to view the results of the analysis for a ring width of 5.
14. Use the radio buttons of the window Select one test function to select a mark correlation function and click the small "**ok**" button to get the result graphic. The expectation of the mark correlation functions under the local independent marking null model differ substantially from that of the independent marking null model and approximate the observed mark correlation functions. This is because the marks do not show other correlations than that imposed by the gradient.



15. Using the GoF test you can verify that there is indeed no departure from the local independent marking null model:



16. If you conducted the analysis with a ring width of one unit and use the "**Combine replicate**" option to view the results of the mark correlation analysis you can also use the cumulative mark correlation functions [see section 3.1.7.2 "Univariate Marked K-Functions (Data Type 6)" in Wiegand and Moloney 2014]. Enable the checkbox "**Cum mcf**" to obtain the cumulative functions:

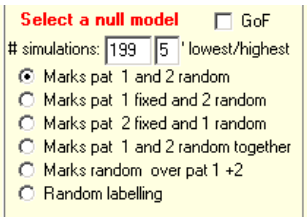


6.1.5 Analysis of univariate patterns with two marks (data type 7)

Quantitatively marked patterns of this type comprise the coordinates of a univariate pattern and **each point carries two additional quantitative marks** that characterize the ecological object that is idealized as point. The quantitative marks are usually a continuous attribute such as the size and height of a tree, but can also be an integer number such as the number of orchids of two species located on host trees, or the number of seeds of two species in feces of seed dispersing animals. Because the bivariate mark variogram is sensitive to differences in the means μ_1 and μ_2 of the two marks you should normalize the marks to yield the same mean. (The other mark correlation functions are independent on the absolute values.)

The major interest in analysis of quantitatively marked patterns with two quantitative marks is to find out whether the two marks show some spatial correlation that depends on the distance *r* between points, conditional on the underlying univariate pattern. In a way this is similar to testing for independence between the two component patterns of a bivariate pattern. For example, the two orchid species may tend to be placed less frequently together on nearby host trees than expected by independent placement, or the seeds of the two species tend to be more frequently placed together in nearby feces than expected.

Depending on the ecological question two types of null models are possible; see section 3.1.7.3 “Two Quantitative Marks Attached to a Univariate Pattern (Date Type 7)” in Wiegand and Moloney (2014):



Null model type 1
For example, if the marks are the number of orchids of two species, we may ask whether they are independently distributed over the host trees. In this case, we can condition on the number of orchids of the first species and shuffle the second mark (i.e., number of individuals of the second orchid species) randomly over the trees of the univariate pattern (**Marks of pat 1 fixed and 2 random**). If none of the two orchid species is antecedent we can also condition on the number of orchids of the second species and shuffle the first mark (**Marks of pat 2 fixed and 1 random**). If appropriate, we may also randomize the locations of both marks (**Marks of pat 1 and 2 random**). This null model thus tests if the placement of the two orchid species on the host trees was spatially independent as opposed by positive or negative associations that could be promoted by species interactions between the two species or by shared or opposed habitat requirements.

Null model type 2
If we analyze the spatial correlation in marks representing the number of seeds of two species in feces, we ask if there is a spatial correlation in the co-occurrence of the two marks (i.e., number of seeds of the two species). In this case we cannot separate the two marks because they occurred together in the same feces. Thus, we need to shuffle the vector of marks of the point i, given by (m_{i1}, m_{i2}), randomly over the points of the univariate pattern (**Marks pat 1 and 2 random together**). Similar augments can be made for example for the case where the marks are the size and the height of a tree. Here the null model simulation must also keep these two properties together.

6.1.6 Data preparation for data type 7

The data files must be an ASCII file with the *.mcf extension with the following format :

```
0  200  0  191  600
 51.14   48.99  12  0.80  0.59
112.80   45.85  12  1.35  1.08
  5.59   61.90  12  0.42  1.73
  6.41   62.55  12  0.61  1.57
 53.95   74.64  12  0.83  0.58
123.54    4.92  12  1.06  1.02
…
```

where the first line gives the size of the observation window (200 × 191 units in the example) and the number of points in the pattern.

- the first two columns of the following lines are the coordinates of the points. The third column has the value "12" for all points which indicates that each point can carry two marks.
- The forth column carries the first mark of the point
- The fifth column carries the second mark of the point.

The data file must be a space or tab delimited ASCII file with the *.dat extension.

6.1.7 Steps data type 7 (DataType7.res)

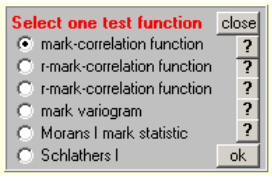
Programita estimates for the first mark the univariate mark correlation functions described above, and additionally the corresponding bivariate test statistics that consider the two quantitative marks.

To understand the bivariate mark correlation functions it is important to note that they involve the first mark m_{i1} of the first point i and the second mark m_{j2} of the second point j. A bivariate mark correlation function for this data type therefore estimates the mean value $c_t(r)$ of a test function $t(m_{i1}, m_{j2})$ taken over all pairs of points i and j that are located at distance r, and normalizes with the mean c_t taken over all pairs of points.

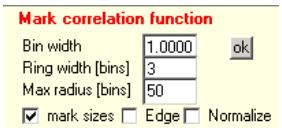
For example, the mark variogram $\gamma_{m1m2}(r)$ estimates the squared difference 0.5 $(m_{i1} - m_{j2})^2$ between the first mark m_{i1} of the first point i and the second mark m_{j2} of the second point j which is located at distance r of the first point. Note that the r-mark correlation functions $k_{m1\,.}(r)$ and $k_{.\,m2}(r)$ are the same as the univariate r-mark correlation functions for marks 1 and 2, respectively, and therefore not of interest for a bivariate analysis.

The bivariate methods are explained in detail in section 3.1.7.3 "Two Quantitative Marks Attached to a Univariate Pattern (Date Type 7)" in Wiegand and Moloney (2014):

- the mark correlation function $k_{m1m2}(r)$
- the r-mark correlation function $k_{m1\,.}(r)$
- the r-mark correlation function $k_{.\,m2}(r)$
- the mark variogram $\gamma_{m1m2}(r)$
- a Moran's I type mark statistics $I_{m1m2}(r)$
- Schlather's correlation function $I_{m1m2}(r)$



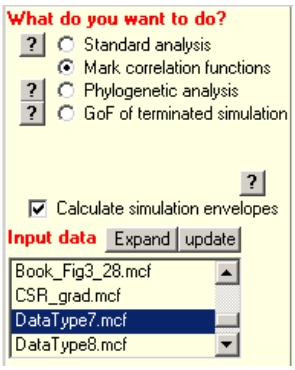
The default for mark correlation functions are the normalized functions, i.e., $k_t(r) = c_t(r)/c_t$ where c_t is the normalization constant for a given test function t. However, you can also use the non-normalized mark correlation functions $c_t(r)$ when disabling the checkbox "**Normalize**".



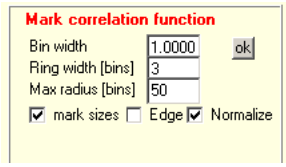
The mark correlation mode can be accessed with the following sequence of actions:

1. Select "**Mark correlation functions**" in window What do you want to do?
2. Highlight data file you want to analyze in Input data file. In the example it is file "DataType7.mcf".

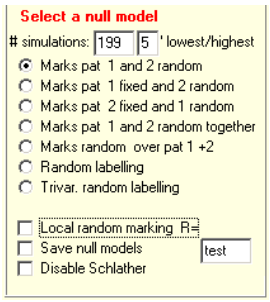
   This data file was generated by using an intensity function $\lambda(\mathbf{x})$ (int_Book_Fig2_26_R1_30.int) and simulating the two marks $m_1(\mathbf{x})$ and $m_2(\mathbf{x})$ for 600 random points (i.e., following CSR) stochastically with $m(\mathbf{x}) = \lambda(\mathbf{x})\ (0.3 + \varepsilon\ ) + \varepsilon$ where ε is a random number equally distributed between 0 and 1. As a consequence the values of the marks are itself spatially correlated (due to the underlying intensity function) and marks m_1 and m_2 are positively correlated. These deterministic relationships are made noisy with the factor ε.



3. Click "**List with coordinates, no grid**" in MCFunction
4. Provide in the window mark correlation functions the bin width in data units, an appropriate ring width, and a maximal distance r of the analysis. Select in the example a bin of 1m, and a ring width of 3m.
5. If you use a bin of 1 unit, you can later use the function "**Combine replicates**" to load the results of the analysis, to change the bin, and to use the corresponding cumulative summary statistic (see "View results of mark correlation analysis").
6. Disable "**Normalize**" if you want to use the non-normalized mark correlation functions. The default is "**Normalize**"
7. Check "**Edge**" if you want to use the Ripley edge correction. Default is no edge correction. Note that edge correction is not required for mark correlation functions.
8. Check "Calculate simulation envelopes" in window What do you want to do? The subwindow "Select a null model" appears.

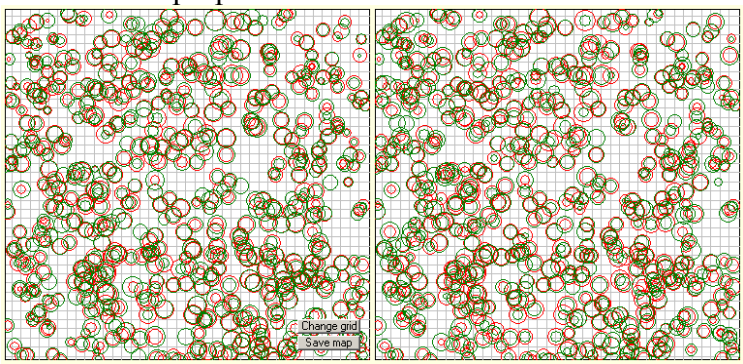


9. Select an appropriate null model, the number of simulations of the null model (199), and the rule for the simulation envelopes (5' lowest and highest). Note that not all null models are appropriate for all mark correlation function data types and that they must be selected with care.



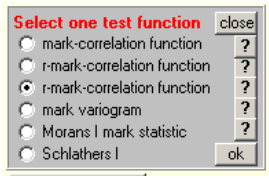
   Select for the null model "**Marks pat 1 fixed and 2 random**" that shuffles the first mark randomly over the points. (You can also use the alternative null model "**Marks pat 2 fixed and 1 random**")

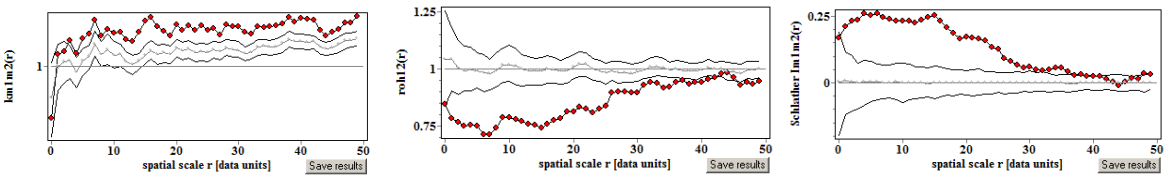
10. Press button "**Calculate Index**" and *Programita* shows the observed and simulated pattern. The red circles represent the first mark and the green circles the second mark, and the area of the disk is proportional to the mark:



   You notice that the points do not change their location, but that the size of the green points changes because the second mark is randomly shuffled over all points.

11. Use the radio buttons of the window Select one test function to select a mark correlation function and click the small "**ok**" button to get the result graphic:





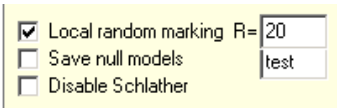
   Note that the *r*-mark correlation functions are not of interest here because they are identical with the respective univariate *r*-mark correlation functions. To obtain the univariate $k_{m1\,.}(r)$ you must select the null model "**Marks pat 2 fixed and 1 random**" or "**Marks pat 1 and 2 random**" that randomizes the first mark.

19. As expected by the construction of the pattern, the bivariate mark correlation function $k_{m1m2}(r)$ shows that the product of the first mark m_{i1} of the first point i and the second mark m_{j2} of the second point j is larger than expected by the null model. Nearby points have marks m_{i1} and m_{j2} that are more similar than expected (mark variogram) and the values of m_{i1} and m_{j2} are positively correlated (Schlather's I)

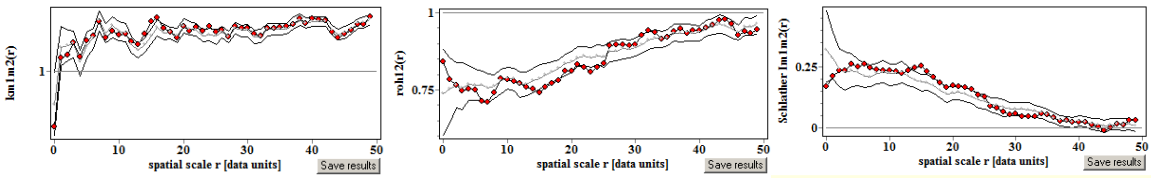
Local independent marking for univariate patterns with two marks

The pattern DataType7.mcf was generated without simulating spatial interactions between the two marks and the spatial correlations in the two marks were only imposed by the intensity function. To show this we can use the null model variant that conducts the randomization of the marks only locally.

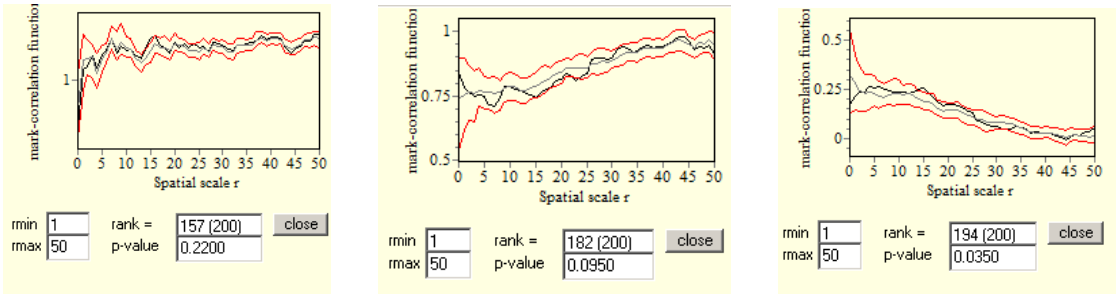
1. Select "**Mark correlation functions**" in window What do you want to do?
2. Highlight data file you want to analyze in Input data file. In the example it is file "DataType7.mcf".
3. Click "**List with coordinates, no grid**" in MCFunction
4. Provide in the window mark correlation functions the bin width in data units, an appropriate ring width, and a maximal distance *r* of the analysis. Select in the example a bin of 1m, and a ring width of 3m.
5. Check "Calculate simulation envelopes" in window What do you want to do? The subwindow "Select a null model" appears.
6. Select for the null model "**Marks pat 1 fixed and 2 random**" that shuffles the first mark randomly over the points. (You can also use the alternative null model "**Marks pat 2 fixed and 1 random**"). Select also the number of simulations of the null model (199) and the rule for the simulation envelopes (5' lowest and highest).
7. To approximately factor out the large-scale heterogeneity in the marks click the option "**Local independent marking**" and select an appropriate maximal distance *R* for points that should switch their marks (select here *R* = 20).



8. Press button "**Calculate Index**" and *Programita* shows the observed and simulated pattern. The red circles represent the first mark and the green circles the second mark, and the area of the disk is proportionally to the mark.
9. The mark correlation functions are now well within the simulation envelopes:



although the GoF test shows a weak departure for the Schlather's I:

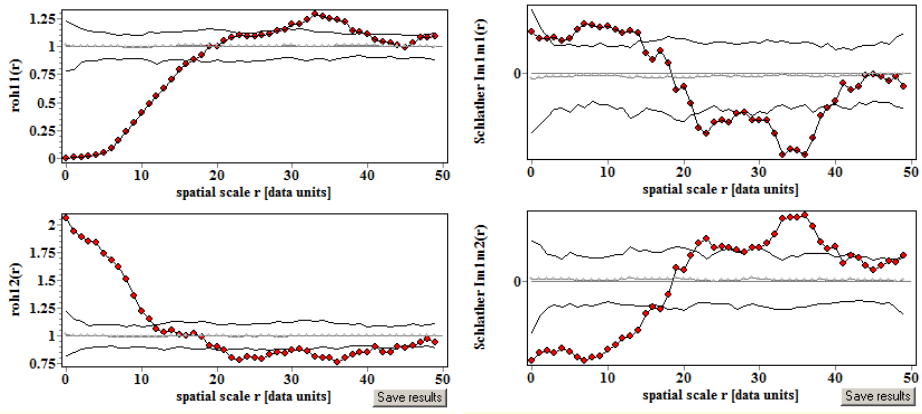


Independent marking of mark vector for univariate patterns with two marks

If the two marks of a point constitute a unit such as seeds of different species that were dispersed together in the same feces or if the marks characterize two different properties of the point (e.g., height and dbh) the randomization of the null model needs to shuffle the entire vector of marks (m_{i1}, m_{i2}) over the points i of the pattern. This is also required if one of the marks is a "constructed" mark, i.e., a mark that was calculated based on some neighborhood property of the pattern.

In the example Book_Fig2_16_bi.res we used such a constructed mark as second mark which was the number of points within distance of 10m around the point.

1. Select "**Mark correlation functions**" in window What do you want to do?
2. Highlight data file you want to analyze in Input data file. In the example it is file "Book_Fig2_16_bi.mcf".
3. Click "**List with coordinates, no grid**" in MCFunction
4. Provide in the window mark correlation functions the bin width in data units, an appropriate ring width, and a maximal distance *r* of the analysis. Select in the example a bin of 1m, and a ring width of 5m.
5. Check "Calculate simulation envelopes" in window What do you want to do? The subwindow "Select a null model" appears.
6. Select for the null model "**Marks pat 1 and 2 random together**" that shuffles the vector (m_{i1}, m_{i2}) of marks randomly over the points, the number of simulations of the null model (199), and the rule for the simulation envelopes (5' lowest and highest).
7. Press button "**Calculate Index**" and *Programita* shows the observed and simulated patterns. The red circles represent the first mark and the green circles the second mark, and the area of the disk is proportional to the mark.
8. The bivariate mark variogram shows that the two marks of nearby points are up to distances of 10m more different than expected (and between 20 and 45m more similar than expected) and Schlather's I indicates for distances up to 15m a negative correlation between the two marks of nearby points (and between 23 and 40m a positive correlation):

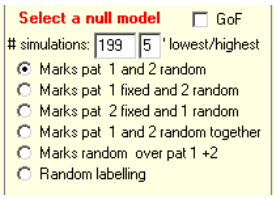


Clearly, this is caused by the construction of the first mark which is proportional to the inverse of the number of points within 10m.

6.1.8 Analysis of pattern with one qualitative and one quantitative mark (data type 8)

Programita allows also analyzing a data type where each point contains one qualitative and one quantitative mark. For example, the quantitative mark could be the size of a tree and the qualitative mark indicates whether the tree survived or died during the last 5 years.

The basic interest in analyzing patterns of this type can be twofold, first we may be interested to find out whether the quantitative mark (e.g., size) shows a spatial correlation with the qualitatively marked points (i.e., surviving and dead), or we may be interested to find out whether the qualitative mark (i.e., surviving and dead) shows a spatial correlation with the quantitative mark (i.e., size). *Programita* thus offers two contrasting null models, one that randomizes only the quantitative mark and one that only randomized the qualitative mark.



Null model type 1
For example, we may expect that dead trees located near surviving trees would be smaller than expected given the overall sizes of surviving and dead trees. This question requires a null model where the qualitative marks (e.g., surviving and dead) are fixed, but the size of the quantitative mark is randomly shuffled over all points (e.g., surviving and dead trees) (**Marks random over pat 1 + 2**).

Null model type 2
Alternatively, we may randomly shuffle the qualitative marks (e.g., surviving vs. dead) over all locations and retain the quantitative mark (e.g., size) as fixed **(Random labeling**).

The decision between the two null models depends on the data and the ecological question on hand. Details on this data type are provided in see section 3.1.7.4 “One Qualitative and One Quantitative Mark (Data Type 8)” in Wiegand and Moloney (2014).

6.1.9 Data preparation for data type 8

The data files must be an ASCII file with the *.mcf extension with the following format (the example are the first lines of file DataType8.mcf):

```
0   500   0   500   600
0.60    35.35   1   1.281   1.281
0.70   274.90   1   1.976   1.976
1.80   274.60   2   2.138   2.138
1.15   342.20   2   3.072   3.072
1.10   385.85   1   1.361   1.361
…
```

where the first line gives the size of the observation window (500 × 500 units in the example) and the number of points in the pattern.

- the first two columns of the following lines are the coordinates of the points. The third column codes the qualitative mark; here a “1” for dead and a “2” for surviving.
- the fourth and fifth column carry the mark of the type 1 and 2 point, respectively (you can write the mark of the points in both columns).

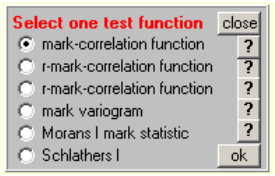
6.1.10 Steps for data type 8 (DataType8.res)

Programita estimates for the univariate pattern of type 1 points the univariate mark correlation functions and the corresponding bivariate test statistics considering the marks of the two types of points. However, because the null model randomizes the marks over the entire pattern the results of the univariate analysis do not coincide with the results of the univariate analysis of the type 1 points.

To understand the bivariate mark correlation functions of data type 8 it is important to note that the bivariate mark correlation function for this data type estimates the mean value $c_t(r)$ of a test function $t(m_i, m_j)$ taken over all pairs of points i and j that are located at distance r and where the first point i is of type 1 and the second point j of type 2. This conditional mean is then normalized with the mean c_t taken over all pairs of points where the first point i is of type 1 and the second point j of type 2.

The bivariate methods are explained in detail in section 3.1.7.4 "One Qualitative and One Quantitative Mark (Data Type 8)" in Wiegand and Moloney (2014). You can select the following "uni" and bivariate mark correlation functions:

- the mark correlation function $k_{mm}(r)$
- the r-mark correlation function $k_{m\,.}(r)$
- the r-mark correlation function $k_{.\,m}(r)$
- the mark variogram $\gamma_{mm}(r)$
- a Moran's I type mark statistics $I_{mm}(r)$
- Schlather's correlation function $I_{mm}(r)$

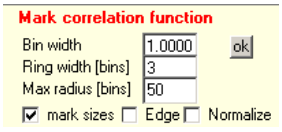


Because the two null models randomize over all points (either shuffling the qualitative or the quantitative mark over all points), both the "univariate" and the bivariate mark correlation functions are of interest here. For example,

- the "univariate" r-mark correlation function $k_{m\,.}(r)$ estimates the mean size of a dead tree (type 1) that has another dead tree (type 1) at distance r whereas the
- the "bivariate" r-mark correlation function $k_{m\,.}(r)$ estimates the mean size of a dead tree (type 1) that has a surviving tree (type 2) at distance r.

- the "univariate" mark variogram $\gamma_{mm}(r)$ estimates the mean squared size difference between a dead tree (type 1) and another dead tree (type 1) at distance r whereas the
- the "bivariate" mark variogram $\gamma_{mm}(r)$ estimates the mean squared size difference between a dead tree (type 1) and surviving tree (type 2) at distance r

The default for mark correlation functions are the normalized functions, i.e., $k_t(r) = c_t(r)/c_t$ where c_t is the normalization constant for a given test function t. However, you can also use the non-normalized mark correlation functions $c_t(r)$ when disabling the checkbox "**Normalize**".

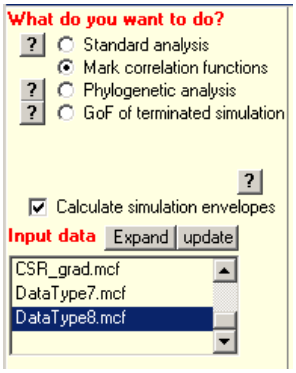


Random labeling for patterns with one qualitative and one quantitative mark

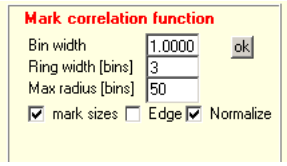
The mark correlation mode for a pattern with one qualitative and one quantitative mark can be accessed with the following sequence of actions:

1. Select "**Mark correlation functions**" in window What do you want to do?
2. Highlight data file you want to analyze in Input data file. In the example it is file "DataType8.mcf".

   This data file was generated by using the data file Book_Fig2_16a.mcf and randomly assigning to 213 of the 600 points the type 2 and to the other 378 points the type 1. Thus, the null model "**Random labeling**" should not yield significant departures in none of the mark correlation functions.

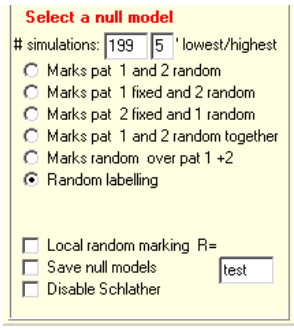


3. Click "**List with coordinates, no grid**" in MCFunction
4. Provide in the window mark correlation functions the bin width in data units, an appropriate ring width, and a maximal distance *r* of the analysis. Select in the example a bin of 1m, and a ring width of 3m.
5. If you use a bin of 1 unit, you can later use the function "**Combine replicates**" to load the results of the analysis, to change the bin, and to use the corresponding cumulative summary statistic (see "View results of mark correlation analysis").
6. Disable "**Normalize**" if you want to use the non-normalized mark correlation functions. The default is "Normalize"
7. Check "**Edge**" if you want to use the Ripley edge correction. Default is no edge correction. Note that edge correction is not required for mark correlation functions.

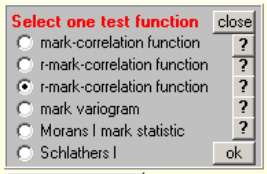


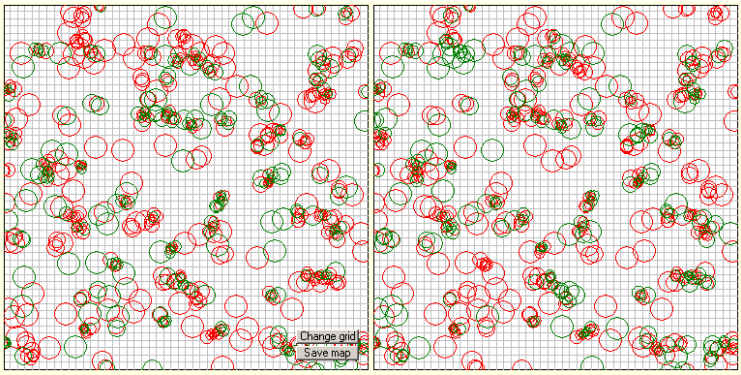
8. Check "**Calculate simulation envelopes**" in window What do you want to do? The subwindow "Select a null model" appears.
9. Select the number of simulations of the null model (199), and the rule for the simulation envelopes (5' lowest and highest). Note that not all null models are appropriate for all mark correlation function data types and that they must be selected with care.

   Select the null model "**Random labeling**" that shuffles the qualitative mark randomly over the points.

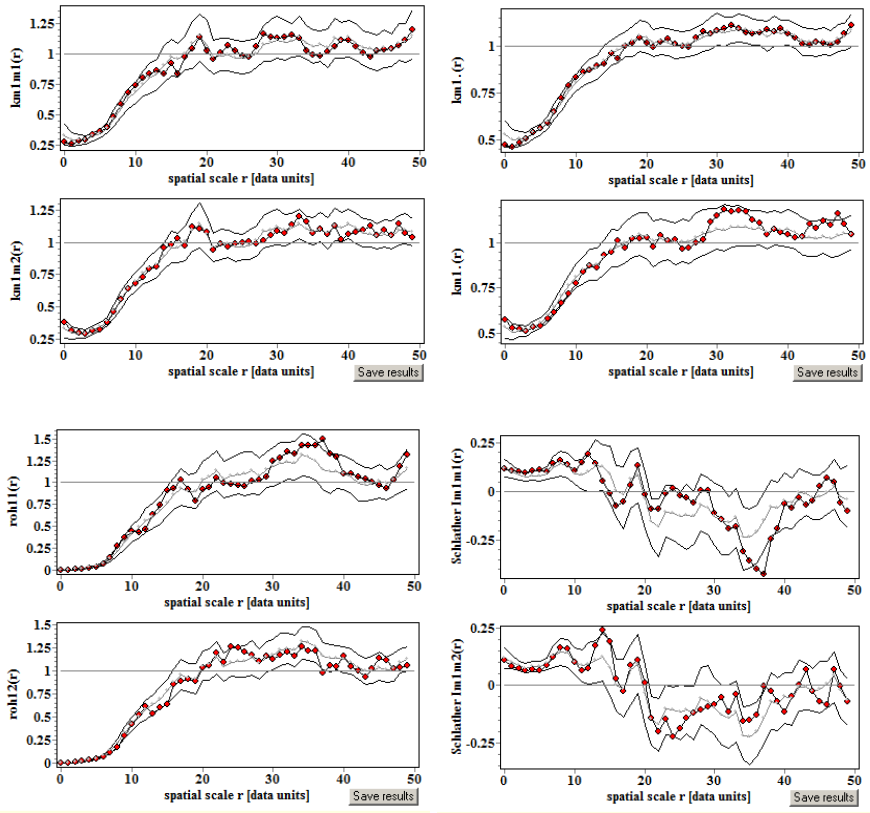


10. Press button "**Calculate Index**" and *Programita* shows the observed and simulated pattern. The red disks are type 1 points and the green disks are type 2 points, and the area of a disk is proportional to the mark. Note that the null model changes only the type (indicated by changing the red and green color), but not the location or size of the marks:





11. Use the radio buttons of the window **Select one test function** to select a mark correlation function and click the small "**ok**" button to get the result graphic:
12. As expected, all mark correlation functions are within the simulation envelopes of the random labeling null model:



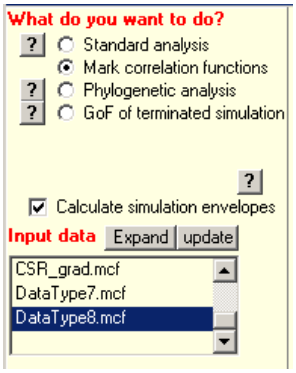
13. Note that the null model of random labeling yields, as for qualitatively marked patterns, as expectation the univariate mark correlation functions of the joined pattern of type 1 and type 2 points which clearly differs here from the expectation of independent marking where the values of the marks are randomly shuffled.

14. The random labeling null model can also be applied with local random marking.

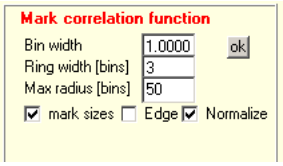
Independent marking for patterns with one qualitative and one quantitative mark.
We now apply the null model of independent marking (**Marks random over pat 1 + 2**) to the previous example.

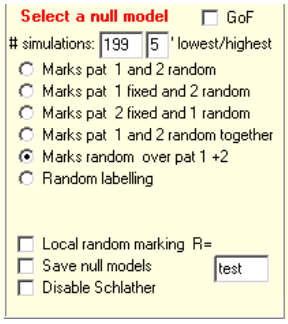
1. Select "**Mark correlation functions**" in window What do you want to do?
2. Highlight data file you want to analyze in Input data file. In the example it is file "DataType8.mcf".

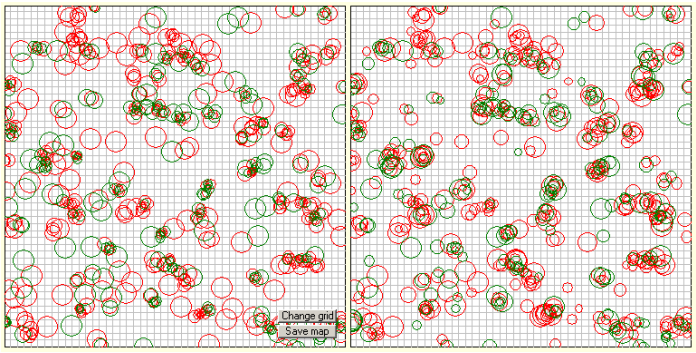
   This data file was generated by using the data file Book_Fig2_16a.mcf and randomly assigning to 213 of the 600 points the type 2 and to the other 378 points the type 1. Thus, the null model "**Marks random over pat 1 + 2**" should yield significant departures if the quantitative marks show a spatial structure (as the case in Book_Fig2_16a.mcf).



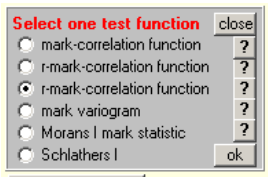
3. Click "**List with coordinates, no grid**" in MCFunction
4. Provide in the window mark correlation functions the bin width in data units (1), an appropriate ring width (3), and a maximal distance *r* of the analysis (50).
5. Check "**Calculate simulation envelopes**" in window What do you want to do? The subwindow "Select a null model" appears.
6. Select the null model "**Marks random over pat 1 + 2**" that shuffles the quantitative mark randomly over the points, the number of simulations of the null model (199), and the rule for the simulation envelopes (5' lowest and highest).
7. Press button "**Calculate Index**" and *Programita* shows the observed and simulated patterns. The red disks are type 1 points and the green disks are type 2 points, and the area of a disk is proportional to the mark. Note that the null model changes only the size, but not the location or type of the points (i.e., the color):



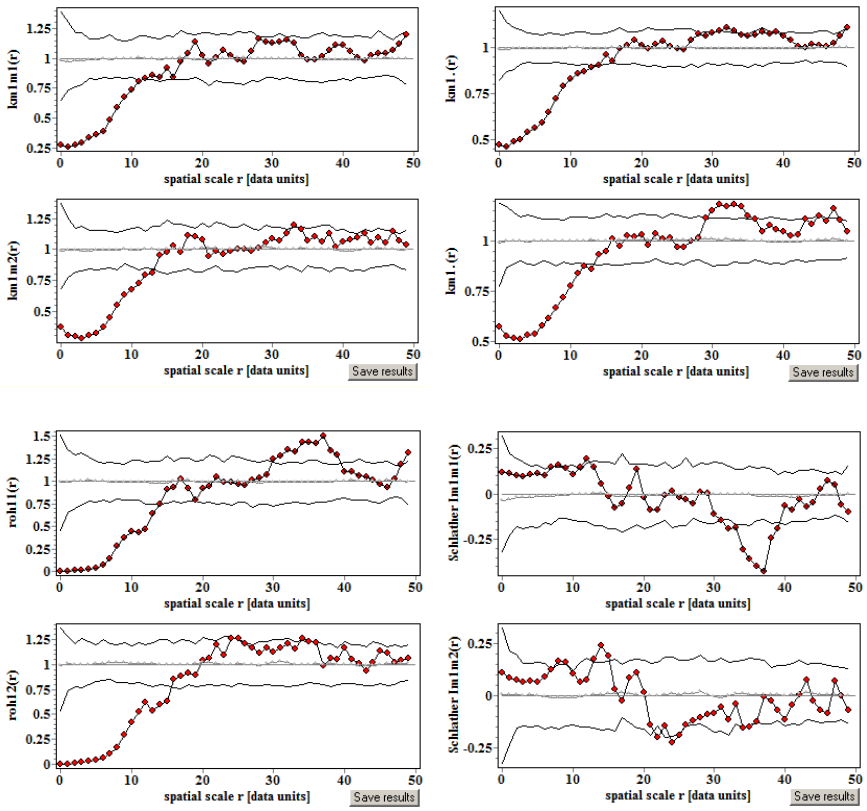




8. Use the radio buttons of the window Select one test function to select a mark correlation function and click the small "**ok**" button to get the result graphic:



9. As expected, the mark correlation functions show departures from the independent marking null model that were caused by the small-scale correlations in the quantitative marks:



10. The independent marking null model can also be applied with local independent marking.

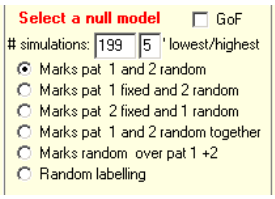
The two examples show that the selection between random labeling and independent marking null models depends on the underlying hypothesis and that the expectations of the two null models may greatly vary.

6.1.11 Analysis of a bivariate pattern with one quantitative mark (data type 9)

Programita allows also analyzing a data type where a bivariate pattern carries one quantitative mark. For example, we may have two different tree species and with the quantitative mark size of the trees.

The basic interest in analyzing patterns of this type is to explore the impact of proximity (and mark) of the first pattern (i.e., pattern 1) on the marking of the second pattern (i.e., pattern 2). The mark correlation functions for data type 9 are the same as for data type 8; however, the null models are fundamentally different. While the two alternative null models of data type 8 randomized the (qualitative or quantitative) marks over all points of the underlying univariate pattern, the null models of data type 9 must randomize the quantitative mark only within patterns. For example, we may keep the marks of the "antecedent" pattern 1 fixed and randomize only the marks of the second pattern. Because of the different randomization, the normalization constants c_t differ between data types 8 and 9.

You can use three null models, one that randomizes the quantitative marks of both component patterns (**Marks 1 and 2 random**), one that randomizes only the quantitative marks of pattern 2 (**Marks 1 fixed and 2 random**), and one that randomizes only the quantitative marks of pattern 1 (**Marks 2 fixed and 1 random**).



6.1.12 Data preparation for data type 9

The data files must be an ASCII file with the *.mcf extension with the following format (the same as for data type 8):

```
0   500   0   500   600
0.60    35.35   1   1.281   1.281
0.70   274.90   1   1.976   1.976
1.80   274.60   2   2.138   2.138
1.15   342.20   2   3.072   3.072
1.10   385.85   1   1.361   1.361
…
```

where the first line gives the size of the observation window (500 × 500 units in the example) and the number of points in the pattern.

- the first two columns of the following lines are the coordinates of the points. The third column codes the component pattern of the underlying bivariate pattern; here a "1" for the focal pattern 1 and a "2" for the second pattern 2.
- the fourth and fifth column carry the marks of the pattern 1 and 2 points, respectively (you can write the mark of the points in both columns).

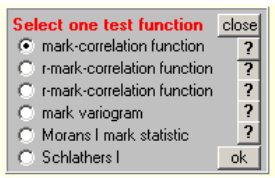
6.1.13 Steps for data type 9 (DataType9.res)

Programita estimates for the univariate pattern of type 1 points the univariate mark correlation functions described under data type 6, and additionally the corresponding bivariate test statistics considering the marks of the two types of points. Because the null models randomize only within the component patterns, the univariate analysis is identical with the univariate analysis of the first component pattern.

To understand the bivariate mark correlation functions of data type 9 it is important to note that the bivariate mark correlation function for this data type estimates the mean value $c_t(r)$ of a test function $t(m_{i1}, m_{j2})$ taken over all pairs of points i1 of pattern 1 and j2 of pattern 2 that are located at distance r. The $c_t(r)$ is then normalized with the mean c_t taken over all pairs of points where the first point i1 is of pattern 1 and the second point j2 of pattern 2.

The bivariate methods are explained in detail in section 3.1.7.5 "Bivariate Pattern with One Quantitative Mark (Data Type 9)" in Wiegand and Moloney (2014). You can select the following "uni" and bivariate mark correlation functions:

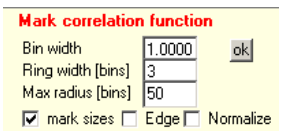
- the mark correlation function $k_{m1m2}(r)$
- the r-mark correlation function $k_{m1\,.}(r)$
- the r-mark correlation function $k_{.\,m2}(r)$
- the mark variogram $\gamma_{m1m2}(r)$
- a Moran's I type mark statistics $I_{m1m2}(r)$
- Schlather's correlation function $I_{m1m2}(r)$



Because the null models randomize within patterns, only the bivariate mark correlation functions are of interest here. For example,

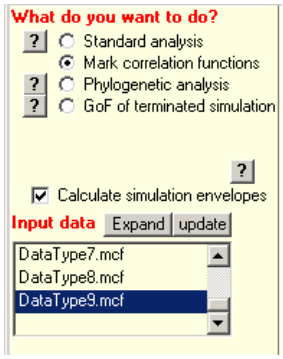
- the bivariate *r*-mark correlation function $k_{.\,m2}(r)$ estimates the mean size of a tree of species 2 that has a focal tree of species 1 at distance r. This is the mark correlation analogue to trivariate random labeling because it investigates the impact of the presence of focal species 1 on the marking of trees of species 2 that are located at distance r of a focal tree.
- the bivariate mark variogram $\gamma_{m1m2}(r)$ estimates the mean squared size difference between a focal tree of species 1 and a tree of species 2 located at distance r. If the mark is size, this allows for an assessment of whether a large focal tree of species 1 causes the size of trees of the second species to be smaller if they are closer to the focal trees.
- the bivariate correlation function $I_{m1m2}(r)$ estimates the correlation between the sizes of the trees of pattern 1 and that of trees of pattern 2 that are separated by distance r. The $I_{m1m2}(r)$ therefore investigates the impact of the mark of the focal species 1 on the marking of nearby trees of species 2.

The default for mark correlation functions are the normalized functions, i.e., $k_t(r) = c_t(r)/c_t$ where c_t is the normalization constant for a given test function t. However, you can also use the non-normalized mark correlation functions $c_t(r)$ when disabling the checkbox "**Normalize**".



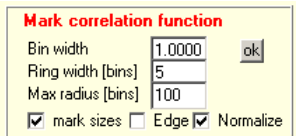
Independent marking for a bivariate pattern with one quantitative mark.

The mark correlation mode for a pattern with one qualitative and one quantitative mark can be accessed with the following sequence of actions:



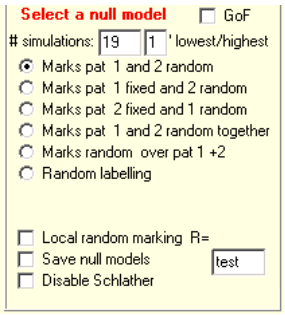
1. Select "**Mark correlation functions**" in window What do you want to do?
2. Highlight data file you want to analyze in Input data file. In the example it is file "DataType9.mcf".

   This data file was generated by generating 600 random points (i.e., CSR), and then randomly selecting 200 points to be of the focal pattern 1. The mark m_1 of pattern 1 was then determined as $m_1 = 2 - \lambda_1 K(r = 25)/2.51$and ranges between 0.01 and 2. Thus, the mark of pattern 1 was smaller if the point had more neighbors within 25m. The mark m_2 of pattern 2 was then determined as $m_2 = 2 - \lambda_1 K_{21}(r = 20)/2.51$and ranges between 0.01 and 2. Thus, the mark of pattern 2 was smaller if the point of pattern 2 had more neighbors of species 1 within 20m.

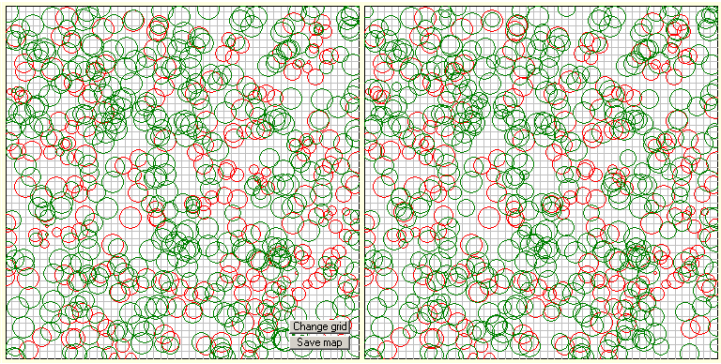


3. Click "**List with coordinates, no grid**" in MCFunction
4. Provide in the window mark correlation functions the bin width in data units (1), an appropriate ring width (5), and a maximal distance *r* of the analysis (100).
5. If you use a bin of 1 unit, you can later use the function "**Combine replicates**" to load the results of the analysis, to change the bin, and to use the corresponding cumulative summary statistic (see "View results of mark correlation analysis").
6. Disable "**Normalize**" if you want to use the non-normalized mark correlation functions. The default is "Normalize"
7. Check "**Edge**" if you want to use the Ripley edge correction. Default is no edge correction. Note that edge correction is not required for mark correlation functions.
8. Check "**Calculate simulation envelopes**" in window What do you want to do? The subwindow "Select a null model" appears.
9. Select an appropriate null model, the number of simulations of the null model (199), and the rule for the simulation envelopes (5' lowest and highest). Note that not all null models are appropriate for all mark correlation function data type and that they must be selected with care.

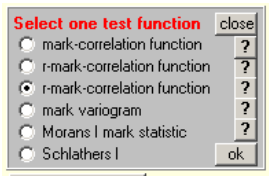
   Select the null model "**Marks pat 1 fixed and 2 random**" that shuffles the qualitative mark randomly over the points of pattern 2 but holds the marks of pattern 1 unchanged.



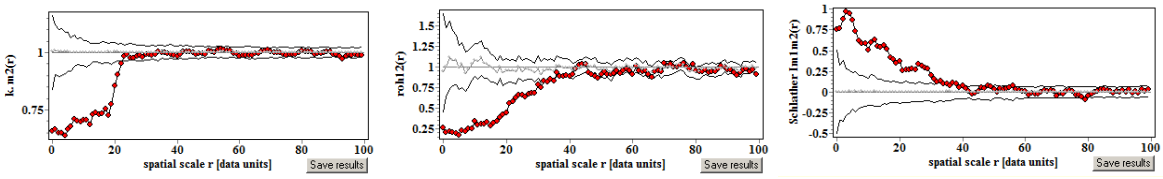
10. Press button "**Calculate Index**" and *Programita* shows the observed and simulated patterns. The red circles represent the mark of pattern 1 and the green circles the mark of pattern 2, and the area of the disk is proportional to the mark. Note that the null model changes only the size of pattern 2 (i.e., the green circles) but not that of pattern 1 (red circles):



11. Use the radio buttons of the window Select one test function to select a mark correlation function and click the small "**ok**" button to get the result graphic:

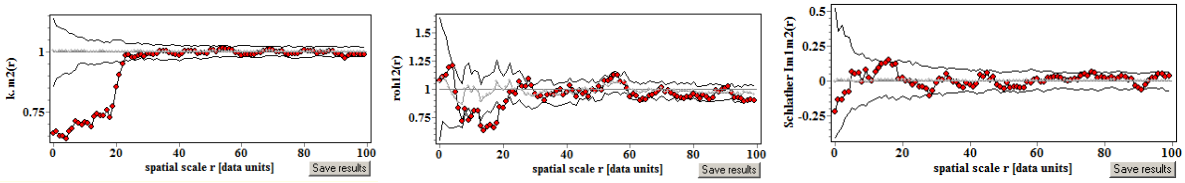


12. Of special interest here is the *r*-mark correlation function $k_{.m2}(r)$ that estimates the mean size of a point of pattern 2 at distance r of a point of pattern 1. As expected, it reveals that the mark of pattern 2 is smaller than expected if the points is within 20m of a point of pattern 1:



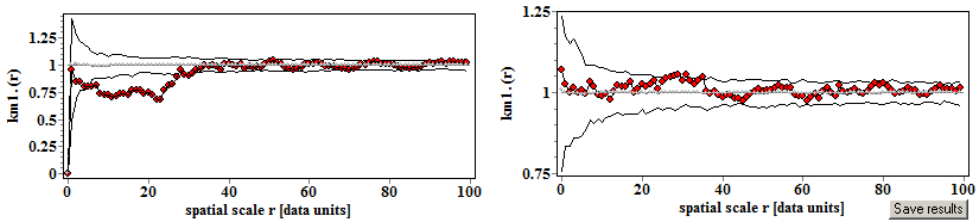
The mark variogram shows that the marks of the two patterns are more similar (i.e., smaller) if they are close together. This is because both, the marks of the points of pattern 1 and of pattern 2 were smaller if more points of pattern 1 were nearby. As a consequence of this, the marks of nearby points of the two patterns are positively correlated (Schlather's I).

13. Now we randomize the marks of pattern 1 to create the data file DataType9R1.mcf and analyze this pattern in the same way as above:

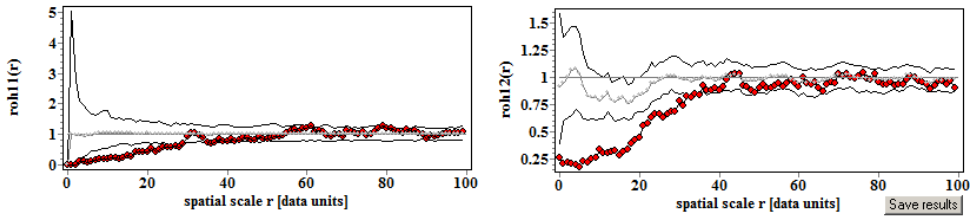


14. Of course, the result of the *r*-mark correlation function $k_{.m2}(r)$ does not change, but because the mark of pattern 1 is by construction independent on the number of neighbors of pattern 1, the strong correlation effects in the mark variogram and especially Schlather's I disappear.

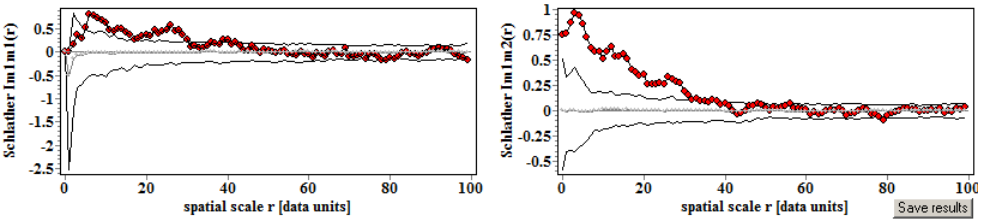
15. Using the alternative null model for pattern DataType9.mcf where only the mark of pattern 1 is randomized ("**Marks pat2 fixed and 1 random**") shows that points of pattern 1 which are located within distance 25m of another point of pattern 1 are smaller than expected but that the mark of pattern 1 is not influenced by presence of a point of pattern 2:



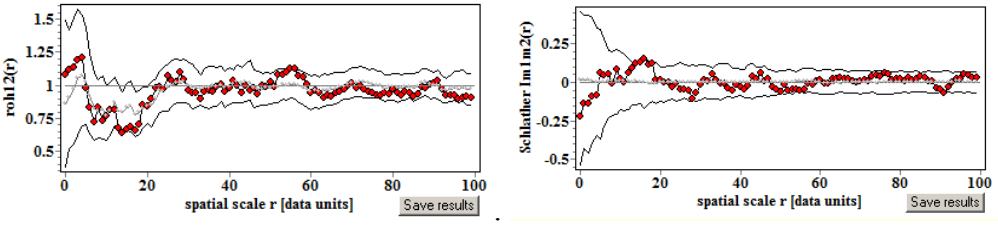
However, because both, the marks of the points of pattern 1 and that of pattern 2 depended on the number of neighbors of pattern 1, the mark variogram still depicts a significant bivariate effect:



The same is true for Schlather's I:



16. However, if we use the data set where the marks of pattern 1 were randomized (DataType9R1.mcf) and use null model **Marks pat2 fixed and 1 random** the latter two effects disappear because the spatial correlation between the marks of pattern 1 and pattern 2 was removed:



6.1.14 Trivariate random labeling with data type 9 (DataType9_triv.res)

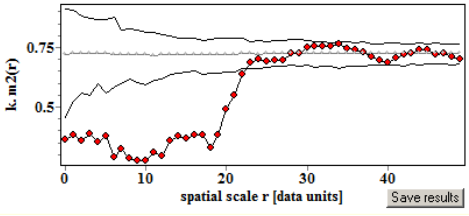
The r-mark correlation function $k_{.m2}(r)$ can be used together with the null model **Marks pat 1 fixed and 2 random** to conduct trivariate random labeling.

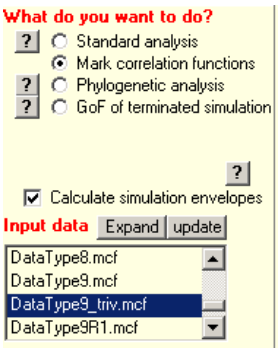
The test function $t(m_1, m_2)$ that corresponds to $k_{.m2}(r)$ yields the mark m_2 of the point of pattern 2. The $k_{.m2}(r)$ thus estimates the mean value of the mark of pattern 2 over all pairs of pattern 1 - pattern 2 points which are distance r apart. In our example above the mark was the size of a tree of species 2 which was smaller if the tree was closer to a tree of pattern 1.

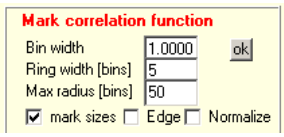
To implement within this framework trivariate random labeling we can use the mark zero if the tree was dead and the mark one if the tree was surviving. The $k_{.m2}(r)$ then estimates the mean value of the mark of pattern 2 over all pairs of pattern 1 - pattern 2 points which are distance r apart. This is the proportion of surviving trees of pattern 2 among all trees of species 2 that are located at distance r of a tree of species 1.

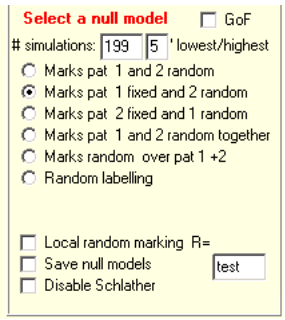
Trivariate random labeling, Example DataType9_triv.res
We use the data set DataType9.mcf, but the binary mark of the points of pattern 2 is defined as 0 if the mark of DataType9.mcf was below 1.3 and 1 otherwise.

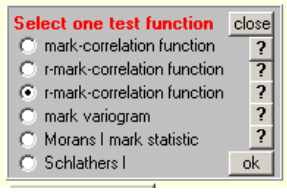
1. Select "**Mark correlation functions**" in window What do you want to do?
2. Highlight data file you want to analyze in Input data file. In the example it is file "DataType9_triv.mcf".
3. Click "**List with coordinates, no grid**" in MCFunction
4. Provide in the window mark correlation functions the bin width in data units (1), an appropriate ring width (5), and a maximal distance r of the analysis (50).
5. Check "**Calculate simulation envelopes**" in window What do you want to do? The subwindow "Select a null model" appears.
6. Select the null model "**Marks pat 1 fixed and 2 random**" that shuffles the mark of pattern 2 randomly over the points of pattern 2, the number of simulations of the null model (199), and the rule for the simulation envelopes (5' lowest and highest).
7. De-select "**Normalize**" to obtain a quantity with the interpretation of a mark connection function.
8. Press button "**Calculate Index**" and *Programita* simulates the null model.
9. As expected, the r-mark correlation function $k_{.m2}(r)$ shows that only 36% of all points of pattern 2 that are closer than 20m from a point of pattern 1 have mark 1 (surviving) as opposed by an average of 75% among all pattern 2 points:











7 Multivariate analysis of species richness

Coming soon

8 Multivariate analysis using a dissimilarity matrices

Coming soon

9 The grid-based standard analysis

Coming soon

10 Using *Programita* in the mode for object of finite size and real shape

Coming soon

11 References

Biganzoli, F., T. Wiegand and W.B. Batista. 2009. Fire-mediated interactions between shrubs in a South American temperate savannah. Oikos 118: 1383-1395

De la Cruz, M., R. L. Romao, A. Escudero, and F. T. Maestre. 2008. Where do seedlings go? A spatio-temporal analysis of seedling mortality in a semi-arid gypsophyte. Ecography 31: 1–11.

Diggle, P. J. 2003. *Statistical analysis of spatial point patterns*. 2nd ed. Arnold, London

Diggle P.J., V. Gómez-Rubio, P.E. Brown, A.G. Chetwynd, and S. Gooding. 2007. Second-order analysis of inhomogeneous spatial point processes using case-control data. Biometrics 63: 550-557.

Goreaud, F., and R. Pelissier. 2003. Avoiding misinterpretation of biotic interactions with the intertype *K*-12-function: population independence vs. random labelling hypotheses. Journal of Vegetation Science 14:681–692. 2010

Grimm, V. and S.F. Railsback. 2005. *Individual-based Modeling and Ecology*. Princeton University Press

Illian, J.B., J. Møller, and R. Waagepetersen. 2009. Hierarchical spatial point process analysis for a plant community with high biodiversity. Environmental and Ecological Statistics 16: 389–405

Jacquemyn, H., P. Endels, O. Honnay, and T. Wiegand. 2010. Evaluating management interventions in small populations of a perennial herb Primula vulgaris using spatio-temporal analyses of point patterns. *Journal of Applied Ecology* 47: 431–440

Loosmore, N. B., and E. D. Ford. 2006. Statistical inference using the *G* or *K* point pattern spatial statistics. Ecology 87:1925–1931.

Raventós, J., T. Wiegand, and M. De Luis. 2010. Evidence for the spatial segregation hypothesis: a test with nine-year survivorship data in a Mediterranean fire-prone shrubland show that interspecific and density-dependent spatial interactions dominate. Ecology 91:2110-2120

Stoyan, D., and H. Stoyan. 1996. Estimating pair correlation functions of planar cluster processes. Biometrical Journal 38: 259–271.

Wiegand T., and K. A. Moloney 2004. Rings, circles and null-models for point pattern analysis in ecology. *Oikos* 104: 209-229.

Wiegand, T, C.V.S. Gunatilleke, I.A.U.N. Gunatilleke, and T. Okuda. 2007. Analyzing the spatial structure of a Sri Lankan tree species with multiple scales of clustering. Ecology 88: 3088–3102.

Wiegand, T., A. Huth, S. Getzin, X. Wang, Z. Hao, C.V.S. Gunatilleke, and I.A.U.N. Gunatilleke. 2012. Testing the independent species arrangement assertion made by theories of stochastic geometry of biodiversity. Proceedings B 279: 3312-3320

Wiegand, T, F. He, and S.P. Hubbell. 2013. A systematic comparison of summary characteristics for quantifying point patterns in ecology. Ecography 36: 92-103.

Wiegand T., and K. A. Moloney 2014. A handbook of spatial point pattern analysis in ecology. Chapman and Hall/CRC press, Boca Raton, FL.