Poisson Point Processes

- For our purposes a spatial point process is a random set of event locations occurring in a nonrandom spatial domain, and we are interested in questions relating to the spatial pattern of those points.
- We are going to take a more traditional introductory approach, one that you should have seen before in a one dimensional setting and for which the extension to multiple dimensions will be hopefully more natural.

Homogeneous Poisson Processes

- We will let S be a set in a d-dimensional Euclidean space. Denote the family of all subsets of S by $\mathcal{A}$ and let $N(A)$ be the number of events occurring in some set (or region) A which is one of those subsets. Denote the *area* of A by $\nu(A)$. The following argument borrows heavily from Berry and Lindgren's introductory probability and math stat text and Cliff and Ord's spatial statistics text. We assume that

  1. The probability of a given number of events in any region A of size $\nu(A)$ does not depend on the location of that region.
  2. Events in nonoverlapping regions $A_1, A_2, \cdots$ are independent.
  3. The probability of the occurrence of a single event in a small region A is approximately proportional to $\nu(A)$:

     $$P\left(N(A)=1\right)=\lambda\nu(A)+o\left(\nu(A)\right).$$

     The notation $o\left(\nu(A)\right)$ is an unspecified remainder term that goes to 0 faster than $\nu(A)$ goes to 0, i.e.

     $$\lim_{\nu(A)\to 0}\frac{o\left(\nu(A)\right)}{\nu(A)}=0.$$

  4. The probability of two or more events in a small region A is approximately 0,

     $$P\left(N(A)\geq 2\right)=o\left(\nu(A)\right).$$

  We imagine dividing S into N small regions $A_1, A_2, \cdots, A_N$ each of area $\nu\left(A_i\right)=\nu(A)=\nu(S)/N$. The above assumptions imply that

  $$N\left(A_i\right)\dot{\sim}\text{Ber}\left(\lambda\nu(A)\right)$$

  and

  $$N(S)\dot{\sim}\text{Bin}\left(N,\lambda\nu(A)\right)$$

  where $\dot{\sim}$ means "approximately distributed as". Now let $N\to\infty$ while keeping $N\lambda\nu(A)=\lambda\nu(S)$ fixed. Then, in the limit,

  $$P\left(N(S)=x\right)=\frac{\exp\left(-\lambda\nu(S)\right)\left(\lambda\nu(S)\right)^x}{x!},x=0,1,\cdots,$$

To see this last step let $X \sim \text{Bin}(n, p)$. Let $np = m$ be fixed. Then

$$\begin{aligned} P(X = x) &= \binom{n}{x} p^x (1-p)^{n-x} \\ &= \frac{n(n-1)\cdots(n-x+1)}{x!} \left(\frac{m}{n}\right)^x (1-p)^{n-x} \\ &= \left(\frac{n}{n}\right)\left(\frac{n-1}{n}\right)\cdots\left(\frac{n-k+1}{n}\right)\left(\frac{m}{x!}\right)(1-p)^{-x}(1-p)^n \end{aligned}$$

The result follows from taking the limit as $n \to \infty$.

- The essence of a homogeneous spatial Poisson process is:
  1. If $N(A)$ denotes the number of events in a subregion A of S, then $N(A) \sim \text{Poi}(\lambda\nu(A))$, where the positive parameter λ is the mean number of events in a region of unit area (referred to as the *intensity* of the process).
  2. If $A_1, A_2, \cdots A_k$ are disjoint subregions in S then $N(A_i), i = 1, 2, \cdots, k$ are independent random variables, $N(A_i) \sim \text{Poi}(\lambda\nu(A_i))$.
- The Poisson process is often the starting point for the analysis of point patterns. This may seem strange because we do not expect point patterns to be random but we do it for 2 main reasons:
  1. The Poisson distribution is a natural null distribution when testing for spatially random patterns.
  2. Modifications of Poisson processes can produce nonrandom patterns. This is explored more fully in the latter part of Chapter 3.

Complete Spatial Randomness, Regularity, and Clustering

- The spatial pattern of complete spatial randomness (CSR) is synonymous with a homogeneous Poisson process.
- Let
  - A be a region and $N(A) = n \geq 1$ where $N(A) \sim \text{Poi}(\lambda\nu(A))$
  - $A_1, A_2, \cdots, A_m$ be a disjoint partition of A, i.e. $A_1 \cup A_2 \cup \cdots \cup A_m = A$
  - $k_1, k_2, \cdots, k_m$ are positive integers with $\sum_{i=1}^n k_i = n$.

  Then, conditional on $N(A) = n \geq 1$ event locations

$$P(N(A_1) = k_1, \cdots, N(A_m) = k_m | N(A) = n) = \frac{n!}{k_1! \cdots k_m!} \left(\frac{\nu(A_1)}{\nu(A)}\right)^{k_1} \cdots \left(\frac{\nu(A_m)}{\nu(A)}\right)^{k_m}.$$

  That is, conditional on the number of events being equal to n the distribution of points in subregions follows a multinomial distribution.

- Conditional on n events in a region A the distribution of the points over A follows a uniform distribution. This has implications for Monte Carlo based inference and for simulation studies of Poisson processes.

- There are a number of different methods for testing the null hypothesis of CSR in an observed point pattern. Tests can be based on asymptotic results or on Monte Carlo procedures. R has several spatial packages providing a number of different options.

  Monte Carlo Tests: The basic idea is that the observed data are a realization of a model Ψ. A test statistic Q is used to evaluate the pattern. Q will be computed not only for the observed pattern but for other patterns simulated under an assumption that Ψ holds. Generally, Ψ will be equivalent to an assumption of CSR. A total of g data sets are simulated and a value of Q is computed for each one. The observed value q_0 is combined with the g simulated values $q_1, q_2, \cdots, q_g$ and the $g+1$ values are ranked. The ranking of the observed value provides information on how unusual it is under the assumption of the model Ψ. Unusual will depend on the alternative hypothesis, clustering, regularity, or perhaps both and the particular Q chosen. We have already seen a couple of examples of this approach. Here is a more complete example in a non-spatial context.

  The basic idea is quite simple. We choose a statistic Q to measure how reasonable our data are under a null hypothesis of no pattern. The value of the statistic for the observed data is computed q_{obs}. The data are then randomly reordered a large number of times (s, say) and values of Q are computed for each reordered set of data, $q_1, q_2, \cdots, q_s$. If the null hypothesis of random arrangement is valid then the ordering of the observed data is just one of many equally likely orderings and the value of Q computed from the observed data should be typical with respect to the values generated from the randomly ordered data. If the null hypothesis is false q_{obs} should be unusual when compared to the distribution of the $q_i, i = 1, \cdots, s$. Viewed within the context of significance testing the P-value of the randomization test is the proportion of values of q_i that are as extreme or more extreme than q_{obs}. The procedure has 2 main advantages: first it can be carried out on nonrandomly selected data and second no distributional assumptions (e.g. normality) are required. The main disadvantage is that, with nonrandom sampling, inference can only be drawn to the particular sample and not necessarily to some underlying population of interest.

  This may seem like a major disadvantage but we often work with nonrandom samples. Almost all medical experiments comparing different treatments are carried out on convenience samples of patients who have consented to take part in a study, hardly a random sample from the underlying population of interest. Many observational studies also suffer from the failure of this assumption.

  Example 1: This example is taken from Bryan Manley's text: Randomization, Bootstrap and Monte Carlo Methods in Biology. *It is nonspatial but simple. The data*

are 20 observations on mandible length in 20 golden jackals, 10 male and 10 female. Do mandible lengths in males differ on average from mandible lengths in females? Measurements were made on museum specimens, hardly a random sample. The data are given below.

Males:	*120*	*107*	*110*	*116*	*114*	*111*	*113*	*117*	*114*	*112*
Females:	*110*	*111*	*107*	*108*	*110*	*105*	*107*	*106*	*111*	*111*

This might be the type of example you would see in an introductory statistics text in the chapter on pooled 2-sample t-tests. The assumptions of this test are simple random samples from the 2 independent populations of interest, equal population variances, and normally distributed mandible lengths within each population. We do not have simple random samples from male and female golden jackal populations. The sample sizes are too small to allow for any rigorous assessment of the validity of the other 2 assumptions although we know that the t-procedures are fairly robust to violations of those, especially with equal sample sizes. However, the t procedures are most definitely not robust to violations of the simple random sampling assumption. Recall that the test statistic is

$$T = \frac{\left(\overline{X}_m - \overline{X}_f\right)}{s\sqrt{1/n_m + 1/n_f}}$$

with the pooled standard deviation given by

$$s = \sqrt{\frac{\left[\left(n_m - 1\right) s_m^2 + \left(n_f - 1\right) s_f^2\right]}{\left(n_m + n_f - 2\right)}}.$$

If the null hypothesis of no difference is true and if all assumptions are met then the test statistic follows a t-distribution with 18 degrees of freedom. The two-sided t-test (assuming equal variances) yields

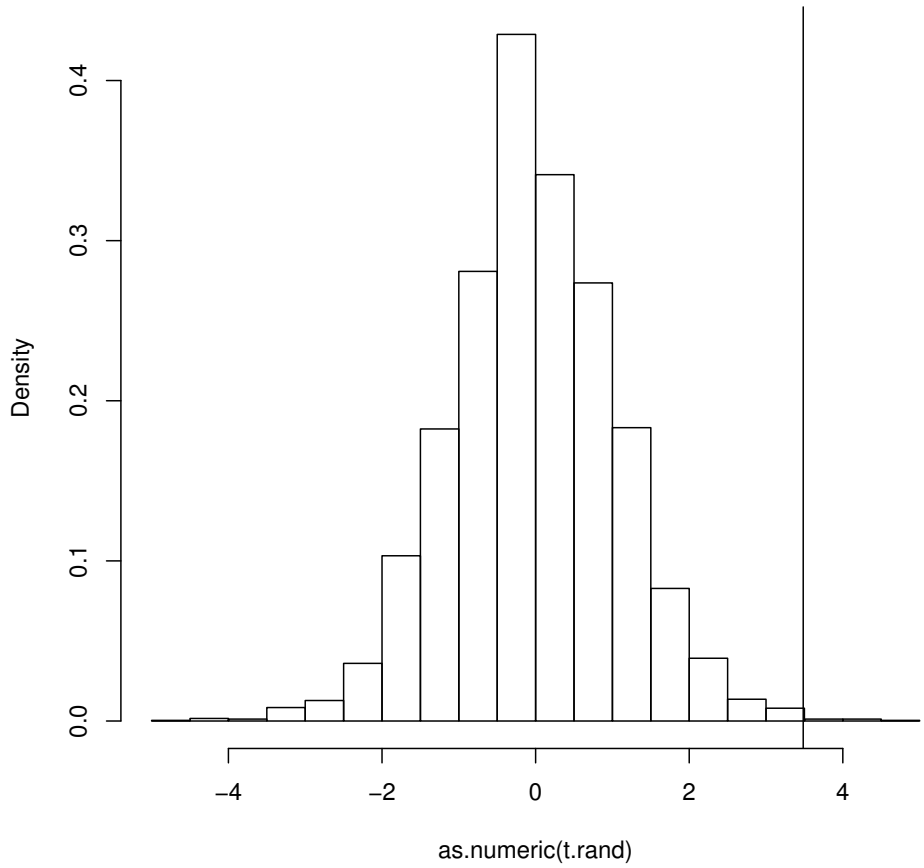
```
t.test(male,female,var.equal=T)

        Two Sample t-test

data:  male and female
t = 3.4843, df = 18, p-value = 0.002647
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 1.905773 7.694227
sample estimates:
mean of x mean of y
    113.4     108.6
```

The P-value of 0.00265 is low enough to constitute strong evidence against the null hypothesis.

The randomization procedure is carried out as follows. After computing the observed value of the test statistic of $t = 3.4843$ *we randomly reallocate 10 of the mandible lengths to the "male" group and the other 10 to the "female" group. We compute the value of the test statistic for each such reallocation. We repeat this step a large number of times to generate an approximation to the randomization distribution of the test statistic. If the observed value appears to be a typical value from this distribution then we conclude that the data are consistent with the null hypothesis and if the observed value is atypical we conclude that the observed data are not consistent with the null hypothesis. This is easy to do in R.*

```
# set up a storage vector
t.rand<-rep(0,5000)
t.rand[1]<-3.4843
jackal<-c(male,female)
# set up a gender indicator variable: 1=male, 0=female
gvec<-c(rep(1,10),rep(0,10)
for(i in 2:5000){
gvec.rand<-sample(gvec,rep=F)
t.rand[i]<-t.test(jackal[gvec.rand==1],jackal[gvec.rand==0],var.equal=T)$statistic}
# t.rand is a list so we convert to numeric to get summaries
t.rand<-as.numeric(t.rand)
mean(t.rand)
[1] -0.01926697
hist(t.rand,prob=T,main="Histogram of randomized test statistics")
abline(v=3.4843)
# compute 2-sided P-value
sum(abs(t.rand)>=3.4843)/5000
[1] 0.0036
```

The histogram is shown below.



Clearly the observed value is unusual and we would be safe in concluding that the observed difference in this sample is not due to chance but indicative of a real difference. Note that the randomization distribution of the test statistic appears classically bell-shaped and the randomization P-value is very close to the P-value from the 2 sample t-test.

We would probably have been justified in carrying out a one-sided test with an alternative hypothesis of mean male length being greater than mean female length. The adjustment in the P-value would be accomplished in the usual way.

```
sum(t.rand)>=3.4843)/5000
[1] 0.0016
```

The above procedure of determining P-values is reasonable when the randomization distribution is (approximately) symmetric. A number of different suggestions

have been made as to how to proceed when the randomization distribution is not symmetric. One intuitively reasonable approach is to find the proportion of values lying at or below the observed value of the test statistic for lower-tailed alternatives or the proportion lying at or above the observed value for upper-tailed alternatives. For 2 sided alternatives it has been suggested that one report the minimum of $2P_L$ and $2P_U$ (with P_L (P_U) being the proportion below (above) the observed value). Others have suggested that the minimum of P_L and P_U be reported along with the direction in which the observed value is extreme.

We will see more spatial examples below.

- Goodness of Fit Tests based on Quadrat Counts: We can take several different approaches here but all are based on a comparison of observed quadrat counts to what would be expected under an assumption of CSR. Often we have an observed collection of points. the area is gridded off into quadrats, and the number of points in each quadrat is determined. In a very real sense point process data have been converted into regional data. We will illustrate the approaches using simulated data in Table 3.1 on page 92 in the text. The data are simulated counts in 25 quadrats arranged in a 5 by 5 grid each of unit area. Counts were simulated under assumptions of CSR, clustering, and regularity. The methods we will look at include

  1. A chi-squared test based on an index of dispersion

$$X^2 = \sum_{i=1}^{r}\sum_{j=1}^{c}\frac{(n_{ij}-\overline{n})^2}{\overline{n}} = (rc-1)S^2/\overline{n}$$

  where r and c are the number of rows and columns, respectively, n_{ij} is the count in quadrat ij, and $\overline{n} = n/(rc)$ is the number expected in each quadrat under CSR. The null distribution is approximately chi-squared with $rc-1$ degrees of freedom. Generally, chi-squared tests are considered to be one-tailed tests but this one is not. We should get small values of X^2 under regularity and large values under clustering.

  Example: We will look at some examples using the simulated data on page 92 of the text. The data are counts on 25 quadrats arranged in a 5 by 5 grid. The data are reproduced below (Table 1).

  The index of dispersion defined above is $I = S^2/\overline{n}$. Under an assumption of CSR the statistic

$$X^2 = (rc-1)S^2/\overline{n} = (rc-1)I$$

  will be asymptotically chi-squared distributed with $rc-1$ degrees of freedom where r is the number of rows and c is the number of columns in the grid.

```
CSR<-c(6,3,7,4,5,5,3,1,3,4,2,3,8,3,5,5,5,5,5,4,1,4,2,3,4)
CLS<-c(2,1,7,5,10,3,6,2,3,10,8,4,1,6,5,1,6,0,2,1,1,2,8,3,3)
```

```
REG<-c(7,5,3,2,6,3,2,4,5,7,4,5,3,5,3,2,4,7,3,3,4,4,4,2,3)
# Compute the values of X 2
Xsq.csr<-24*var(CSR)/mean(CSR)
Xsq.cls<-24*var(CLS)/mean(CLS)
Xsq.reg<-24*var(REG)/mean(REG)
c(Xsq.csr,Xsq.cls,Xsq.reg)
[1] 17.0 52.0 14.5
```

As indicated above we have to be a bit careful here with testing. Generally, we consider a chi-squared test to be a one-tailed test but if we took that approach here we would not necessarily be able to distinguish CSR from a regular or clustered pattern and in fact the standard approach might lead to a failure to reject a clearly nonrandom regular pattern. Note that under an assumption of CSR we expect the value of the test statistic to be around $rc - 1$.

If the data are clustered the variance should be large relative to the mean and the value of the test statistic should exceed $rc - 1$. *If the data are regularly distributed we should see less variability relative to the mean and the value of the test statistic should be less than* $rc - 1$. *Generally, you would have some a priori idea of which pattern to expect and that would often be clustering. A test of CSR against clustering involves a determination of whether or not the observed value is too large. For the CLS example the p-value would be calculated as*

```
1-pchisq(52,24)
0.0007823918
```

A p-value for a test of CSR against a regular pattern involves determination of whether or not the observed value is too low. For the third data set this is given by

```
pchisq(14.5,24)
0.06546004
```

Table 1. Complete Spatial Randomness

6	3	7	4	5
5	3	1	3	4
2	3	8	3	5
5	5	5	5	4
1	4	2	3	4

Table 2. Clustered Pattern

2	1	7	5	10
3	6	2	3	10
8	4	1	6	5
1	6	0	2	1
1	2	8	3	3

Table 3. Regular Pattern

7	5	3	2	6
3	2	4	5	7
4	5	3	5	3
2	4	7	3	3
4	4	4	2	3

Notice that you need to give some thought to which alternative hypothesis is tenable and also to what fail to reject (if you feel you must make such a decision) really means. For example, if you had mistakenly thought that the pattern in 2 *was produced by a spatial process producing regular patterns on average (which would be your alternative hypothesis) you would have computed the p-value to be*

```
pchisq(52,24)
0.9992176
```

which would obviously be interpreted as weak evidence against the null hypothesis. But you would not be justified in concluding that the observed pattern was the result of a CSR spatial process.

One could take a Monte Carlo approach to testing. We could do this in a number of different way. We will try the latter approach here. Our null hypothesis would be that the observed pattern had been produced by a CSR process, i.e. a homogeneous Poisson process with mean estimated by the mean of the observed data. In each of the cases above the sample mean is 4*. Thus, we will simulate data from a Poisson process with mean* 4 *and assess the reasonableness of CSR for each case above. I will show R code for only one of the Monte Carlo test procedures, the regular pattern.*

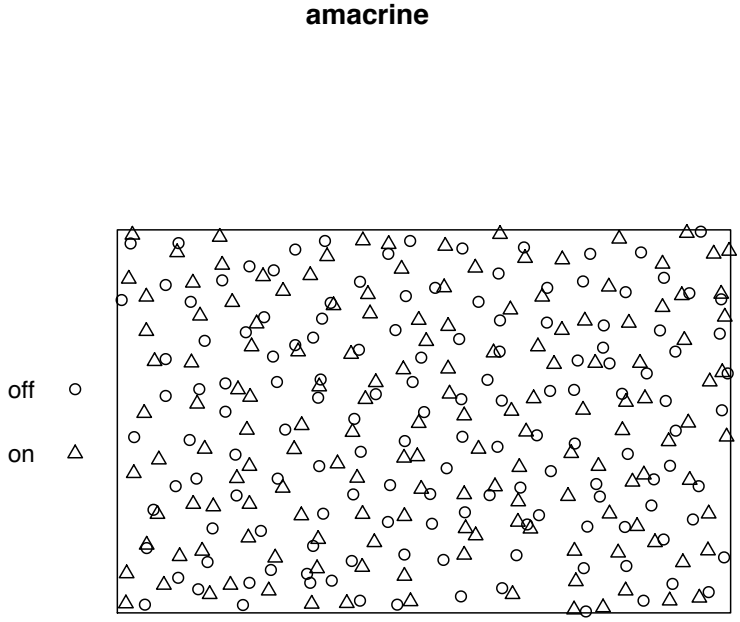
```
Itest<-rep(0,1000)
# Store the observed value of the test statistic in the
#first element of Itest.
Itest[1]<-24*var(REG)/mean(REG)
# Now generate 999 simulated data sets assuming a Poisson
# distribution with mean of 4.
for(i in 2:1000){
x.mc<-rpois(25,4)
Itest[i]=24*var(x.mc)/mean(x.mc)}
# Rank the values of the test statistic. We only need the
#rank of the observed value which is the first element of
#Itest.
rank(Itest)[1]
[1] 59.0
```

This would yield a p-value for a lower tail test of 0.059 *which is pretty close to the p-value produced by the chi-squared test above.*

Note that in this case we carried out the Monte Carlo test based on assuming the mean of 4 *which means that the number of events could vary from the total we saw in our actual data. We could have carried out a test conditional on the observed number of counts* 100*, also. We would have randomly distributed the* 100 *observations over the* 5×5 *grid and repeated the test above.*

This method is not ideal. It may not work well for data collected over a non-rectangular region. And, even if we have a rectangular region, changing the number of grid cells can have big impacts on the results. We can explore the impact of changing the number of quadrats using a function in the `spatstat` package.

Amacrine cell data: The data are available in the `spatstat` *package. They are spatial locations of cells in the retina of a rabbit. A plot is shown below.*



We will ignore the two different types of cells and just focus on the overall pattern. One might expect the pattern to be more regular than random. The locations are the centers of the cells and there is a natural spacing, i.e. a buffer surrounding each location approximately equal to the radius of the cell.

```
quadrat.test(amacrine)
```

```
        Chi-squared test of CSR using quadrat counts
        Pearson X2 statistic
```

```
data:  amacrine
X2 = 11.102, df = 24, p-value = 0.02347
alternative hypothesis: two.sided

Quadrats: 5 by 5 grid of tiles
```

We see some evidence against the null hypothesis of CSR but this is a two-sided test and we are not really interested in clustering as a possibility. We redo the analysis

```
quadrat.test(amacrine,alternative="regular")

        Chi-squared test of CSR using quadrat counts
        Pearson X2 statistic

data:  amacrine
X2 = 11.102, df = 24, p-value = 0.01173
alternative hypothesis: regular

Quadrats: 5 by 5 grid of tiles
```

We have fairly strong evidence against the null of CSR in favor of the alternative of a regular distribution.
Note that the function has, by default, gridded the rectangle into a 5×5 *grid. How sensitive are the results to that choice?*

```
quadrat.test(amacrine,alternative="regular",nx=6,ny=6)

        Chi-squared test of CSR using quadrat counts
        Pearson X2 statistic

data:  amacrine
X2 = 7.2245, df = 35, p-value = 1.287e-07
alternative hypothesis: regular

Quadrats: 6 by 6 grid of tiles
```

```
quadrat.test(amacrine,alternative="regular",nx=4,ny=4)

        Chi-squared test of CSR using quadrat counts
        Pearson X2 statistic
```

```
data:  amacrine
X2 = 2.7075, df = 15, p-value = 0.0002115
alternative hypothesis: regular

Quadrats: 4 by 4 grid of tiles
```

The result, good evidence against CSR, is the same in all 3 *cases but there is still quite a bit of variability in the p-values. It is not hard to imagine scenarios in which the results would be quite different depending on how one gridded an area.*

The function has the ability to carry out a MonteCarlo test.

```
quadrat.test(amacrine,alternative="regular",method="MonteCarlo")

        Conditional Monte Carlo test of CSR using quadrat counts
        Pearson X2 statistic

data:  amacrine
X2 = 11.102, p-value = 0.0125
alternative hypothesis: regular

Quadrats: 5 by 5 grid of tiles
```

The default is to simulate points assuming a Poisson process using the mean of the quadrat counts as the Poisson parameter. We could also carry out the conditional test (i.e. conditon on the observed number of points which is $n = 294$.

```
quadrat.test(amacrine,alternative="regular",method="MonteCarlo",
                              conditional=T)
       Conditional Monte Carlo test of CSR using quadrat counts
        Pearson X2 statistic

data:  amacrine
X2 = 11.102, p-value = 0.0105
alternative hypothesis: regular

Quadrats: 5 by 5 grid of tiles
```

Generally, whether or not you condition does not make that much difference. The Monte Carlo test is probably better - you do not have to meet the minimal expected cell count assumption for the chi-squared approximation to be valid.

2. One can also carry out a goodness-of-fit test of the null hypothesis of CSR (Poisson distributed quadrat counts) versus the alternative that the distribution is not that specified under the null. Let n be the number of quadrats. The counts are classified into k classes. Let O_i be the number of quadrats in class i. Let E_i be the number of quadrats expected in class i under an assumption of CSR. The test statistic is

$$X^2 = \sum_{i=1}^{k} \sum_{j=1}^{k} \frac{(O_i - E_i)^2}{E_i}$$

which is asymptotically chi-square distributed with $k - 2$ degrees of freedom under the null hypothesis. The expected frequencies are generated from a Poisson distribution with parameter λ estimated by the sample mean of the quadrat counts. A standard rule of thumb is that the expected counts should be 2 or more for each class and 5 or more for at least half of the classes.

Example: We can compare the observed distribution of counts to a distribution expected under the assumption of CSR. This is a large sample test requiring expected counts that exceed 5 *and that assumption is violated in all of the above examples so we will look at another example. The Atriplex data set also contained counts of plants in each quadrat. The mean number of plants in each quadrat is 0.297. The observed distribution and the distribution expected under a CSR assumption are shown in the table below.*

	0	1	≥ 2
Observed	191	54	11
Expected	190.24	56.48	9.28

The column labeled ≥ 2 *contains the number of quadrats containing* 2 *or more plants. For the observed data there were no observations of* 3 *or more plants but that cannot be ruled out if the data are truly Poisson. However, the expected number of quadrats containing* 3 *or more plants is only about* 0.9 *so pooling all expected counts of* 2 *or more and comparing to the observed counts of* 2 *or more certainly seems reasonable. The test statistic is*

$$X^2 = \sum_{i=1}^{3} \frac{(O_i - E_i)^2}{E_i}$$

where $k = 3$ *is the number of classes. Expected counts using*

```
256*dpois(0:6,0.297)
```

If the sample sizes are large enough (one commonly cited working rule of thumb is all expected counts ≥ 2 *and at least half of the expected counts* ≥ 5 *) then the test statistic will be approximately chi-squared distributed with degrees equal*

to $k - m - 1$ where m is the number of parameters that had to be estimated to carry out the test. In this case we only need to estimate the mean of the Poisson distribution so $m = 1$ and the degrees of freedom is equal to $3 - 2 = 1$. The value of the test statistic is $X^2 = 0.431$. The null and alternative hypotheses are

*Null: The population distribution is Poisson(*0.297*)*

*Alternative: The population distribution is not Poisson(*0.297*)*

We get a p-value of

```
1-pchisq(0.4307245,1)
[1] 0.5116457
```

which is certainly very weak evidence against the null hypothesis at best. However, in the spirit of actually looking at the data before blindly computing p-values a comparison of the observed and expected counts in the table above clearly indicate that the observed counts are consistent with those expected under a Poisson distribution.

Note: this test is always one sided. Low p-values would indicate the data are not consistent with the null but determination of clustering versus regularity would require examination of the data. For example clustering would be evident if there were more empty quadrats and quadrats with larger counts than expected under the null hypothesis. Cressie (1993; pages 590-591) discusses 6 simple indices developed for further exploration of such data. We will look at a handout of these in class.

Of course you could also carry out a Monte Carlo test using any of these test statistics.

3. Moran's I and Geary's c are also applicable in this setting.

   Example: The data are from a survey of red-cockaded woodpeckers in North Carolina. The study area is 4706×4706 *square feet in area divided into* $32 \times 32 = 1024$ *quadrats of equal size. The data are counts of woodpeckers in a quadrat over a specified period of time. I used the rook definition of spatial contiguity.*

```
moran.mc(woodpecker.dat1[,3],woodpecker.list,nsim=999)

        Monte-Carlo simulation of Moran's I

data:  woodpecker.dat1[, 3]
weights: woodpecker.list
number of simulations + 1: 1000

statistic = 0.6081, observed rank = 1000, p-value = 0.001
alternative hypothesis: greater
```

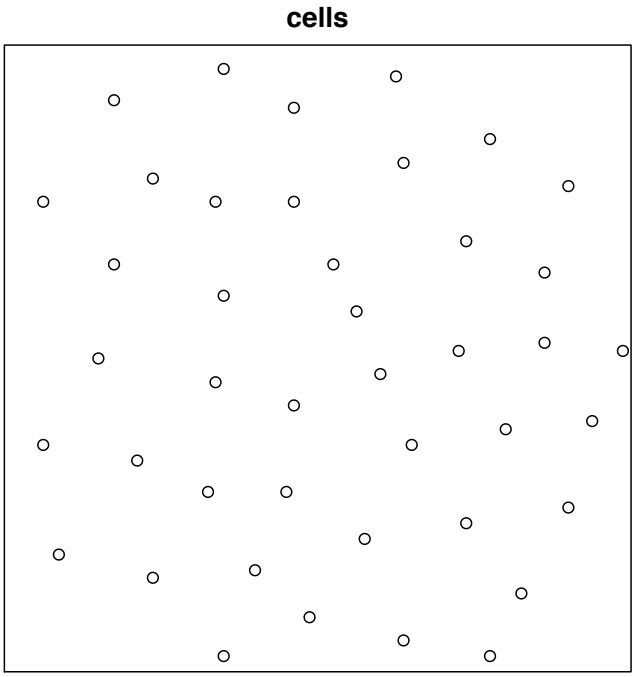
The default alternative `greater` *is a test for clustering. If we had suspected a regular distribution our p-value would be* 0.999*. If we had not had a clue and carried out a* 2 *sided test our p-value could be stated as* $2\min(0.999, 0.001) = 0.002$*.*

- As we have seen quadrat size can have an effect on the results of all the tests (and followup indices) we have discussed so far. Also, there is the fact that despite our textbook examples, we will not always be working with rectangular regions. Quadrat based methods are more of a pain on irregular regions. Tests based on distances are reasonable alternatives. These tests are generally conducted using Monte Carlo methods. Additionally, some distance measures actually estimate a function and the use of simulation envelopes will be presented and illustrated on these results.

  Nearest Neighbor Distances - We have a point pattern in a specified region. We determine the nearest neighbor distances between the event locations. If there are n locations there will be n nearest neighbor distances $h_i, i = 1, \cdots, n$. We can compute the mean of these distances $\overline{h}$ for the observed data. We can then carry out a Monte Carlo simulation of the process, compute the mean of the distances for each simulated pattern, and determine how unusual our observed value is relative to our expectation under an assumption of CSR. If the locations are clustered or clumped the mean should be smaller than expected under CSR. If the locations are regularly distributed then the observed mean should be larger. We can also look at the distribution of the simulated values.

  Example: The first data set is from the `spatstat` *package and is called* `cells`*. It is apparently a collection of spatial coordinates for cells on a* 1 *unit by* 1 *unit rectangle. The data are shown below.*

  ```
  data{cells)
  plot(cells)
  ```



The pattern looks pretty regular.

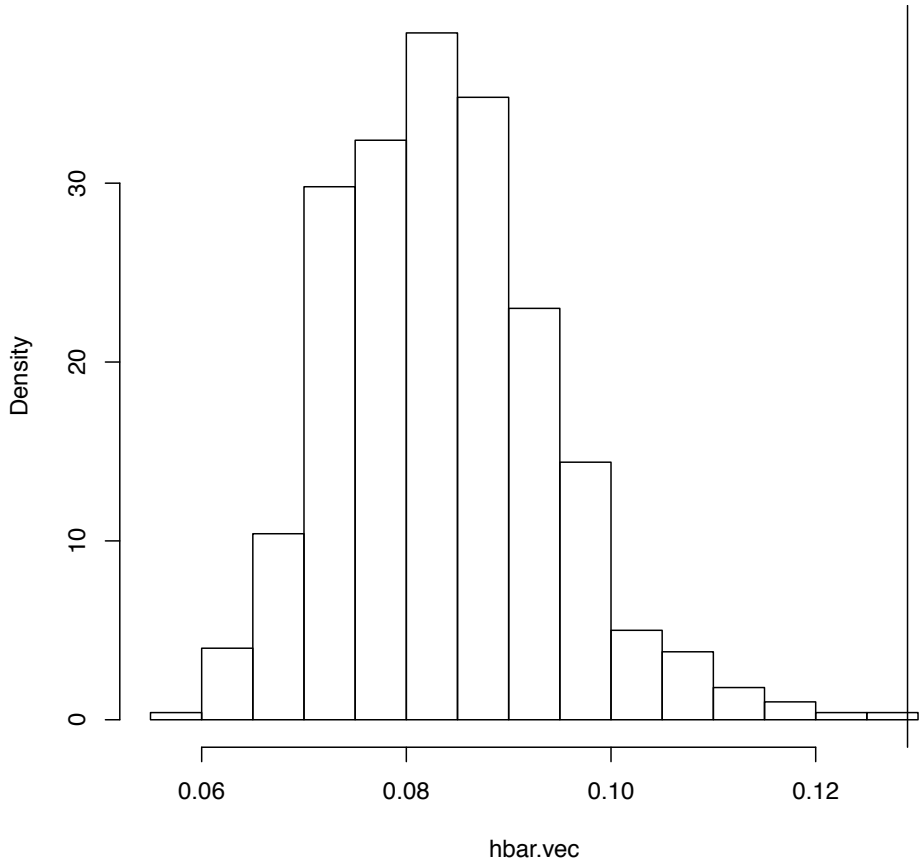
Here is the R code to carry out the analysis. The function `nndist` *in the R library* `spatstat` *will compute nearest neighbor distances. This function works most cleanly on a point process object. The function* `nndist.default` *will compute nearest neighbor distances for sets of spatial coordinates in other objects. We also need to simulate data from a 2 dimensional Poisson process. We use the function* `rpoispp` *for this purpose. This function will produce random event locations.*

```
# First we will compute the nearest neighbor distances for the 42
# observed event
# locations and their mean.
h<-nndist(cells) # nndist is a function in the R package spatstat
h
0.14583895 0.11486514 0.14024621 0.11180340 0.11180340 0.15449595
0.15449595 0.11486514 0.14577380 0.14243595 0.15206906 0.12356780
```

```
0.12356780 0.12500000 0.12356780 0.14313630 0.13036104 0.13036104
0.10697663 0.13861097 0.12224974 0.15206906 0.13861097 0.08363014
0.13036104 0.11200000 0.12224974 0.15089400 0.15037619 0.10662551
0.11819052 0.08363014 0.13462912 0.11200000 0.10662551 0.13852076
0.14313630 0.14313630 0.13953136 0.12801562 0.12801562 0.13852076
hbar<-mean(h)
hbar
[1] 0.1289729
# Now create a storage vector for the Monte Carlo results.
hbar.vec<-rep(0,1000)
hbar.vec[1]<-hbar
# Then simulate the data. We estimate the mean of the underlying
# Poisson distribution to
# be 42, which is the observed number of events in the unit square
# study area and this
# is easy to do with the cells data set.
for(i in 2:1000){
dat.pp<-rpoispp(42)
hbar.vec[i]<-mean(nndist(dat.pp))}
rank(hbar.vec)[1]
 999
```

A histogram of the results is shown below.

```
hist(hbar.vec,prob=T)
abline(v=mean(h))
```



The rank and location of the observed mean distance (vertical line in the plot) indicate that the observed mean is much too large to be consistent with a CSR pattern and is consistent with a regular distribution.

This is an example of a test assuming a Poisson model. We could have carried out a test conditioned on number of observations being fixed at 42*. Given that assumption the* 42 *locations would be uniformly distributed over the rectangle. You will do this on a homework.*

The G Function: Let h_i be the distance from a an observed event at a location $\mathbf{s}_i$ to the next nearest event location and let

$$I\left(h_i \leq h\right) = \begin{cases} 1 & h_i \leq h \\ 0 & h_i > h \end{cases}$$

I is referred to an *indicator function*. The empirical cumulative distribution function (ECDF) of the nearest neighbor distances is

$$\widehat{G}(h) = \frac{1}{n} \sum_{i=1}^{n} I\left(h_i \leq h\right).$$

The ECDF is computed for the observed pattern and also for each of g simulated patterns simulated under an assumption of CSR,

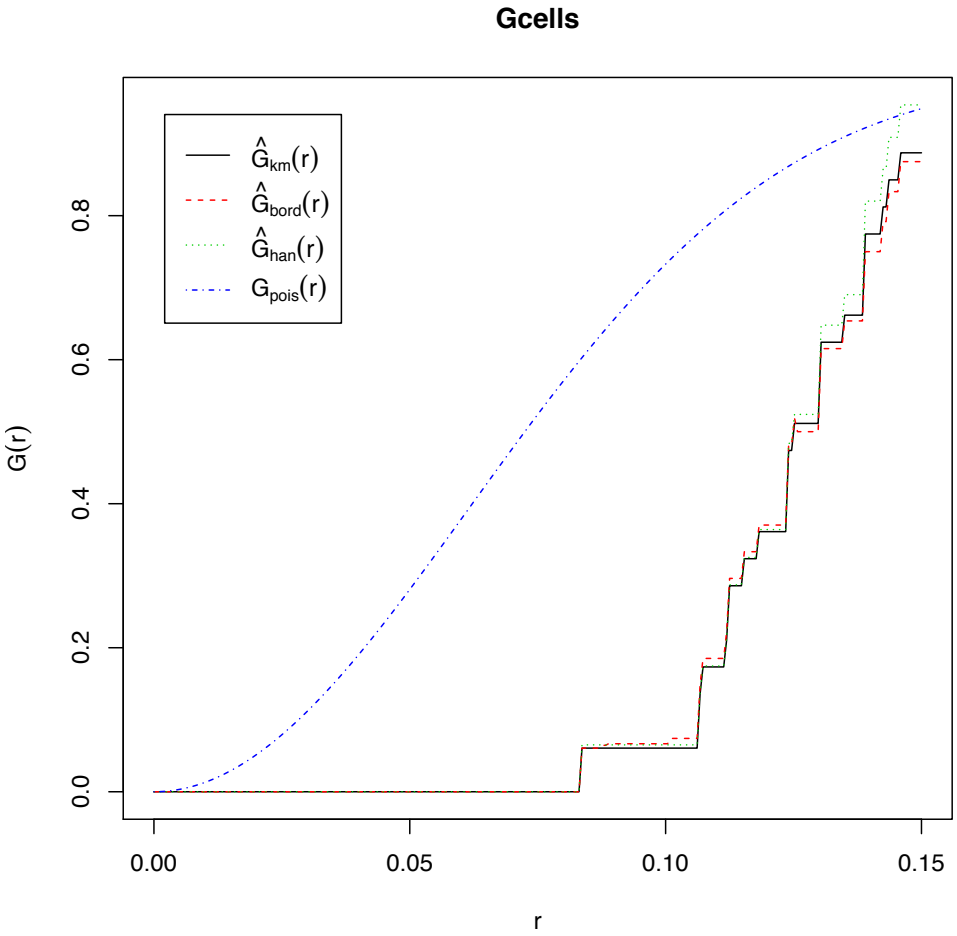
$$G(h) = 1 - \exp\left(-\lambda \pi h^2\right).$$

Edge correction is needed near the boundary of the window. From the documentation:

```
The estimation of G is hampered by edge effects arising from the
unobservability of points of the random pattern outside the window.
An edge correction is needed to reduce bias (Baddeley, 1998; Ripley, 1988).
The edge corrections implemented here are the border method or
reduced sample estimator, the spatial Kaplan-Meier
estimator (Baddeley and Gill, 1997) and the Hanisch estimator
(Hanisch, 1984).
```

We will look at a couple of these in more detail below when we discuss the K function.

Example: We use the `cells` *data set again. The function* `Gest` *computes the empirical cumulative distribution function of nearest neighbor distances for the observed data set and compares it to the cdf expected under CSR. This does not carry out a formal statistical test but the ocular p-value can be pretty low sometimes as it is here.*



The graph indicates a lot fewer short nearest-neighbor distances in the data than expected under an assumption of CSR. This is the result expected with regularly distributed point data.

Simulation Envelopes: A Monte Carlo based procedure for inference at this level is based on the use of simulation envelopes. The envelopes are generated using Monte Carlo simulations. The idea is best illustrated by example. Recall from above that h_i is the distance from $\mathbf{s}_i$ to the next nearest event location and let

$$I\left(h_i \leq h\right) = \begin{cases} 1 & h_i \leq h \\ 0 & h_i > h \end{cases}$$

The empirical cumulative distribution function (ECDF) of the nearest neighbor distances is

$$\widehat{G}(h) = \frac{1}{n}\sum_{i=1}^{n} I\left(h_i \leq h\right).$$

The ECDF is computed for the observed pattern and also for each of g simulated patterns (generally simulated under an assumption of CSR). Selected percentiles (e.g. minimum and maximum) of the g simulated ECDF's are computed and the observed ECDF can be compared to the envelope defined by the percentiles. Deviations from CSR can then be detected. For example, if the data are clustered then there should be more short nearest neighbor distances than expected CSR. The results are usually examined graphically. The `envelope` function produces plots of simulation envelopes.

Here is the description of the computation of simulation envelopes in the documentation of the `envelope` function in the `spatstat` package.

```
Upper and lower critical envelopes are computed in one of the following ways:
pointwise:
by default, envelopes are calculated pointwise (i.e. for each value of
the distance argument r), by sorting the nsim simulated values, and
taking the m-th lowest and m-th highest values, where m = nrank.
For example if nrank=1, the upper and lower envelopes are the
pointwise maximum and minimum of the simulated values.

The pointwise envelopes are not confidence bands for the true
value of the function! Rather, they specify the critical points for a
Monte Carlo test (Ripley, 1981). The test is constructed by choosing a
fixed value of r, and rejecting the null hypothesis if the observed
function value lies outside the envelope at this value of r. This test
has exact significance level

alpha = 2 * nrank/(1 + nsim).

simultaneous:
if global=TRUE, then the envelopes are determined as follows.
First we calculate the theoretical mean value of the summary statistic
(if we are testing CSR, the theoretical value is supplied by fun;
otherwise we compute the average of all the simulated values and take
this as an estimate of the theoretical mean value).
Then, for each simulation, we compare the simulated curve to the
theoretical curve, and compute the maximum absolute difference
between them (over the interval of r values specified by ginterval).
This gives a deviation value d[i] for each of the
nsim simulations. Finally we take the m-th largest of the deviation values,
where m=nrank, and call this dcrit. Then the simultaneous envelopes
are of the form lo = theo - dcrit and hi = theo + dcrit where theo is the
theoretical mean value. Simultaneous critical envelopes have constant
```

width 2 * dcrit.

The simultaneous critical envelopes allow us to perform a different Monte Carlo test (Ripley, 1981). The test rejects the null hypothesis if the graph of the observed function lies outside the envelope at any value of r. This test has exact significance level

alpha = nrank/(1 + nsim).

Edge corrections are handled as described below.

Edge corrections:
It is common to apply a correction for edge effects when calculating a summary function such as the K function. Typically the user has a choice between several possible edge corrections. In a call to envelope, the user can specify the edge correction to be applied in fun, using the argument correction. See the Examples below.

Summary functions in spatstat:
Summary functions that are available in spatstat, such as Kest, Gest and pcf, have a standard argument called correction which specifies the name of one or more edge corrections.

The list of available edge corrections is different for each summary function, and may also depend on the kind of window in which the point pattern is recorded. In the case of Kest (the default and most frequently used value of fun) the best edge correction is Ripley's isotropic correction if the window is rectangular or polygonal, and the translation correction if the window is a binary mask. See the help files for the individual functions for more information.

All the summary functions in spatstat recognise the option correction="best" which gives the best (most accurate) available edge correction for that function.

In a call to envelope, if fun is one of the summary functions provided in spatstat, then the default is correction="best". This means that by default, the envelope will be computed using the best available edge correction.

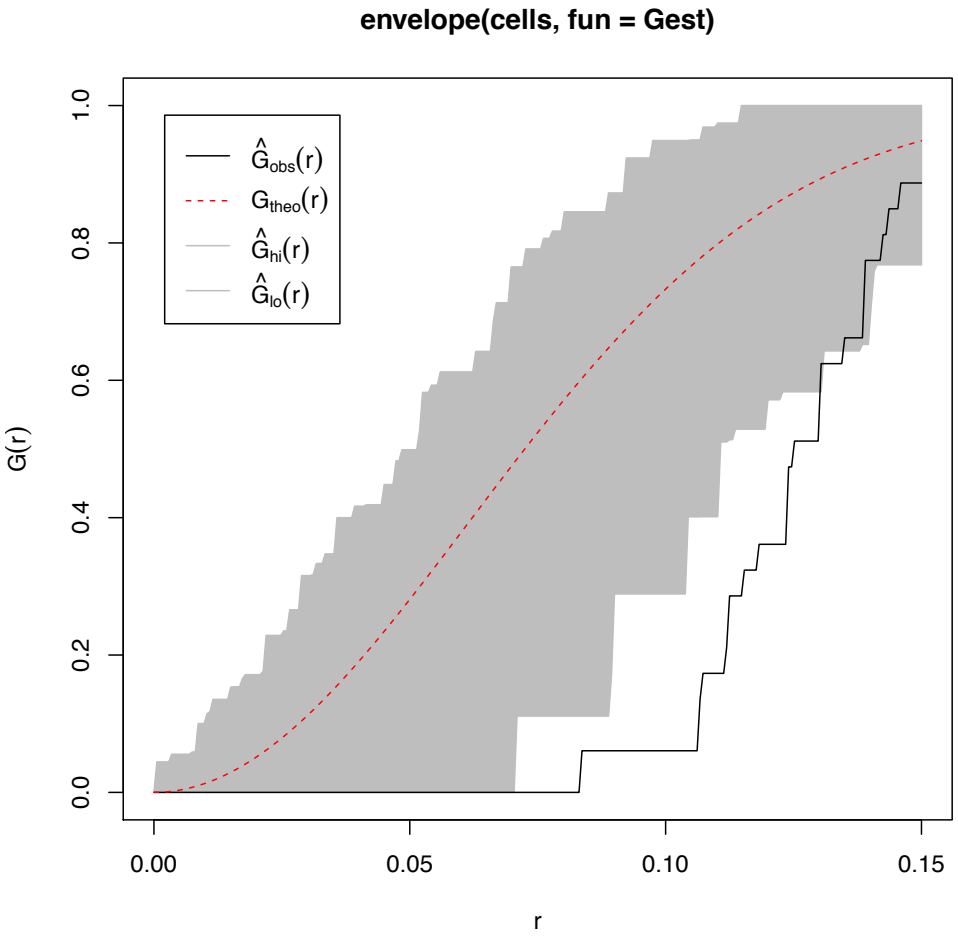
The user can override this default by specifying the argument correction. For example the computation can be accelerated by choosing another edge correction which is less accurate than the best one, but faster to

`compute.`

If you use `correction="best"` argument when you plot the G function as we did above R returns results for both the Kaplan-Meier and the border methods. It is not clear from the documenation which one of these two is being used in the `envelope` function below.

Example: Below is the graph of the simulation envelope for G function using the cell data. We accept the default values of the `envelope` *function.*

```
plot(envelope(cells,fun=Gest))
```



The default returns the pointwise minimum and maximum percentiles. Consider the distance of $h = 0.10$ (note R used r instead of h). Under the null hypothesis of CSR the proportion of nearest neighbor distance less than or equal to 0.10 should be about 0.73. The observed proportion of about 0.08 or so is clearly too low for CSR and outside the simulation envelope at that distance.

The exact two sided significance level of the test of CSR at that distance is

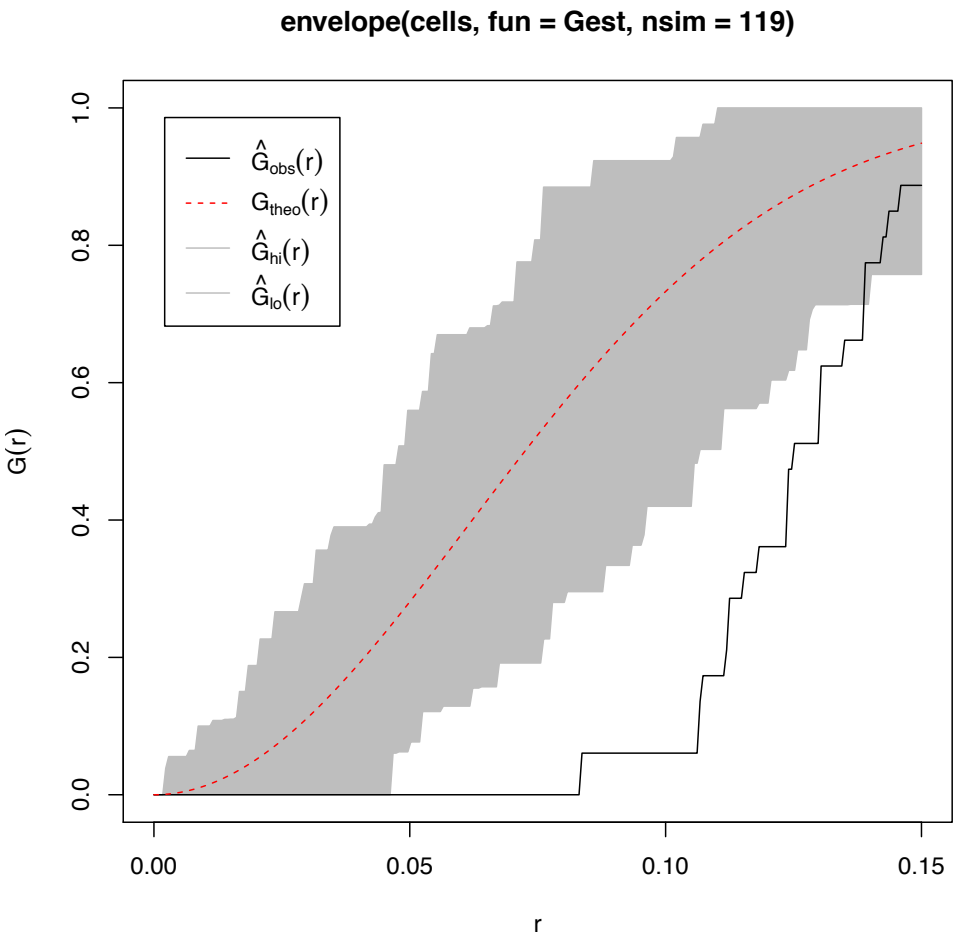
$$\alpha = 2\left[\frac{1}{100}\right] = 0.02.$$

If you wanted to carry out a fixed level α test you would need to find the `nrank` *value to achieve the desired significance level. You might have to also adjust the* `nsim` *value to achieve an exact level. For example if we wanted to conduct the test on the cells data set at $\alpha = 0.05$ with* `nsim` *equal to 99 we would find that we need to set* `nrank` *to 2.5 which will not work because* `nrank` *needs to be an integer. But we can set* `nrank` *equal to 3 and then adjust* `nsim` *to be equal to*

$$\frac{2(3)}{0.05} - 1 = 119.$$

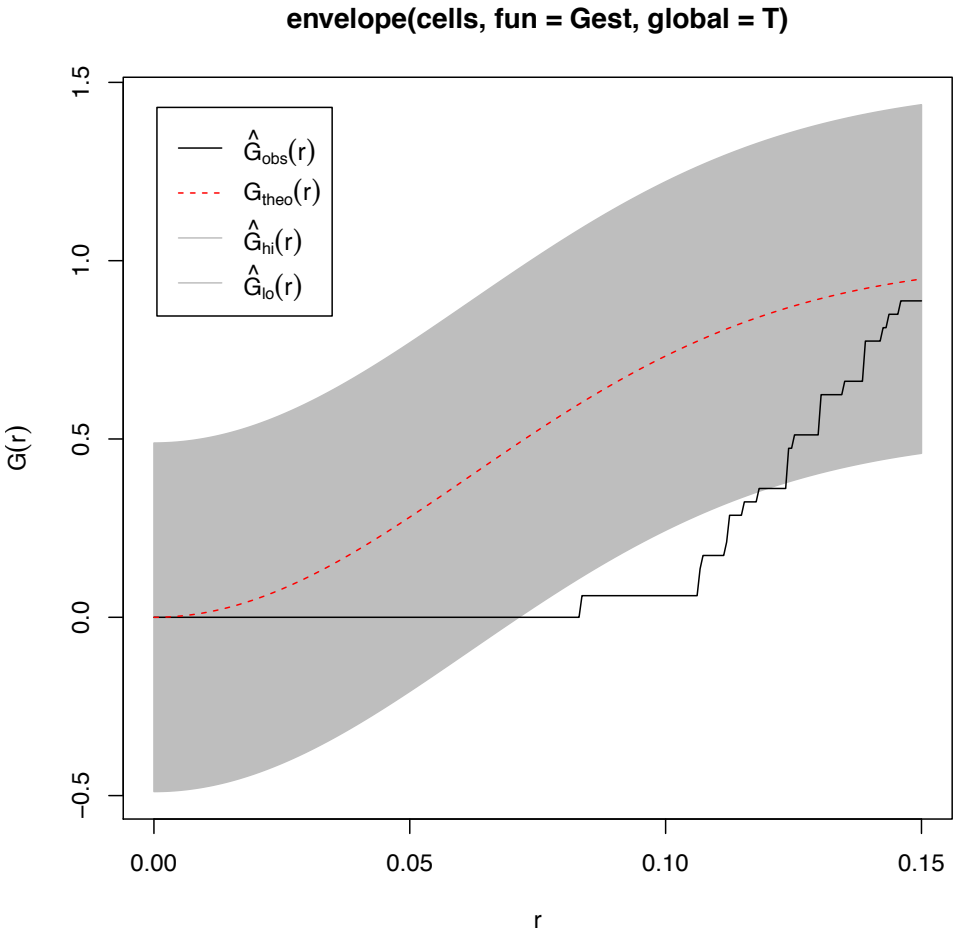
The resulting simulation envelope is seen below.

```
plot(envelope(cells,fun=Gest,nsim=119))
```



Obviously, if we are interested in many different h values then we run into all the problems associated with interpretation of multiple tests. This is less of a problem if one only uses the envelope as a descriptive tool. But if you are concerned about that the global test may be more appropriate. The default result is shown below for the cells data.

```
plot(envelope(cells,fun=Gest,global=T))
```



We can reject the null hypothesis of CSR for the entire function at level

$$\alpha = \frac{nrank}{(1 + \ nsim)}$$

if the observed ECDF falls outside the envelope at any value of h. Thus at level $\alpha = 0.01$ *we reject the null hypothesis of CSR for the G function of the cells data. If you wanted a different* α *level you could adjust* `nrank` *and or* `nsim` *as needed.*

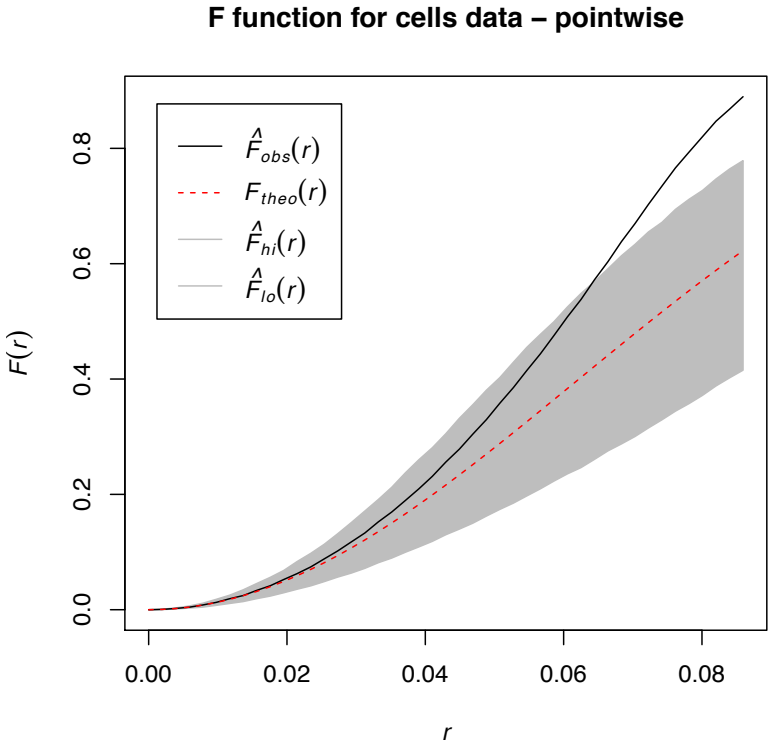
The F function: The G function is used for evaluating event to event distances. Another approach is based on evaluation of the distance between a fixed point in space and the nearest event location. The cumulative distribution of these distances is called the F function. Another name for it is the "empty space function". Letting h be the distance from a random point to nearest event location the derivation proceeds as with the G function and the theoretical cdf (under CSR) is the same

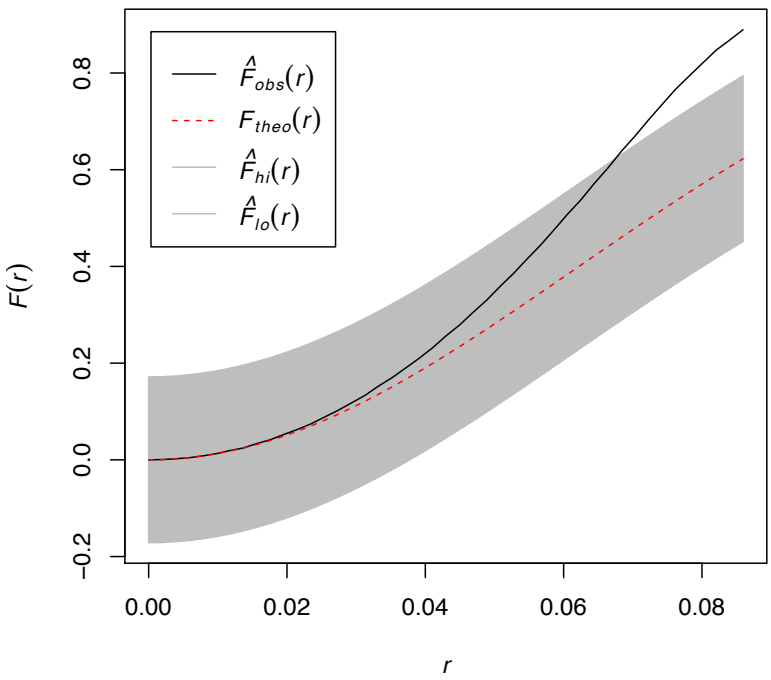
$$F(h) = 1 - \exp(-\lambda\pi h^2),\ h > 0$$

If event locations are regularly distributed then we should tend to see shorter distances from a fixed point to an event location than expected under CSR. If event locations are clustered then we should tend to see longer distances from a fixed point to an event location than expected under CSR. This is the opposite of the patterns we see with the G function.

Edge corrections are handled as described above except the Hanisch method is replaced by the Chiu-Stoyan estimator. We look at plots of simulation envelopes below where the Kaplan-Meier edge correction is used because it is judge to be the best.
Cells Example: The results for the cells example are shown below.

```
plot(envelope(cells,fun=Fest),main="F function for cells data")
plot(envelope(cells,fun=Fest,global=T),main="F function for cells data - global")
```





The estimates are smoother than we saw in the G function. `Fest` *has a parameter* `eps` *that* `Gest` *does not have and this results in the smoother empirical estimate - see documentation for details and note that the documentation suggests relying on the default for* `eps` *if you are new to the function. The results are what we expect. For small lags out to about* 0.02 *there is not much difference but once we get out to about* 0.06 *we see quite a bit more nearby event locations than expected under CSR (*0.6 *versus* 0.45*). This shows up in both the pointwise plot and in the global plot.*

The J function: This function incorporates information from both the G and F functions and is defined as

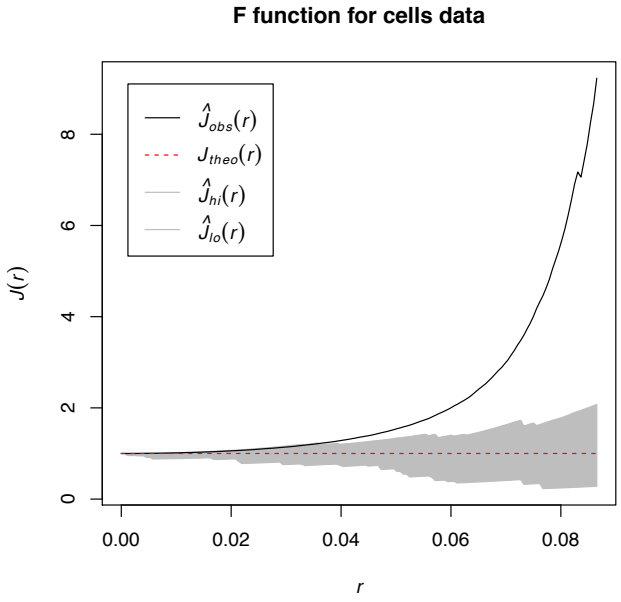
$$J(h) = \frac{1 - G(h)}{1 - F(h)}$$

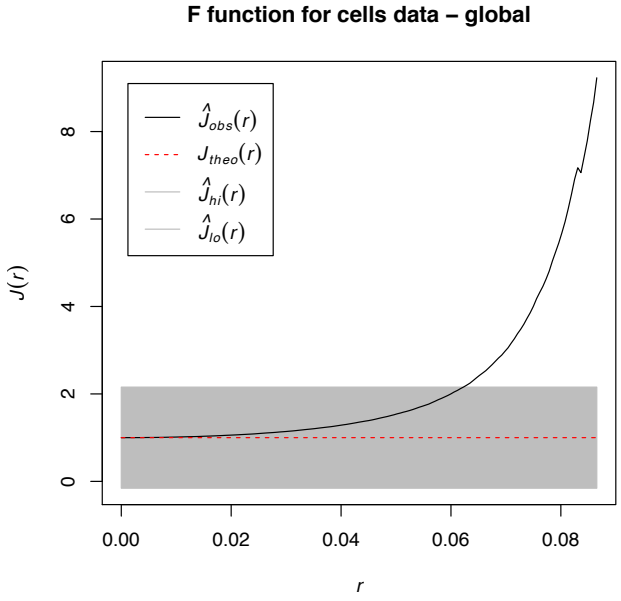
J should be equal to 1 everywhere under CSR with values of $J < 1$ indicating clustered data and $J > 1$ indicating regularly distributed data.

By default the Kaplan-Meier, Hanisch, and border corrected (reduced-sampling) methods for edge correction are all presented when just plotting results from `Jest` but the KM method is judged to be the best and is the default for `envelope`.

Cells Example: The plots below show results for the cells data set.

```
plot(envelope(cells,fun=Jest),main="F function for cells data")
plot(envelope(cells,fun=Jest,global=T),main="F function for cells data - global")
```





In both cases we evidence for a regular distribution of points.

The J function is nice. It is typically easier to detect deviations from a horizontal line than from the curved lines we are working with in the G and F functions. It does not hurt to look at all 3 though.

- The intensity function is a measure of the expected number of events per unit area and is fairly boring (in the sense that it is constant) in homogeneous processes. It will become a lot more interesting when we get to nonhomogeneous or inhomogeneous processes, when we will be concerned with spatial variation of the mean of the point process. The intensity is a measure of *first order* properties of a spatial point process.

- We may be interested in how often events occur within a specified distance of other events. Such relationships between events are measured by *second order* properties. Ripley's K function, and various modifications of it, is most commonly used for this purpose. It is defined to be

$$K(h) = \frac{E\left[\text{number of extra events within } h \text{ of a randomly chosen event}\right]}{\lambda} = \frac{E[h]}{\lambda}.$$

  Note the constant intensity λ. Note also that the expectation is taken over all events in a study area which implies a process that operates independently of location. Thus, implicit in the definition of K are assumptions of stationarity and isotropy.

- Under CSR we would expect

$$K(h) = \pi h^2$$

  because a homogeneous Poisson process will have intensity $\lambda\pi h^2$ in a circle of radius h. Regular processes should produce values of $K(h) < \pi h^2$ and clustered processes should produce values of $K(h) > \pi h^2$.

- *Estimation of* $K(h)$: For a region A the best estimator of λ is

$$\widehat{\lambda} = \frac{N(A)}{\nu(A)}.$$

  The numerator of the K function is an expectation and the first choice for estimation of such quantities is a sample mean of some kind. We need to keep the target in mind: the expected (or mean) number of extra events within a distance h of a randomly chosen event. Consider location $\mathbf{s}_i$ of the ith event. First, count the number of events at locations $\mathbf{s}_j, j \neq i$ within h of $\mathbf{s}_i$. Letting $h_{ij} = \|\mathbf{s}_i - \mathbf{s}_j\|$ this is

$$\sum_{j \neq i}^{n} I\left(h_{ij} \leq h\right).$$

  Then we sum this quantity over all n points

$$\sum_{i=1}^{n} \sum_{j \neq i}^{n} I\left(h_{ij} \leq h\right).$$

Finally, we divide by the number of observed events which yields:

$$\tilde{E}[h] = (1/n) \sum_{i=1}^{n} \sum_{j \neq i}^{n} I\left(h_{ij} \leq h\right).$$

This estimator is biased low however because of edge effects which become more of a problem with increasing h. Events occurring outside the specified region are not counted. There have been several edge correction methods proposed over the years. We will focus on the ones that are implemented in R.

- Border Method: This method is also referred to as the *reduced sample estimator*, a term which gives a clue as to how it works. Let d_i be the closest distance from event location $\mathbf{s}_j$ to a border. As long as $h \leq d_i$ $\mathbf{s}_i$ will be used in estimation of $K(h)$. Once $h > d_i$ $\mathbf{s}_i$ is dropped from the sample. Then (using the notation in the text)

  $$E^*[h] = \left[\frac{1}{\sum_{i=1}^{n} I\left(d_i > h\right)}\right] \sum_{i=1}^{n} \left\{ \sum_{j \neq i}^{n} I\left(h_{ij} \leq h \text{ and } d_i > h\right) \right\}$$

  and the estimator of $E[h]$ is

  $$\widehat{E}_d(h) = \begin{cases} E^*(h) & \sum_{j=1}^{n} I\left(d_j > h\right) > 0 \\ 0 & \text{else} \end{cases}$$

- Isotropic (Ripley) Method: To see how this one works we assume that we have a specified distance h and an event location $\mathbf{s}_i$. A neighbor $\mathbf{s}_j$ lies $h_{ij} < h$ from $\mathbf{s}_i$ but instead of automatically giving this neighboring location its full weight (as in the naive estimator above) we weight it by the inverse of the proportion p_{ij} of the area of a circle with radius h_{ij} that lies in the region. Thus the estimator is

  $$\widehat{E}(h) = \frac{1}{n} \sum_{i=1}^{n} \sum_{j \neq i}^{n} \left(\frac{1}{p_{ij}}\right) I\left(h_{ij} < h\right).$$

- There are at least 2 K function estimation functions available in R. One, `Kfn`, is in Ripley's `spatial` library and actually estimates the L function which we will talk about a bit more below. It uses only the isotropic edge correction method. The other, `Kest` is available in `spatstat` and it has the capability of using either the isotropic and boundary methods (as well as a third method called the translation method which we will ignore).

- As indicated above the expected value of the K function under CSR is πh^2 so a plot of $K(h)$ versus h should be a parabola and deviations from that expected parabola

indicate a lack of randomness. Visually it easier to detect deviations from a straight so the K function is often transformed into the L function:

$$L(h) = \left(\frac{K(h)}{\pi}\right)^{1/2}$$

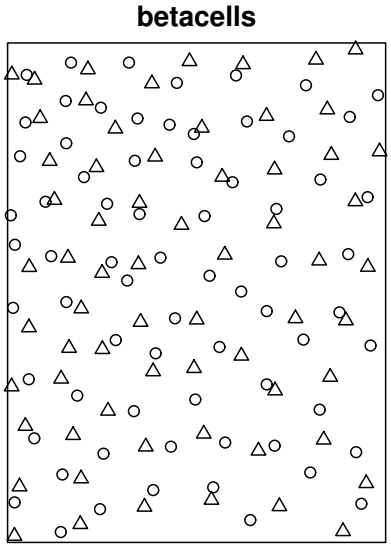
which has expectation under CSR of

$$E\left[L(h)\right] = h.$$

$L(h)$ can be plotted versus h as a diagnostic tool. Some authors suggest plotting $\widehat{L}(h) - h$ versus h. This is analogous to a residual plot. Under CSR $\widehat{L}(h) - h$ has expectation 0 so points above the horizontal line at $\widehat{L}(h) - h = 0$ indicate clustering and points below that line indicate regularity.

- Random Thinning: Diggle defines the random thinning of a point process P as "a point process whose events are a subset of the events of P generated by retaining or deleting the events of P in a series of mutually independent Bernoulli trials." One reason K functions are often used is that their interpretation is not impacted by random thinning as long as the thinning is "spatially neutral". This would not be true of some of the other functions we have looked at. Diggle proves this result in the text I showed you in in class. This has nice implications in, for example, spatial epidemiology where we rarely have all cases of a disease. However, we can still explore spatial clustering/regularity for disease cases as long as we are willing to assume they are a realization of a spatially neutral random thinning.

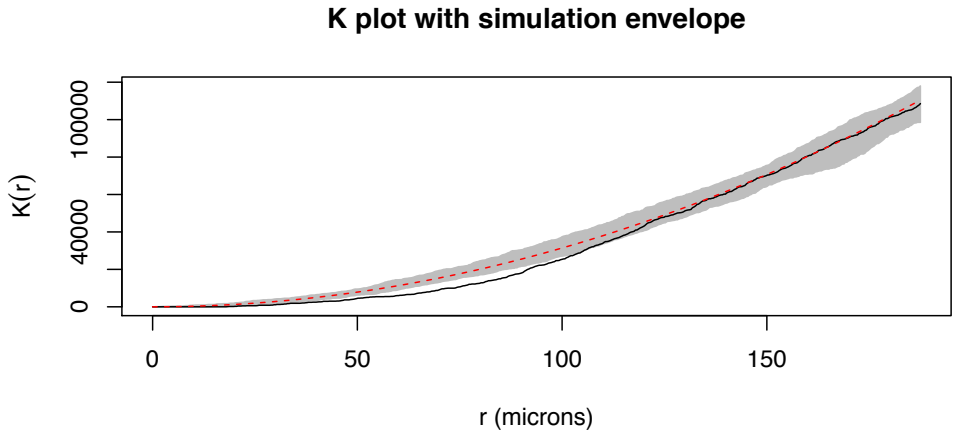
- *Example: The* `spatstat` *library contains a data set called* `betacells` *which contains 135 locations of 2 different types of cells in the retina of a cat. The cells are of 2 different types, 65 cells are* on *and 75 cells are* off. *A plot of the cells is shown below.*

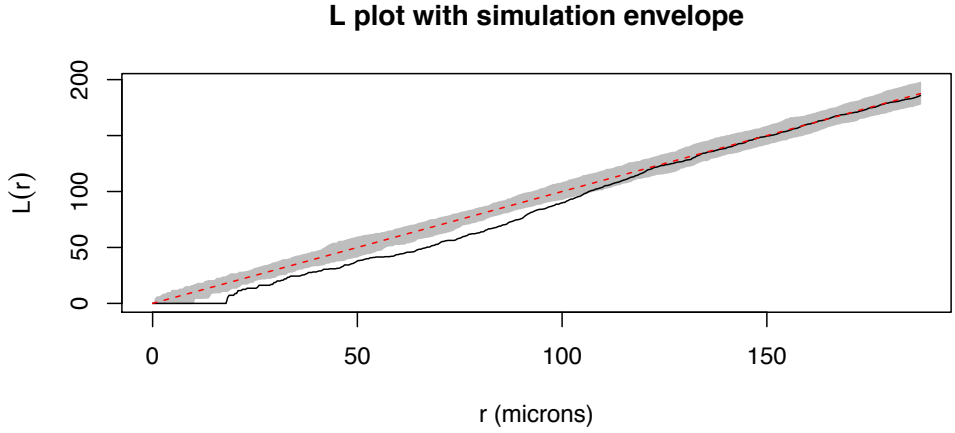


The 2 types are indicated by different plotting characters.

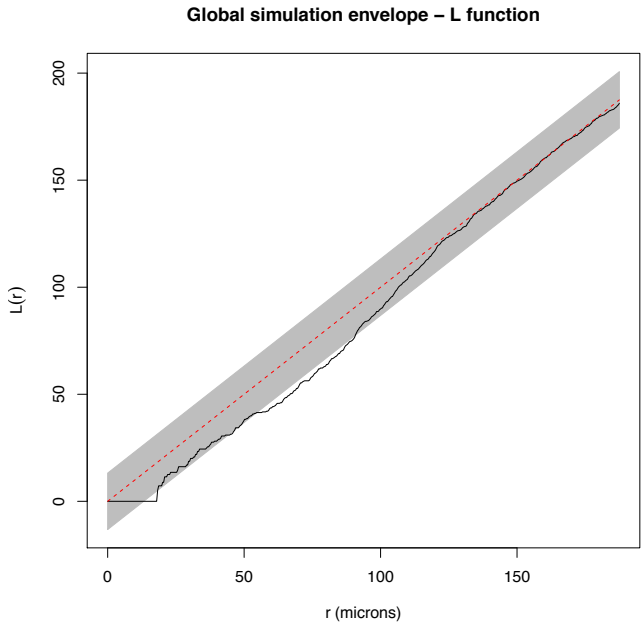
First we will look at plots of the K and L functions for all the data (ignoring marks) along with simulation envelopes with `nsim=99`.

```
plot(envelope(betacells,fun="Kest",correction="iso"))
plot(envelope(betacells,fun="Lest",correction="iso"))
```





We are only looking at the isotropic edge correction results (I believe that the isotropic method is the default but I cannot find any discussion of this in the documentation and I wanted to be sure). There is some evidence of a regular distribution. I think it is more evident in the L plot than in the K plot. The global simulation plot also suggests this.



This plot was produced by

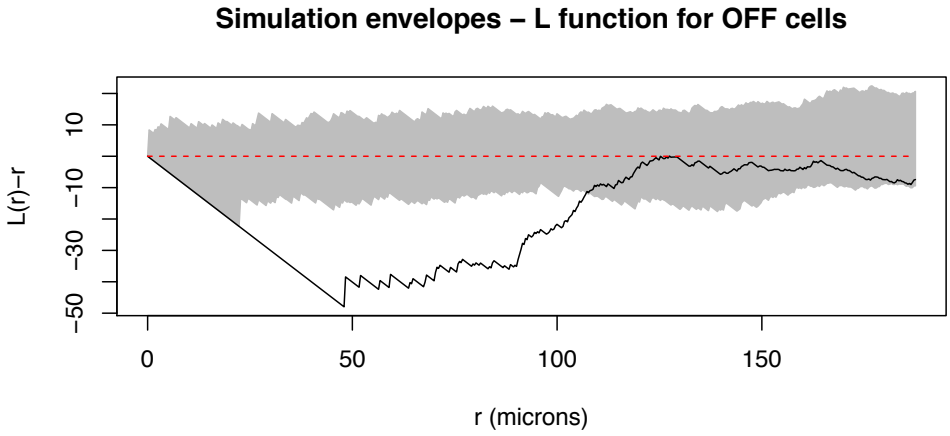
```
Lenv<-envelope(betacells,fun=Lest,global=T,correction="iso",nsim=99)
plot(Lenv,legend=F,main="Global simulation envelope - L function")
```

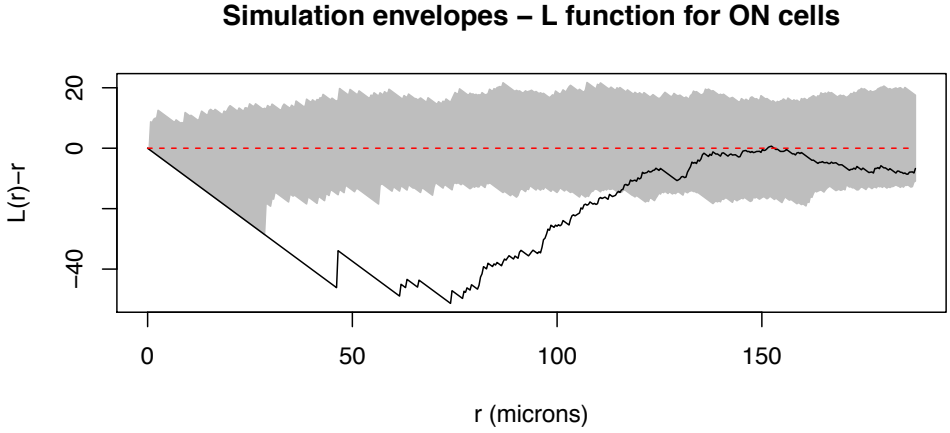
I actually also checked the plot with `nim=499` *partly to see how long it took. It was quite fast although the band was quite a bit wider and evidence for regularity was weaker as expected Normally you would be consistent with* `nsim`.

In the plot of the point pattern above we see that the 2 types are indicated by different plotting characters. We should look at plots for the two types of point processes separately. Below is a plot of $\widehat{L} - h$ *for both the on and off cells.*

As just mentioned we saw evidence of regularity for both processes evaluated simultaneously. We see pretty strong visual evidence for regularity in both point patterns separately. Further the 2 appear to be quite similar to one another.

```
betacells.off<-split(betacells)$off
betacells.on<-split(betacells)$on
Loff.env<-envelope(betacells.off,Lest)
Lon.env<-envelope(betacells.on,Lest)
plot(Loff.env,.-r~r,ylab="L(r)-r",
  main="Simulation envelopes - L function for OFF cells")
plot(Lon.env,.-r~r,ylab="L(r)-r",
  main="Simulation envelopes - L function for ON cells")
```





- *Comparing Two Patterns*: The `betacells` data set is an example of a marked point process with a binary auxiliary variable indicating cell type. We have seen that the intensity is a mean, the K function is analogous to a variance, and now we need a "covariance" type function. The function is called a cross-K function and is defined to be

$$K_{ij}(h) = \frac{E\left[\text{number of type } j \text{ events within distance } h \text{ of an arbitrary type } i \text{ event}\right]}{\lambda_j}$$

where λ_j is the intensity of the type j process. If $i = j$ this reduces to the K function for the i type of process. The isotropic edge corrected estimator of K_{ij} is

$$\widehat{K}_{ij}(h) = \frac{1}{\widehat{\lambda}_i \widehat{\lambda}_j \nu(A)} \left[\sum_k \sum_l w\left(\mathbf{s}_k, \mathbf{u}_l\right)^{-1} I\left(h_{kl} \leq h\right) \right]$$

where $\widehat{\lambda}_i$ is the estimated intensity of process i, $\widehat{\lambda}_j$ is the estimated intensity of process j, $h_{kl} = \|\mathbf{s}_k - \mathbf{u}_l\|$, and $w\left(\mathbf{s}_k, \mathbf{u}_l\right)$ is the proportion of the circumference of a

circle centered at location $\mathbf{s}_k$ with radius h_{kl} that lies inside region A. Although K_{ij} is symmetric, $\widehat{K}_{ij}$ is not, so the recommended estimator (and the one implemented in `spatstat`) is

$$K^*_{ij} = \frac{\widehat{\lambda}_j \widehat{K}_{ij}(h) + \widehat{\lambda}_i \widehat{K}_{ji}(h)}{\widehat{\lambda}_j + \widehat{\lambda}_i}.$$

We are once again assuming second order stationarity and isotropy.

There are 2 related null hypotheses that can be tested: (1) the 2 processes are independent and (2) the random labeling hypothesis.

1. Independence: The spatial locations and the binary marks are determined simultaneously and independently of one another. If true then

   $$K_{ij}(h) = \pi h^2.$$

   Diggle notes that this follows because under independence the expected number of Type 2 events within h of a Type 1 event is the same as the expected number of such events within h of an arbitrary point. That expected number is $\lambda_2 \pi h^2$. Also note that this holds regardless of the pattern in the two events.

   Schabenberger and Gotway suggest working with a cross L function

   $$L^*_{ij}(h) = \sqrt{\frac{K^*_{ij}(h)}{\pi}}$$

   which is equal to h under the null hypothesis of independence. Plots of L^*_{ij} versus h or plots of $L^*_{ij}(h) - h$ versus h can be examined to assess the null. If $L^*_{ij}(h) - h > 0$ then the 2 processes attract one another while $L^*_{ij}(h) - h < 0$ indicates inhibition.

2. Random Labelling: Locations arise from a univariate spatial process with marks subsequently assigned randomly and independently by a second process. Under this hypothesis

   $$K_{11}(h) = K_{22}(h) = K_{12}(h) = K(h)$$

   because under random labelling K_{11}, K_{22} and K_{12} are just randomly thinned versions of K.

   A test statistic for comparing K_{ii} and K_{jj} is

   $$D(h) = \widehat{K}_{ii}(h) - \widehat{K}_{jj}(h)$$

   where $\widehat{K}_{ii}$ and $\widehat{K}_{jj}$ are the estimated K functions for the 2 types of processes.

These 2 hypotheses are not the same. They imply that 2 different random processes are operating. Under independence we assume that locations and marks are determined simultaneously and independently of one another, i.e. independence arises from a bivariate point process model with the 2 components operating independently of one another. Random labeling implies that locations arise from a univariate point process followed by a second random process determining the marks. Put another way, the spatial process of type i events can be viewed as a random thinning of the unmarked process which itself is a superposition of type i and type j events. These 2 hypotheses will be equivalent if and only if the superposition of the 2 processes yields a completely random pattern (CSR).

Testing generally involves Monte Carlo procedures.

1. It is difficult to formally test the null hypothesis of independence because a complete bivariate null model must be specified. Diggle (1983: Statistical Analysis of Spatial Point Patterns) suggests a method for rectangular regions based on toroidol shifts. The region is converted into a cylinder and then one of the patterns is shifted randomly around the cylinder. A suitable test statistic is calculated for the observed data and compared to simulated values. One logical test statistic is $\widehat{K}_{12}$ itself. The `spatstat` package does not have the capability to do this (so far as I can determine) but the `splancs` package does.

2. It is easier to carry out Monte Carlo tests of the random labelling hypothesis. Suppose we have a data set with n spatial locations, n_1 of Type 1 and n_2 of Type 2. We estimate the K function for the Type 1 process, K_{11}. We then randomly sample n_1 locations from the n locations, estimate the K function for the resulting "thinned" process and compare it to the K function computed for the original Type 1 process. We do this a specified number of times and produce simulation envelopes when done. If the random labelling hypothesis is valid then the K function for the original Type 1 locations should not be unusual when compared to the simulation results. Alternatively and equivalently, we could randomly scramble the marks among the n locations, compute K functions for the new "Type 1" process and compare randomized K functions to the true K function for that process.

   Another approach is to test for equality of $K_{ii}(h) = K_{jj}(h)$ using

$$D(h) = \widehat{K}_{ii}(h) - \widehat{K}_{jj}(h)$$

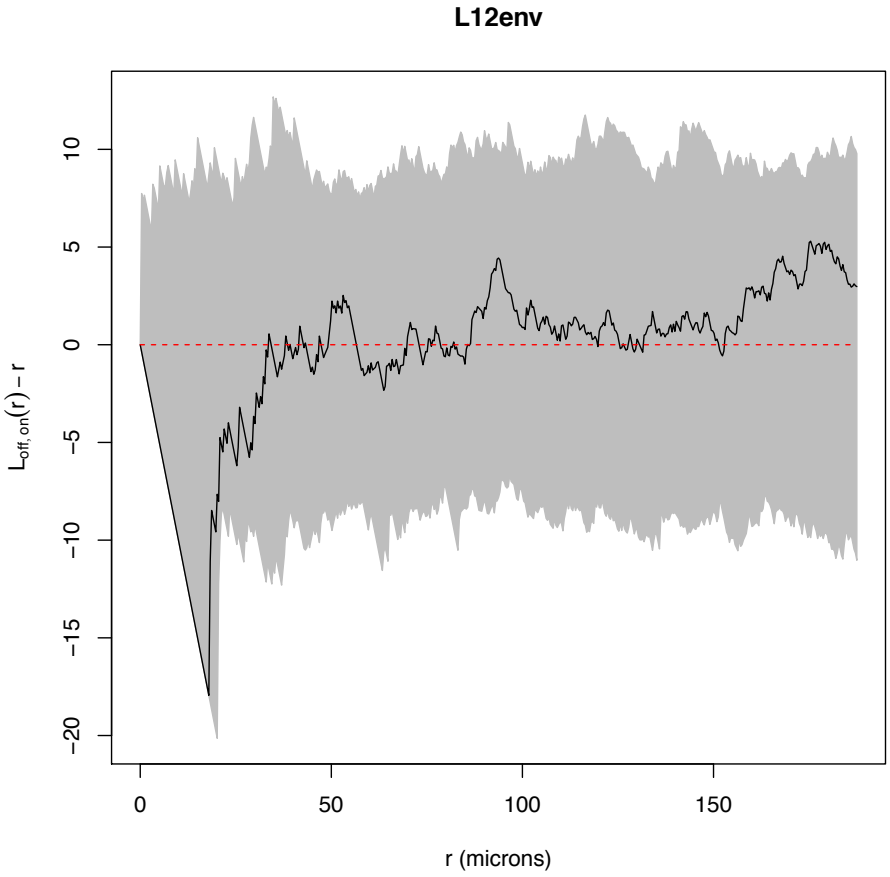
   where $\widehat{K}_{ii}$ and $\widehat{K}_{jj}$ are the estimated K functions for the 2 types of processes. Starting with the observed data, estimate K functions for both types of processes separately and compute D. Then randomly relabel the locations, and compute D for each relabelling. The observed value of D can be compared to the randomization distribution. There is a function in `splancs` that will do this.

However, it is important to note that equality of K_{11} and K_{22} is not enough for random labelling; they both have to equal K_{12} (and K). This is something Waller and Gotway do not even address.

Example: We continue with the `betacells` *example. Typically an investigator would have some idea of whether independence or random labelling is the primary hypothesis of interest. We would not expect to see both in this example because the point pattern is clearly not CSR. But we will look at both here. Based on the above we have graphical evidence of a regular pattern for both types of cells and evidence that the two patterns are similar to one another.*

We are of course interested in the relationship between these two patterns. A plot of a modified cross-L function ($\widehat{L}_{12} - h$) and an associated simulation envelope from `spatstat` *is easy to produce. Note that we are only looking at the isotropic edge corrected version.*

```
# betacells is a marked process with 2 different types of marks
# and we only want the on/off binary
betacells.new<-betacells
betacells.new$marks<-betacells$marks[,1]
L12env<-envelope(betacells.new,fun=Lcross,correction="iso")
plot(L12env,.-r~r,legend=F)
```



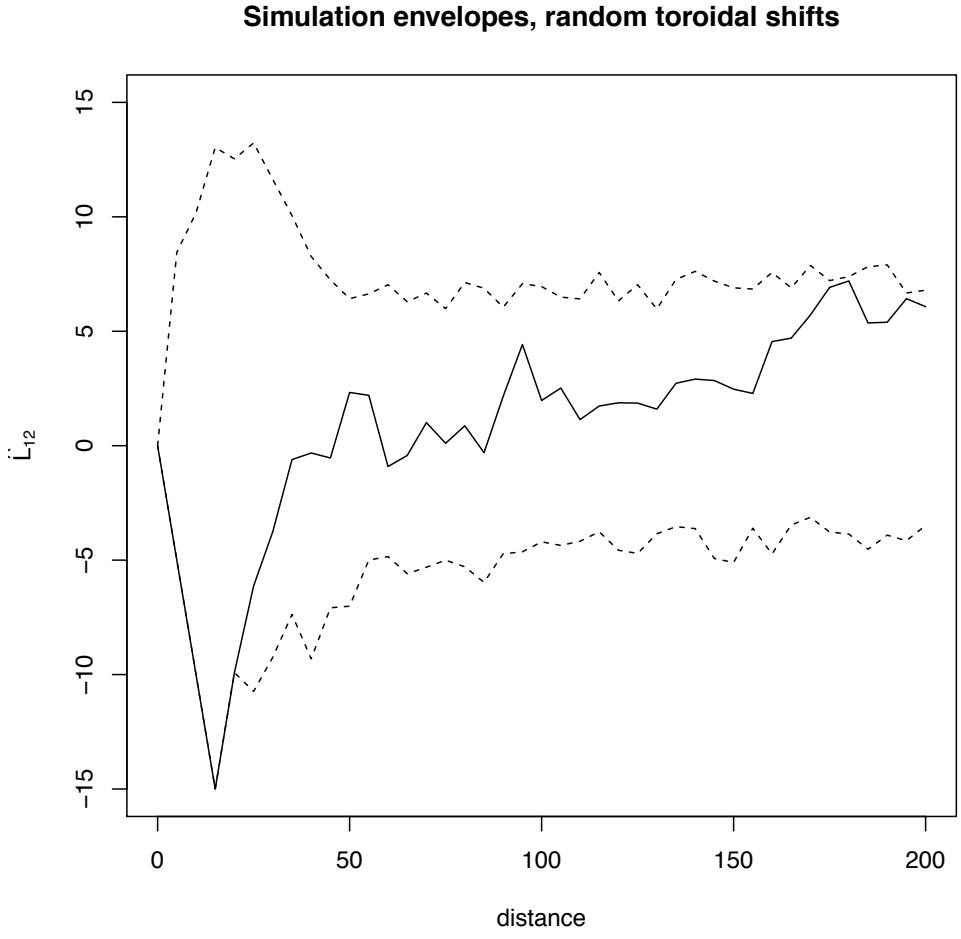
It appears that the empirical cross L function is consistent with independence, although that behavior at distances of about 20 *microns is a bit odd. The empirical function is within the envelope but there is a lot of variability .*

In this example, we have some evidence that the cells of both types are regularly distributed. If we simulated two independent regularly distributed processes would we get the same or a very similar envelope?

As far as I can determine `spatstat` *does not have the capability to produce this type of plot. The* `splancs` *package does possess this capability though. The relevant functions are* `k12hat` *and* `Kenv.tor` *with the latter implementing the torus idea described above. R code for such an assessment is shown below. It is harder to produce the plot. I set* `nsim=499`*. It runs very fast and results for smaller* `nsim` *values were quite variable whereas thiings stabilized quite a bit with a higher number of simulations.*

```
# get the coordinates of the two types
betacells.poly <- list(x=c(betacells.on$x, betacells.off$x),
```

```
y=c(betacells.on$y,betacells.off$y))
# specify the range of radii for the plot
h<-seq(0,200,5)
# Produce a modified cross L function
Lcross.plot<-sqrt(k12hat(as.points(betacells.on),as.points(betacells.off),
             bboxx(bbox(as.points(betacells.poly))),h)/pi) - h
plot(h,Lcross.plot, xlab="distance",
     ylab=expression(hat(L)[12]), ylim=c(-15,15), type="l",
     main="Simulation envelopes, random toroidal shifts")
# Get the bounds on the simulation envelope.
Lcross.env<- Kenv.tor(as.points(betacells.on), as.points(betacells.off),
     bboxx(bbox(as.points(betacells.poly))), nsim=499, s=h)
lines(h, sqrt(Lcross.env$upper/pi)-h, lty=2)
lines(h, sqrt(Lcross.env$lower/pi)-h, lty=2)
```



The results are similar but not the same as seen in the plot produced by `spatstat`*. In particular the simulation envelope is narrow at greater distances and the difference appears to be too large to be due to simulation variability. The results are consistent with independence.*

The independence assumption does appear reasonable but we also have evidence that the two processes are similar to one another. It would be nice to somehow assess how similar and one obvious way is to look at $\widehat{K}_{11} - \widehat{K}_{22}$ *and an associated simulation envelope of the difference. It is easy to compute the difference but the envelope is harder.*

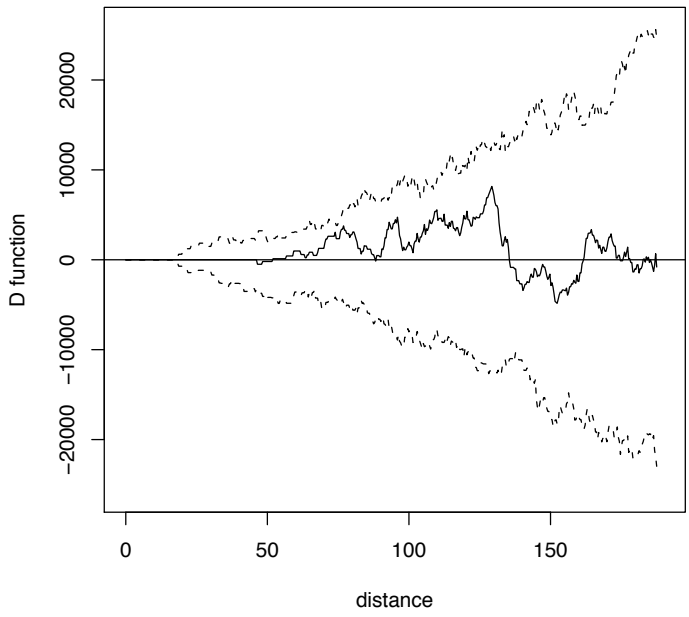
When I first covered this material there were no R functions available to carry out any of the Monte Carlo tests of random labelling so I wrote some code to do it.

```
## D function calculation
# First calculate the observed value of D
#
betacells.off<-split(betacells)$off
betacells.on<-split(betacells)$on
Dfun<-Kest(betacells.off)$iso - Kest(betacells.on)$iso
## Set up
mark.vec<-betacells$marks
sim.mat<-matrix(0,nrow=513,ncol=100)
sim.mat[,1]<-Dfun
betacells.new<-betacells
## Loop
for(i in 2:100){
   betacells.new$marks<-sample(mark.vec,rep=F)
   betacells.newoff<-split(betacells.new)$off
   betacells.newon=split(betacells.new)$on
   sim.mat[,i]=Kest(betacells.newoff)\$iso -
               Kest(betacells.newon)\$iso
}
```

We can get the minimum and maximum values of the D function and plot the results.

```
Dfun.min<-apply(sim.mat[,-1],1,min)
Dfun.max<-apply(sim.mat[,-1],1,max)
hdist<-Kest(betacells)$r
min(Dfun.min)  #  -22985.15
max(Dfun.max)  #   25608.21
max(hdist) #  187.5
plot(c(0,190),c(-26000,26000),type="n")
```

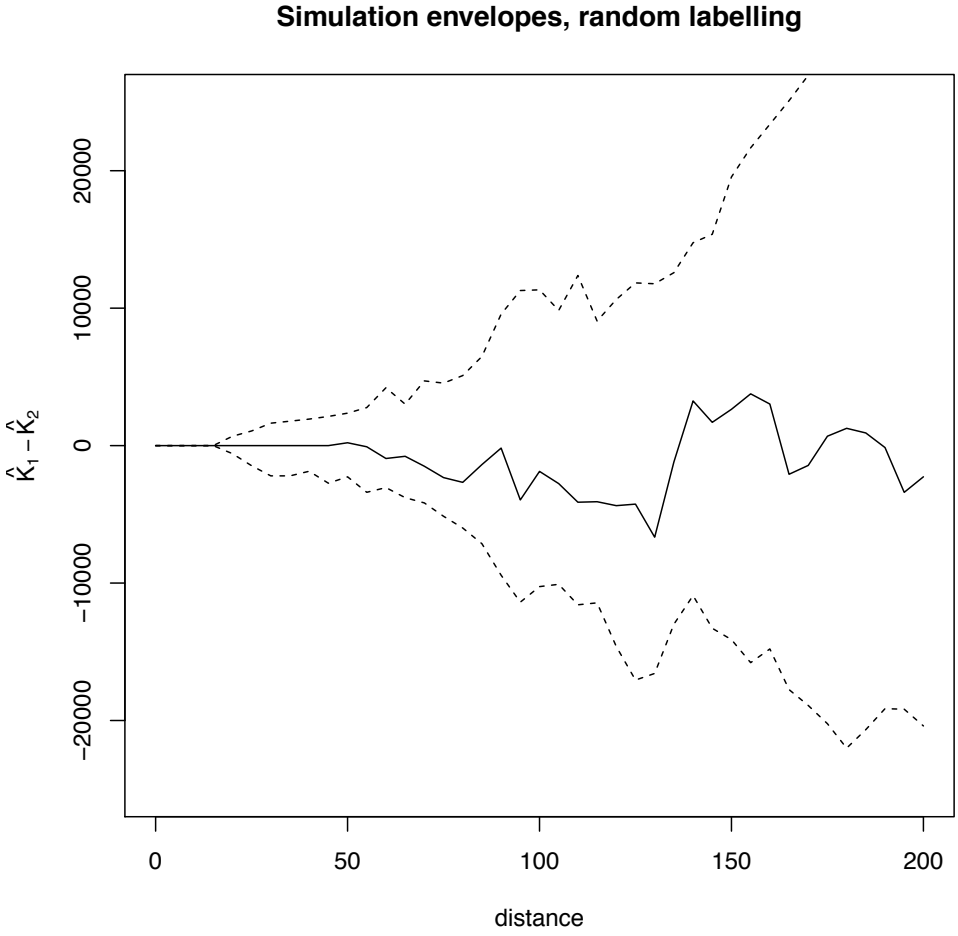
```
lines(hdist,Dfun)
abline(h=0)
plot(c(0,190),c(-26000,26000),type="n",xlab="distance",ylab="D
function")
abline(h=0)
lines(hdist,Dfun.min,lty=2)
lines(hdist,Dfun.max,lty=2)
lines(hdist,Dfun)
```



However, there is now a function `Kenv.label` *in the* `splancs` *package that will do it for us. The two are equivalent although the* `Kenv.label` *function is much more efficient and designed for general use.*

```
# compute the separate K functions and get their difference
K1.hat <- khat(as.points(betacells.on), bboxx(bbox(as.points(betacells.poly))), h)
K2.hat <- khat(as.points(betacells.off), bboxx(bbox(as.points(betacells.poly))), h)
K.diff <- K1.hat-K2.hat
# plot the results
plot(h, K.diff, xlab="distance", ylab=expression(hat(K)[1]-hat(K)[2]),
ylim=c(-25000,25000), type="l", main="Simulation envelopes, random labelling")
# generate the envelope
env.lab <- Kenv.label(as.points(betacells.on), as.points(betacells.off),
```

```
          bboxx(bbox(as.points(betacells.poly))), nsim=99, s=h)
lines(h, env.lab$upper, lty=2)
lines(h, env.lab$lower, lty=2)
```

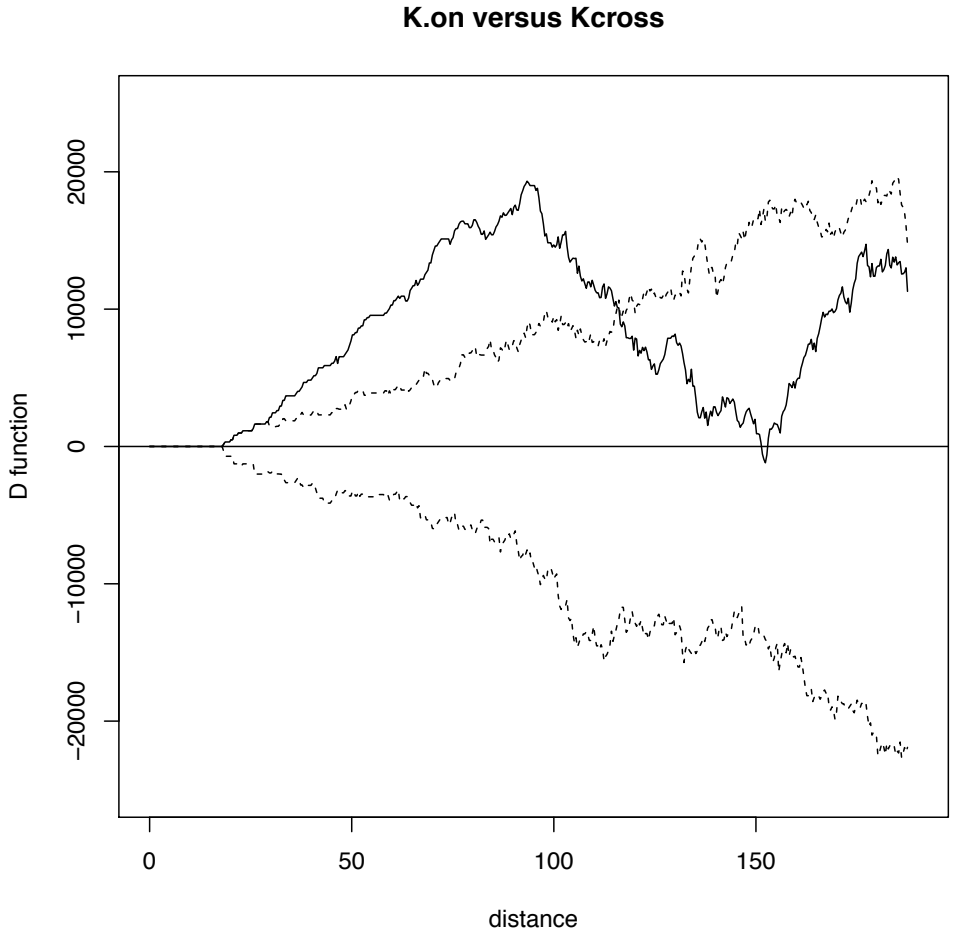


This suggests that the 2 processes may be very similar. However, recall that equality of K_{11} and K_{22} does not by itself indicate random labelling. We also need these two to be equal to K_{12}. I modified my clunky R code above to compare K_{11} and K_{22} to K_{12} do this as `Kenv.label` *cannot be easily modified for this purpose and the results do suggest evidence against random labelling*

Could we have random labelling? Under independence we expect $K_{12}(h) = \pi h^2$. Under random labelling we expect $K_{11}(h) = K_{12}(h) = K_{22}(h)$. At this point the graphical evidence that $K_{12}(h) \approx \pi h^2$ is not compelling but and $K_{11}(h) \approx K_{22}(h) \neq K_{12}(h)$. In other words the evidence for independence is evidence against random labelling. If we did have reason to expect random labelling one way to test that

would be to look at the difference between K_{11} and K_{12}. The `splancs` *package has a function* `Kenv.label` *which will do this for us. The plot below shows the results for comparison of the K_{11} versus K_{12}. The plot for K_{22} versus K_{12} is very similar.*

```
## D function calculation - cross K and K.on
# First calculate the observed value of D
# I had to modify the code a bit because betacells
# has two types of marks and Kcross had trouble with that.
# I only needed to work with
# the binary type (on,off) marks.
betacells.new<-betacells
betacells.new$marks<-betacells$marks[,1]
betacells.newoff<-split(betacells,f=marks(betacells.new))$off
betacells.newon<-split(betacells,f=marks(betacells.new))$on
## Set up
mark.vec<-marks(betacells.new)
sim.mat<-matrix(0,nrow=513,ncol=100)
Dfun<-Kcross(betacells.new)$iso - Kest(betacells.newon)$iso
sim.mat[,1]<-Dfun
## Loop
for(i in 2:100){
   indx<-sample(1:135,rep=F)
   betacells.new$marks<-betacells.new$marks[indx]
   betacells.newoff<-split(betacells.new,f=marks(betacells.new))$off
   betacells.newon<-split(betacells.new,f=marks(betacells.new))$on
   sim.mat[,i]<-Kcross(betacells.new)$iso -
               Kest(betacells.newon)$iso
}

Dfun.min<-apply(sim.mat[,-1],1,min)
Dfun.max<-apply(sim.mat[,-1],1,max)
hdist<-Kest(betacells)$r
min(Dfun.min)  #  -22669.51
max(Dfun.max)  #   19640.8
max(hdist) #  187.5
plot(c(0,190),c(-25000,25000),type="n",xlab="distance",ylab="D function")
abline(h=0)
lines(hdist,Dfun.min,lty=2)
lines(hdist,Dfun.max,lty=2)
lines(hdist,Dfun)
```



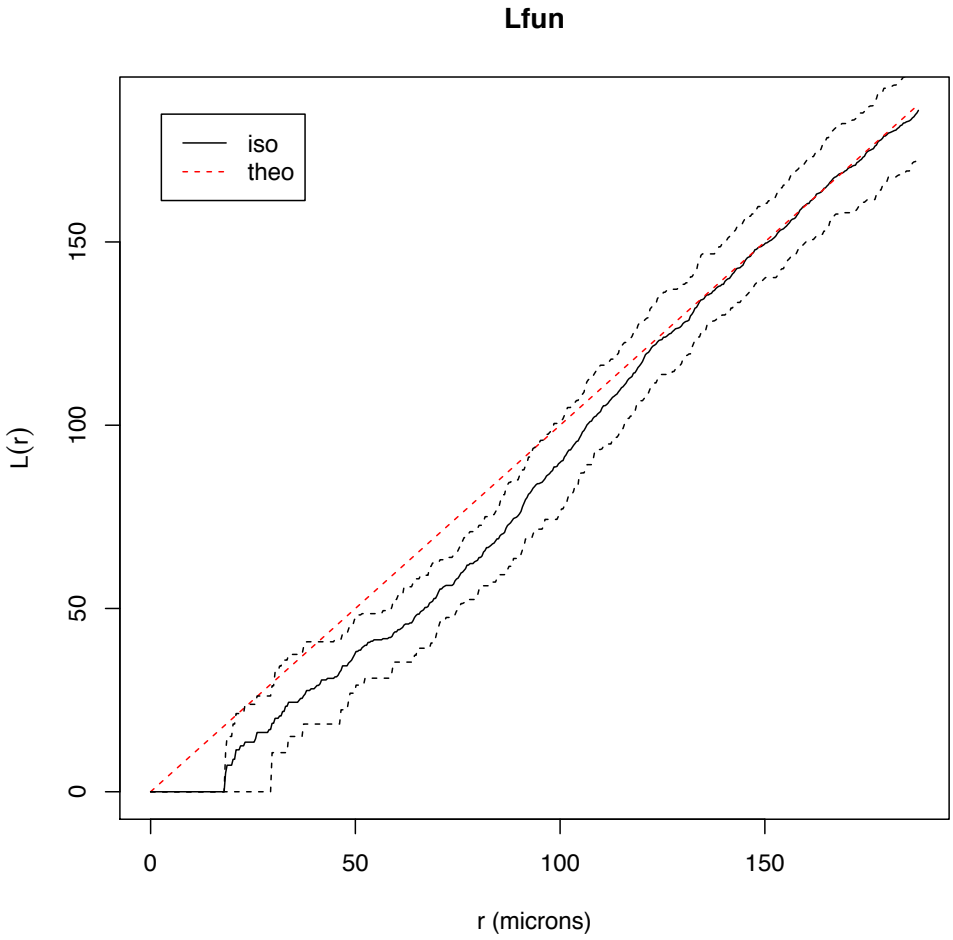
It is pretty obvious that we do not have $K_{11} = K_{12}$ *and that is strong evidence against the random labelling hypothesis.*

Waller and Gotway suggest that random thinning (see page 144 in their text) can be used to test for random labelling. I tried this by randomly choosing 65 *locations from the total of* 135 *and fit L functions to those points. I did this* 99 *times and compared the minimum and maximum values of the L functions at the values of h the L function computed using all the data. The R code and a plot are shown below. I did not use the modified L function as it was not as clear (that strange behavior near the origin really confused things).*

```
# compute the L function for all the data
Lfun<-Lest(betacells,correction="iso")
# for convenience pull out the 513 radii
r<-Lfun$r
```

```
# set up a storage matrix and put the observed L function in the
# first column
sim.mat<-matrix(0,nrow=513,ncol=100)
sim.mat[,1]<-Lfun$iso
# random thinning
for(i in 2:100){
betacells.rand<-betacells[sample(1:135,65,replace=F),]
sim.mat[,i]<-Lest(betacells.rand,correction="iso")$iso
  }
Lmax<-apply(sim.mat,1,max)
Lmin<-apply(sim.mat,1,min)
plot(Lfun)
lines(r,Lmax,lty=2)
lines(r,Lmin,lty=2)
```

The plot is shown below. It appears from this that the evidence is consistent for random thinning but in this case it is not consistent for random labelling. It is important to note that they did not even consider independence as a possibility for the graves data set and with that the comparison of the two component processes would give them some insight into random labelling. But in this case independence is possible and the equality of the two component processes by itself is not enough.



As always subject matter expertise is needed. But it certainly appears that the data are consistent with a hypothesis of two similar processes operating independently of one another. A homework assignment on a similar data set will hopefully make some of the above a bit clearer. Be sure to read over the example involving medieval grave sites in Waller and Gotway (pages 141-146). We will look at these results in a bit more detail in a handout in class.

- A homogeneous spatial Poisson process has a constant intensity, $\lambda(\mathbf{s}) = \lambda$ for all $\mathbf{s}$. The null hypothesis of CSR may be rejected if clustering of events is observed. However, clustering may be due to any number of reasons. Perhaps events are not independent of one another or perhaps they are occurring independently of one another but the intensity varies spatially.

- A spatially varying intensity function is characteristic of a *heterogeneous* spatial point process. A heterogeneous Poisson process is defined by

  1. the number of events $N(A)$ in a region A is a random variable following a Poisson distribution with mean

     $$\mu(A) = \int_A \lambda(\mathbf{s}) d\mathbf{s}$$

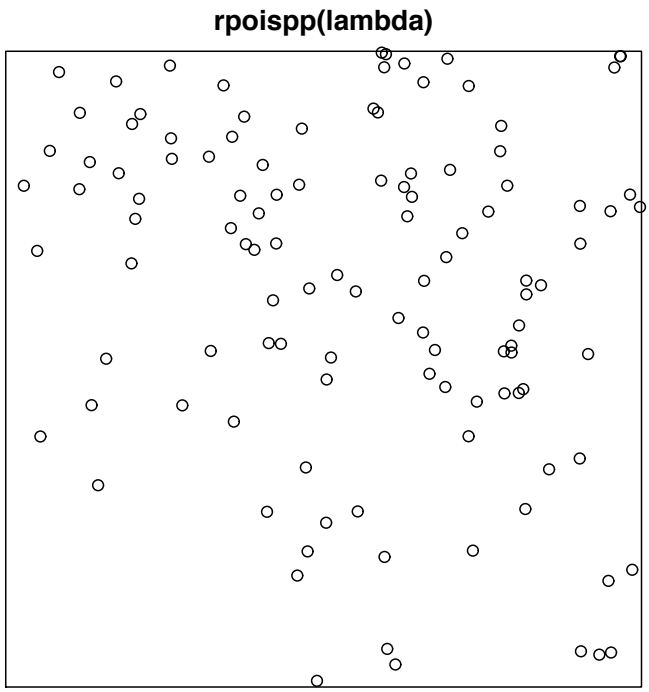
     and

  2. given the total number of events $N(A) = n$ in A, the number of events is an independent random sample of n locations, with the probability of sampling a particular point $\mathbf{s}$ proportional to $\lambda(\mathbf{s})$.

  The numbers of events in nonoverlapping regions A_1 and A_2, say, are still independent Poisson random variables.

- Here is a simple example of a realization of a heterogeneous Poisson process in R.

```
lambda<-function(x,y){
 100*(x+y)}
plot(rpoispp(lambda))
```



The mean of the process is

$$\int_0^1 \int_0^1 100(x+y)dxdy = 100.$$

- In epidemiology this is often referred to as the *constant risk hypothesis*. Each individual in the population at risk has the same probability of contracting some disease but disease counts will cluster in regions of higher density. These clusters would represent a violation of a CSR hypothesis but not necessarily a violation of the constant risk hypothesis. The goal in epidemiology is to find clustering over and above what is expected based on a null hypothesis of constant risk, i.e. additional clustering after controlling for spatial variations in density.

- *Estimating Intensity Functions*: The most common method of estimating spatially varying intensity functions is *kernel density estimation*. Intensity functions are not density functions but they are proportional to density functions as we will see below.

Although it is not obvious, statistical estimation as taught in most statistics courses can be motivated as density estimation. A general form or family is assumed (e.g. a normal distribution with unknown mean and variance), data are collected and estimates of the mean and variance are calculated. The result is an estimate of the true normal density, i.e. a particular member of the family of all normal distributions. This approach could be called parametric density estimation. We will focus instead on nonparametric kernel density estimation. You are already familiar with one such method - density histograms. We will briefly look at kernel density estimation, starting with a single variable and extending the results to two variables.

The basic idea is really quite simple. Observations have been taken from the underlying unknown density. The estimate of the density f at an arbitrary value x_0 is determined by taking a weighted average of the observed points, $x_1, x_2, \cdots, x_n$. The closer an observation is to x_0 the more weight it receives. A suitable weight function (kernel) needs to be specified. Generally it is desired that the estimated density be a mathematically valid density function and this requirement places certain restrictions on the kernel. In particular, the kernel should be a pdf itself. The kernel density estimator of f at x_0 is

$$\widehat{f}(x_0) = \frac{1}{nh} \sum_{i=1}^{n} k\left(\frac{x_0 - x_i}{h}\right)$$

where n is the sample size, $x_1, x_2, \cdots, x_n$ are the observed data, k is the designated kernel, and h is the bandwidth or smoothing parameter. Silverman (1986, Section 2.3) provides a nice discussion of how kernel estimation works. Briefly, the estimated density at an arbitrary value x_0 is constructed by centering the kernel over the n observations , yielding a series of "bumps". The choice of kernel determines the shape of the bump and the choice of h determines the width of the bumps. The larger the bandwidth the wider the bump and the smoother the estimates. If k is a pdf then $\widehat{f}$ will be a pdf.

Choice of Kernel k: There are many possible candidates for a kernel. Fortunately, the final estimate $\widehat{f}$ is, in general, robust to the choice of k. By far the most common choice is the Gaussian kernel:

$$k(t) = \frac{1}{\sqrt{2\pi}} \exp\left(-t^2/2\right).$$

There are other popular choices but the default in almost all software packages will be the Gaussian kernel.

Choice of Bandwidth h: This is a critical choice. Choosing a bandwidth that is too small will result in a noisy estimate with lots of spurious structure. Choosing

a bandwidth that is too large will result in an oversmoothed estimate for which important structure cannot be discerned. Most work in kernel density estimation over the past few years has focused on finding objective methods of determining bandwidth. The methods have generally been motivated by finding bandwidths that are optimal in the sense of providing estimates $\widehat{f}$ that are "close" to the true underlying f. A favored approach is to find h to minimize the Mean Integrated Squared Error ($MISE$):

$$MISE = \int E\left(\widehat{f}\,(x,h) - f(x)\right)^2 dx.$$

It would seem that choice of h requires knowledge of f and some methods make this assumption. For example, if f is assumed to be Gaussian with variance σ^2 and a Gaussian kernel is used the optimal bandwidth is

$$h_{opt} = 1.06\sigma n^{-1/5}.$$

A more robust estimator is

$$h_{opt} = 0.9An^{-1/5}$$

where

$$A = \min\left(\sigma, IQR/1.34\right)$$

and IQR is the Interquartile Range.

A commonly applied method that does not require specification of f is Least Squares Cross Validation ($LSCV$). We expand $MISE$ to get

$$MISE\left(\widehat{f}(s)\right) = \int E\left(\widehat{f}(x)^2\right)dx - 2\int E\left(\widehat{f}(x)f(x)\right)dx + \int f(x)^2 dx.$$

The last term on the right does not affect the solution so we can ignore it. It can be shown that an unbiased estimator of the remaining terms is

$$LSCV(h) = \int \widehat{f}(x,h)^2 dx - \frac{2}{n}\sum_{i=1}^{n}\widehat{f}_{-i}\left(x_i,h\right)$$

where

$$\widehat{f}_{-i}\left(x_i,h\right) = \frac{1}{h(n-1)}\sum_{j\neq i} k\left(\frac{x-x_j}{h}\right).$$

This is the density estimate based on deleting the ith observation from the sample, hence the use of the term "cross-validation" to describe it. In practice it is not uncommon to find multiple local minima and typically plots of $LSCV$ versus h are used to find a solution. This basic idea has been modified and extended with varying results. Other methods motivated by different criteria have been developed. The literature is extensive and we will not go there. Despite all these "objective"

methods almost all researchers in this area urge users to visually assess the results, i.e. after all the effort to find an objective bandwidth the final decision still will be at least partly subjective.

The above ideas can be extended to two dimensions. We now have observed data in the form of coordinates: $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$. One simple method (using *product kernels*) is to estimate the density at an arbitrary location $\mathbf{s}_0 = (x_0, y_0)$ by

$$\widehat{f}(x_0, y_0) = \frac{1}{nh_1h_2} \sum_{i=1}^{n} \left\{ k\left(\frac{x_0 - x_i}{h_1}\right) k\left(\frac{y_0 - y_i}{h_2}\right) \right\}.$$

Under this notation the same kernel is used for each of the 2 variables but that is not necessary and may not be desirable in all cases. Selection of bandwidths is more important than selection of kernels and the determination of optimal bandwidths in multiple dimensions remains an open research problem. One suggestion for a starting point for an optimal bandwidth in this setting is

$$h_i = \sigma_i n^{-1/6}$$

for $i = 1, 2$. The standard deviation can be estimated by the sample standard deviations of the x and y coordinate values, respectively. Many software packages assume that $h_1 = h_2 = h$, at least as a default.

Edge effects can be substantial. The text gives one edge-corrected version. There are families of boundary kernels available that could also be used although there appears to be little in the way of readily available software to fit these.

Kernel density estimation as described above focuses on estimation of densities not intensities. However, the intensity is proportional to the density in that

$$f(\mathbf{s}) = \frac{\lambda(\mathbf{s})}{\mu(A)}.$$

Density estimates can be converted into intensity estimates:

$$\widehat{\lambda}(\mathbf{s}_0) = \frac{n\widehat{f}(\mathbf{s}_0)}{\nu(A)}$$

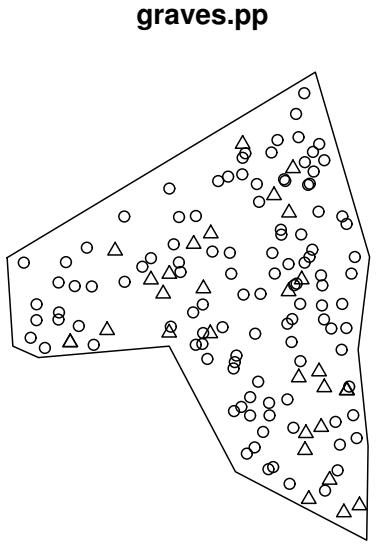
but many practitioners do not bother with this rescaling.

The base R software package contains density estimation functions and there are others in Ripley's `MASS` library. There are several packages devoted entirely to the topic (e.g. `sm`, `KernSmooth`). The `spatstat` library has a function `density.ppp` which produces kernel estimates of the intensity function in two dimensions. We will look at examples in class. Those of you in the animal ecology area may have seen kernel density homerange estimation software packages. These could actually be used for estimation of intensity functions.

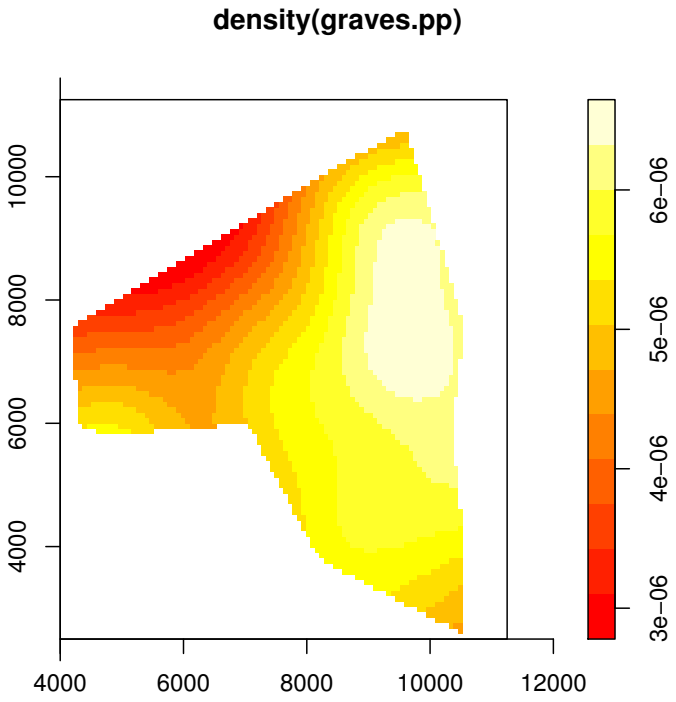
Example: This is an example using the graves data set we have seen on some handouts. This example is useful for a couple of reasons, one of which is that it allows us a chance to work with a nonrectangular region. I converted a text file of the grave locations using the ***ppp*** *function. First, I needed to specify the nonrectangular region using the* ***owin*** *function.*

```
graves<-read.table(file="../../../DataSets/graves.txt",header=F)
names(graves)<-c("id","marks","x","y")
polyg<-list(x=c(4200,4300,4750,7039,8200,10500,10524,10350,10550,9600),
            y=c(7550,6000,5800,6000,3800,2600,4250,5937,7562,10800))
w<-owin(poly=polyg,xrange=c(4000,11250),yrange=c(2500,11250))
graves.pp<-ppp(graves$x,graves$y,window=w,marks=as.factor(graves$marks))
```

A plot of the grave locations is shown below (produced by simply typing `plot(graves.pp)`*). This is a marked process. The locations marked by triangles are graves where a tooth defect was noted and the other sites are graves whose individuals were not affected by the defect.*

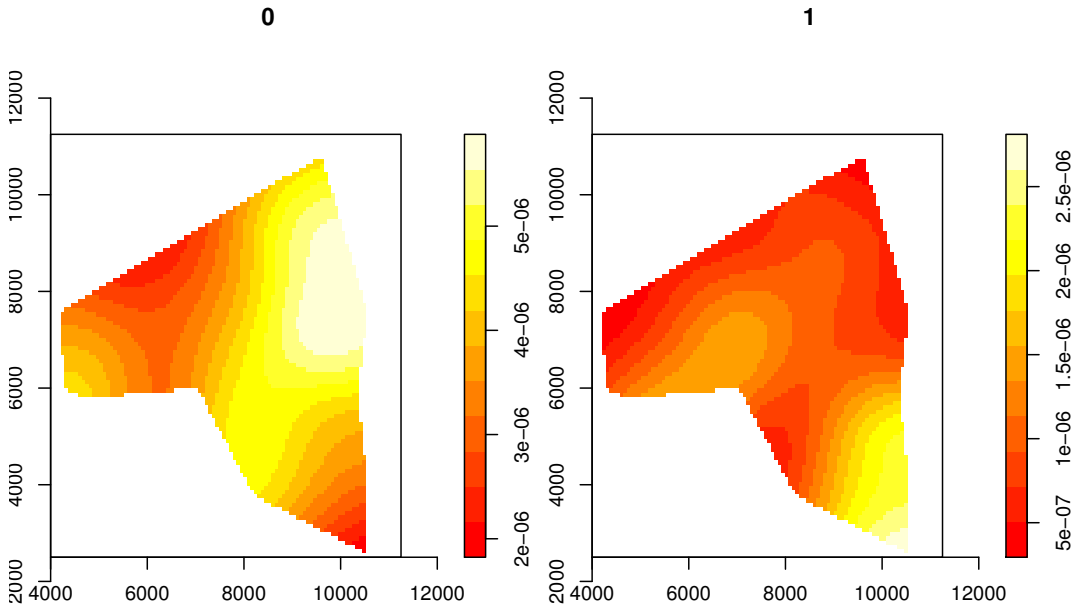


The `density` *function is used to produce plots of the intensity function. This function is actually a function available in R, but when it is applied to a point process object it defaults to* `spatstat`*'s* `density.ppp` *function. We can play around with the bandwidths but for this data set the default computations in the function work fine. A plot of the intensity function surface is shown below.*



It appears that the graves are more likely to occur along the western border of the study site. One question of interest is whether or not the affected sites and unaffected sites vary in their intesity functions. This is easily checked.

```
plot(density(split(graves.pp)))
```



It does appear that the peak intensity of the 2 processes occur at different spatial locations. However, our look at the 2 processes using K functions did not give us additional supporting evidence for different processes for the 2 types of grave sites.

- *Ratio of Intensity Functions* We suppose we have n_1 locations of events of one type and n_2 locations of events of a second type. In epidemiological settings these will generally denote the locations of cases and controls. We let $\lambda_1(\mathbf{s})$ and $\lambda_2(\mathbf{s})$ denote the 2 intensity functions. The log ratio of the 2 intensity functions

$$r(\mathbf{s}) = \log\left\{\frac{\lambda_1(\mathbf{s})}{\lambda_2(\mathbf{s})}\right\}$$

provides information on local clustering of cases relative to controls. Recalling that the intensity functions are proportional to the density functions f_1 and f_2 we can rewrite the above as

$$r(\mathbf{s}) = \log\left\{\frac{f_1(\mathbf{s})}{f_2(\mathbf{s})}\right\}.$$

In this form it is known as the *log relative risk*, i.e. the natural log of the ratio of the "probability" of a case at location $\mathbf{s}$ to the "probability" of a control at location $\mathbf{s}$.

- Density estimates of f_1 and f_2 can be used to estimate r. There are some added problems here, especially as regards selection of bandwidth. Optimal bandwidths for estimation of f_1 may not be optimal for estimation of f_2. Bandwidths optimal for estimation of f_1 and f_2 may not be optimal for estimation of r. Choice of a finite tail kernel can yield $\widehat{f}_2$ values of 0 at some locations $\mathbf{s}$ with obvious problems for estimation of r at those locations. Typically density estimation software produces density estimates at grid points in a selected study area. In fact it may not even be possible to get density estimates at the observed event locations.

- Log relative risk ratios theoretically allow for determination of the location of clusters of cases (and controls) as well as determination of overall clustering.

    1. Local clustering: We estimate r using the original data. We condition on the number of cases and controls. We then randomly relabel cases and controls a large number of times and estimate r each time using the bandwidth determined from estimation of r with the original data. We can construct histograms and Monte Carlo p-values for each grid point. Local clusters of increased risk may be highlighted by small upper tail p-values and clusters of reduced risk may be highlighted by small lower tail p-values. It must be kept in mind though that these are pointwise tests and one may be working on a very large grid. The default grid set up by the `density` function in `spatstat` has 10000 grid points.
    2. Global clustering: The global null hypothesis is

    $$H_0 : r(\mathbf{s}) = 0, \quad \text{for all} \quad \mathbf{s} \in D$$

    One statistic suggested for a test of this hypothesis is

    $$\int_D \widehat{r}(\mathbf{s})^2 d\mathbf{s}$$

    The integration cannot be carried out in closed form of course but the test statistic can be approximated by calculating the sum of the estimated volume under the surface for each grid cell ($\widehat{f}(\mathbf{s})$ (area of grid cell)). The grid cells will typically all be of the same size so for Monte Carlo test procedures it is enough to consider the test statistic

    $$T = \sum \widehat{r}(\mathbf{s})^2$$

    The value of T for the observed data is computed followed by computation of values subsequent to random relabeling of the marks (cases and controls). This can take a long time if the number of simulations in large.

- Technically we should be assessing a null hypothesis of constant (relative) risk but the tests are generally based on random relabeling for convenience. An example based on the Medieval grave site data will be shown in class.

- Most of what follows is from a set of notes on a course presented by Adrian Baddeley in Australia which he made available on the web. We are assuming a heterogeneous Poisson process. It can be shown that the log likelihood for such a process with spatially varying intensity $\lambda(\mathbf{s})$ is

$$L(\lambda) = \sum_{i=1}^{n} \log \lambda(\mathbf{s}_i) - \int_A \lambda(\mathbf{s})\, d\mathbf{s}.$$

  A "natural" model for the intensity is the log-linear model

$$\log \lambda(\mathbf{s}) = \sum_{i=1}^{n} \beta_i x_i(\mathbf{s}).$$

  Under this assumption the log-likelihood should be relatively well-behaved.

- The `ppm` function in the `spatstat` package will fit such models. The function was designed to be similar (in spirit at least) to the `lm` and `glm` functions in R. We can fit models with trend surfaces using

```
ppm(X,~trend)
```

  The log-linear model for the intensity is the default so that `trend` is the natural log of the intensity function. We will look at a relatively simple example involving the `bei` data set in the `spatstat` package. The data are the locations of 3605 trees in a tropical rain forest. Also, available is covariate information on elevation and slope.

  A simple trend model is fit as

```
bei.fit1<-ppm(bei,~x+y)
summary(bei.fit1)
Point process model
fitted by maximum pseudolikelihood (Berman-Turner approximation)
Call:
ppm(Q = bei, trend = ~(x + y), correction = "isotropic")
Edge correction: isotropic
--------------------------------------------------
Quadrature scheme = data + dummy + weights
Data pattern:
Planar point pattern: 3604 points
Average intensity 0.00721 points per square metre
```

```
Window: rectangle = [0, 1000] x [0, 500] metres
Window area =  5e+05 square metres
Unit of length: 1 metre


Dummy quadrature points:
( 130 x 130 grid, plus 4 corner points)
Planar point pattern: 16904 points
Average intensity 0.0338 points per square metre

Window: rectangle = [0, 1000] x [0, 500] metres
Window area =  5e+05 square metres
Unit of length: 1 metre


Quadrature weights:
(counting weights based on 130 x 130 array of rectangular tiles)
All weights:
        range: [1.64, 29.6]     total: 5e+05
Weights on data points:
        range: [1.64, 14.8]     total: 41000
Weights on dummy points:
        range: [1.64, 29.6]     total: 459000

--------------------------------------------------
FITTED MODEL:

Nonstationary Poisson process


 ---- Intensity: ----

Trend formula: ~(x + y)

Fitted coefficients for trend formula:
  (Intercept)             x             y
-4.7245290274 -0.0008031288  0.0006496090


----------- gory details -----

Fitted regular parameters (theta):
```

```
  (Intercept)             x             y
-4.7245290274 -0.0008031288  0.0006496090

Fitted exp(theta):
(Intercept)           x           y
0.008874893 0.999197194 1.000649820
```

The fitted intensity is

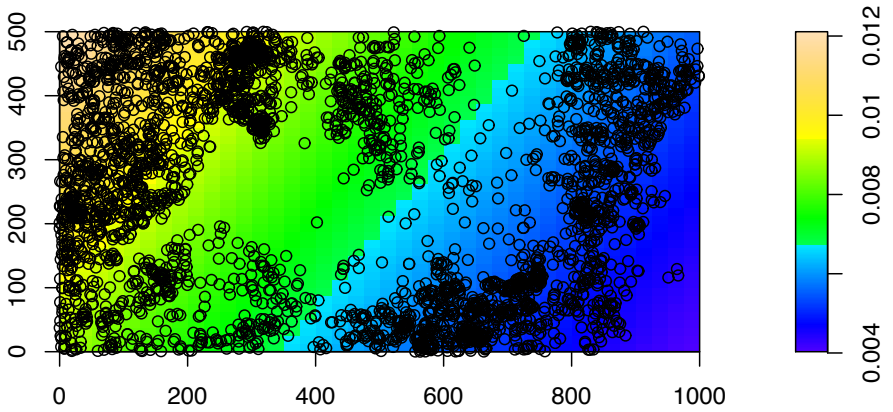
$$\widehat{\lambda}(\mathbf{s}) = \exp(-4.72 - 0.0008x + 0.00065y)$$

where (x, y) are spatial coordinates.

A plot of the fitted surface is shown below.

```
plot(bei.fit1,how="image")
```

This function actually produces several plots. The image version with the superimposed points is shown below.



We can fit a polynomial trend model involving quadratic terms for x and y along with an interaction xy using the `polynom` function. The model for the intensity is

$$\log \lambda(\mathbf{s}) = \beta_0 + \beta_1 x + \beta_2 y + \beta_3 x^2 + \beta_4 y^2 + \beta_5 xy.$$

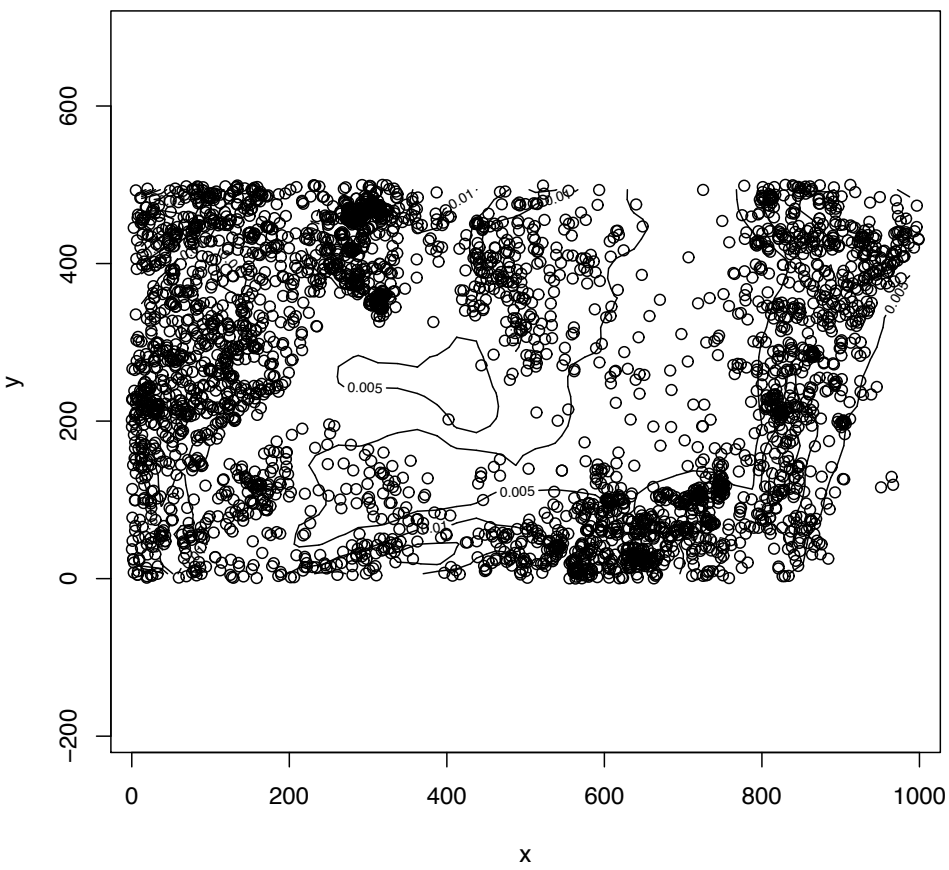
```
bei.fit2<-ppm(bei,~polynom(x,y,2),correction="isotropic")
bei.fit2
Nonstationary Poisson process

Trend formula: ~polynom(x, y, 2)

Fitted coefficients for trend formula:
          (Intercept)   polynom(x, y, 2)[x]   polynom(x, y, 2)[y]
        -4.275762e+00         -1.609187e-03         -4.895166e-03
polynom(x, y, 2)[x^2] polynom(x, y, 2)[x.y] polynom(x, y, 2)[y^2]
         1.625968e-06         -2.836387e-06          1.331331e-05
```

We can incorporate spatial covariates as follows. The `bei` data contain information on elevation and slope. These are images defined on the study area with values of the covariates at each of the tree locations.

```
grad <- bei.extra$grad
bei.fit3<-ppm(bei, ~slope, covariates = list(slope = grad),correction="isotropic")
bei.fit3
plot(bei.fit3,how="contour")
```

The results are

```
bei.fit3
Nonstationary Poisson process

Trend formula: ~slope

Fitted coefficients for trend formula:
(Intercept)       slope
  -5.390553    5.022021
```



The contour plot shows the results a bit clearer than the image plot in this case.

We can fit models with both trend and spatial covariates.

```
bei.fit4<-ppm(bei,~x+y+slope,covariates=list(slope=grad),correction="isotropic")
```

We can summarize the results of fitted models using the standard R functions. The extractor functions `fitted`, `predict`, `coef`, and `vcov` all work as expected. We can use `vcov` to get standard errors.

For example, the variance covariance matrix associated with the coefficients in the trend plus slope model are

```
vcov(bei.fit4)
             (Intercept)             x             y         slope
(Intercept)  2.457074e-03 -1.006351e-06 -3.888432e-06 -7.060301e-03
```

```
x           -1.006351e-06  3.225037e-09 -1.515090e-10 -3.465928e-06
y           -3.888432e-06 -1.515090e-10  1.242177e-08  6.691385e-06
slope       -7.060301e-03 -3.465928e-06  6.691385e-06  6.683144e-02
```

The standard errors are the square roots of the diagonal elements of this matrix.

```
sqrt(diag(vcov(bei.fit4)))
 (Intercept)            x            y        slope
4.956888e-02 5.678941e-05 1.114530e-04 2.585178e-01
```

Model comparison tools are also available. For nested models an analysis of deviance approach works. One obvious null model is a model of CSR. This null can be fit as follows.

```
bei.null<-ppm(bei,~1,correction="isotropic")
```

We can test any of the other models against this null using an analysis of deviance.

```
anova(bei.null,bei.fit3,test="Chi")
Analysis of Deviance Table

Model 1: .mpl.Y ~ 1
Model 2: .mpl.Y ~ slope
  Resid. Df Resid. Dev    Df Deviance P(>|Chi|)
1     20507    18728.4
2     20506    18346.1     1    382.3 4.018e-85
```

The p-value is 0 to round-off in R providing strong evidence against the null model, i.e. our proposed model fits a lot better than the null (although it still may not fit very well).

It is often the case that we wish to compare non-nested models. The analysis of deviance approach will not work in that case but there are other tools we can use. A popular tool currently used a lot in ecological applications is the Akaike Information Criterion (AIC).

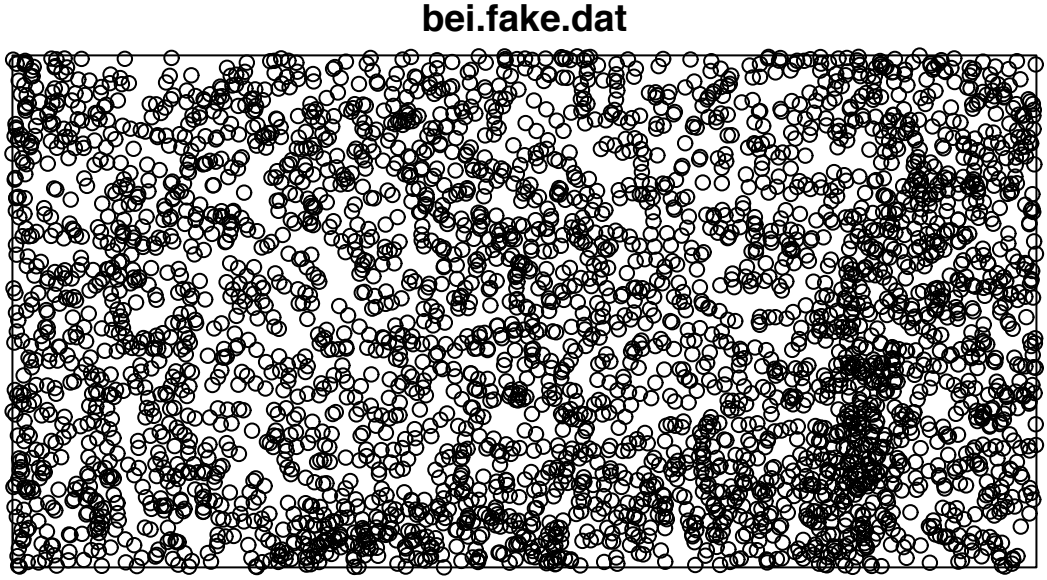
```
AIC(bei.null)
[1] 42763.92
AIC(bei.fit3)
[1] 42383.66
```

Lower values of AIC are better and these results suggest that the model containing slope fits better than the null model of CSR (although again the slope model may still not fit very well).

We can simulate data from a fitted model using the `rmh` function.

```
bei.fake.dat<-rmh(bei.fit3)
plot(bei.fake.dat)
```

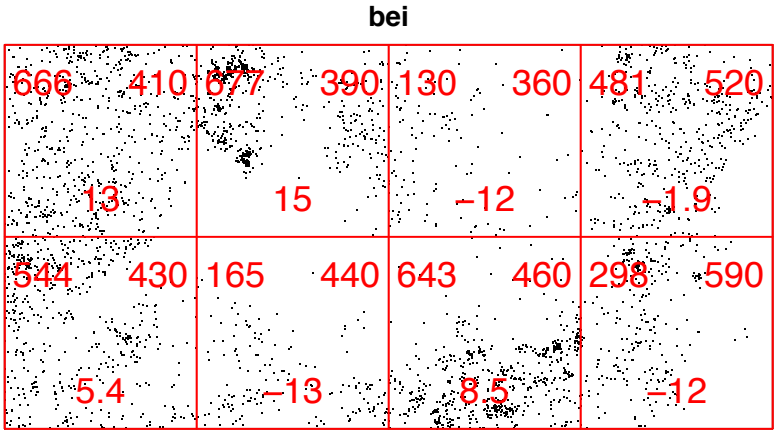
The plot (seen at the top of the next page) doesn't seem to match the observed pattern very well. We could actually use this function to carry out Monte Carlo tests of how good a particular model is. For example, we could compare G functions or nearest neighbor distances from our observed data set with values from simulated data sets to see how good they match up. There are other model fit diagnostics available in `spatstat`, however.



The `quadrat.test` function which we saw earlier can be used as a goodness-of-fit test for fitted model objects. The test compares the observed pattern with the fitted pattern generated by the model.

```
M <- quadrat.test(bei.fit3, nx = 4, ny = 2)
M
```

```
        Chi-squared test of fitted model bei.fit3 using quadrat counts

data:  data from bei.fit3
X-squared = 946.5998, df = 6, p-value < 2.2e-16

plot(bei,pch=".")
plot(M, add = TRUE, cex = 1.5, col = "red")
```



The null hypothesis is that the actual data were generated by the fitted model. We have very strong evidence against this, i.e. our model does not fit very well. The

plot below shows that it misses in quite a few places. In each quadrat the value in the upper left corner is the observed count, the value in the upper right is the predicted count and the value at the bottom is a Pearson residual

$$e = \frac{O - E}{\sqrt{E}}$$

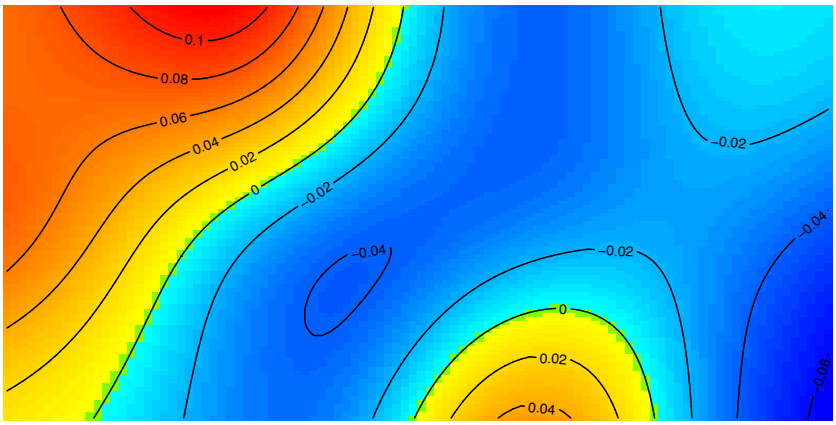
where O and E are the observed and expected counts, respectively. If the model is true then these residuals have mean 0 and standard deviation 1 and we see many quadrats that are obviously not consistent with this.

We can also examine spatial plots of residuals. The plot above is a crude way of doing this. R has other options, however. One potentially useful plot is a smoothed residual plot created using the `diagnose.ppm` function. The plot is actually an image of what Baddeley refers to as a smoothed residual field defined as

$$e(\mathbf{s}) = \widehat{\lambda}(\mathbf{s}) - \lambda^{+}(\mathbf{s})$$

where $\widehat{\lambda}$ is a nonparametric kernel density based estimate of the intensity and λ^{+} is a smoothed estimate of the intensity function from the fitted model.

```
diagnose.ppm(bei.fit3,which="smooth")
```



This should be a noisy plot with no obvious spatial trend and with about as many positive as negative residuals. We seem to have obvious spatial trends here. The plot shows where the lack of fit is worst. It suggests that we are missing other important variables. We might try to clean it up by adding additional trend terms or other spatial covariates.

- We have already seen examples of marked point processes when the marks are binary factors. There are 2 other classes of marked processes discussed in the text.

    1. Multivariate patterns which are just an extension of the binary case to multiple categories. For example, we might have a data set containing locations of trees of several different species. There do not appear to be true multivariate methods for analyzing multivariate patterns. Typically, the approach is to examine pairs of the factors. (There is one function in R `Kdot` ($K_{i\cdot}$) which estimates "the multitype K function which counts the expected number of other points of the process within a given distance of a point of type i", where the expectation is of points of all other types.)
    2. Continuous marked patterns in which the marks associated with each event location is a continuous variable. An example here might be locations of trees with associated measurements on each tree such as age or diameter at breast height (dbh). Our text does not deal with analysis of these types of processes at all, at least in this chapter. One approach suggested in the `spatstat` documentation is to break the interval of continuous values into categories (e.g. High and Low) and work with the resulting binary marked process. However, there are 2 simple functions available in R that compute "correlations" and we will take a quick look at those.

- *"Correlation" Analysis of Marked Point Patterns*: An intuitively simple way to examine "clustering" of marks is to compute the nearest neighbor correlation, i.e. the correlation between the mark (i.e. value) associated with a particular event and that of its nearest neighbor. The R function `nncorr` in the `spatstat` library will do just this. The function also works with categorical marks but returns Pearson's correlation only for continuous marks. The function returns 3 different "correlations". All 3 work for continuous marks but only 2 for categorical marks. The 3 measures are

    1. an unnormalized nearest neighbor correlation (Stoyan and Stoyan, 1994, section 14.7) is defined as

    $$\overline{n}(f) = E\left[f\left(M, M^*\right)\right]$$

    where E denotes mean value, M is the mark attached to a typical point of the point process, and M^* is the mark attached to its nearest neighbor (i.e. the nearest other point of the point process).

    Here f is any function $f(m_1, m_2)$ with two arguments which are possible marks of the pattern, and which returns a nonnegative real value. Common choices of f are: for continuous real-valued marks,

$$f(m_1, m_2) = m_1 m_2$$

and for discrete (works also with categorical) marks (multitype point patterns),

$$f(m_1, m_2) = (m_1 == m_2).$$

In the second case, the unnormalized nearest neighbor correlation $\overline{n}(f)$ equals the proportion of points in the pattern which have the same mark as their nearest neighbor.

2. a normalized nearest neighbor correlation defined by

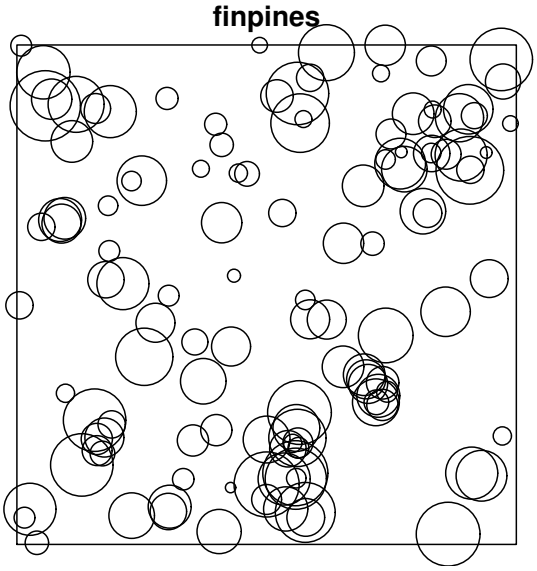
$$\overline{m}(f) = \frac{E[f(M, M^*)]}{E[f(M, M')]}$$

where again M is the mark attached to a typical point, M^* is the mark attached to its nearest neighbor, and M' is an independent copy of M with the same distribution. I am not quite sure what is meant by *an independent copy of M* and our library does not have the cited reference. I assume that M' is just the mark associated with a randomly selected point from the process.

3. the Pearson correlation (only computed for continuous marks).

The unnormalized and normalized versions are not "correlations" in the usual statistical sense which raises the question of how to interpret them. The unnormalized statistic can be greater than 1. The normalization results in a measure where values near 1 suggest "lack of correlation": if the marks attached to the points are independent and identically distributed, then $\overline{m}(f) = 1$. The interpretation of values larger or smaller than 1 depends on the choice of function f.

Example: One of the data sets provided with `spatstat` *is a point pattern of tree locations and associated heights (apparently in meters). I modifed the data set a bit to pull out only the mark I was interested in which was height. A plot is shown below.*



Larger diameter circles indicate taller trees. The distribution of heights is an added item of interest.

```
nncorr(finpines)
unnormalised    normalised    correlation
   7.9618981    0.9954150     -0.1839798
```

There appears to be a slight negative correlation between heights of neighboring trees. The default f in the `nncorr` *function is $f(m_1, m_2) = m_1 m_2$. Note that with this definition of f the normalized value $\overline{m} = 0.995$ suggesting a lack of correlation. A randomization test could be used to determine if the negative Pearson correlation of -0.184 is significantly different from 0. It probably is not.*

We can also use the function on categorical data.

```
nncorr(betacells)
unnormalised   normalised
          NA           NA
Warning messages:
1: * not meaningful for factors in:
```

```
Ops.factor(m1, m2)
2: * not meaningful for factors in:
Ops.factor(m1, m2)
nncorr(betacells,function(m1,m2) {m1==m2})
 unnormalised  normalised
  0.03636364  0.07262765
```

Note that when I attempted to use `nncorr` *on the categorical marks "off" and "on" in the* `betacells` *data set I got a warning message and unhelpful output. The default f does not work in this case. I need to redefine f as indicated. We see that only about 0.04 of the nearest neighbors have the same mark and the normalized version is not near 1 suggesting a lack of independence in the locations of the marks. Again a randomization test would be worth doing.*

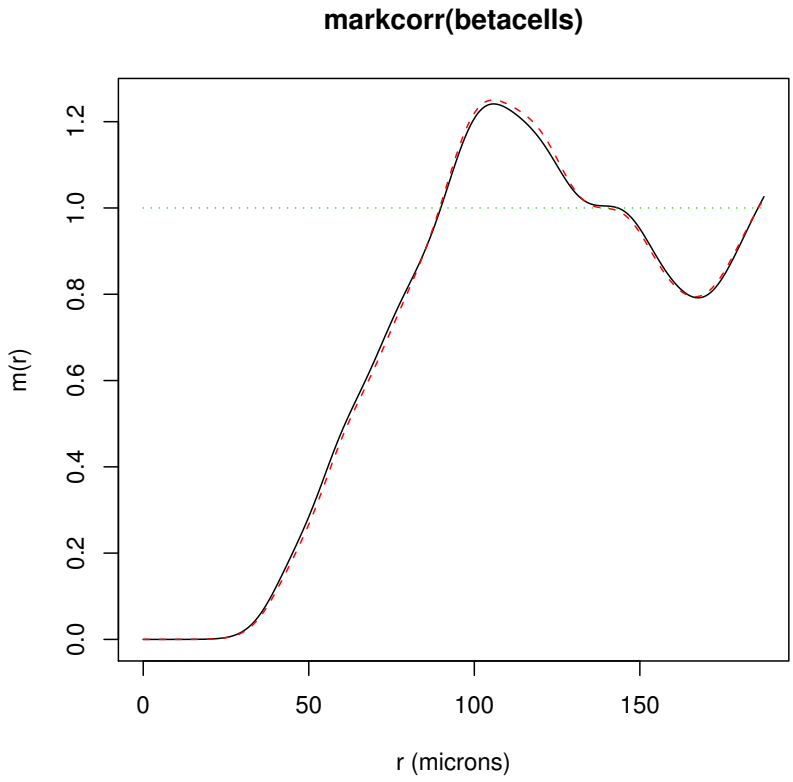
A second statistic available in R is `markcorr`. The mark correlation function $\rho_{f(h)}$ of a marked point process X is a measure of the dependence between the marks of two points of the process a distance h apart. It is informally defined as

$$\rho_{f(h)} = E\left[f\left(M_1, M_2\right)\right] / E\left[f\left(M, M^{'}\right)\right]$$

where E denotes expectation and M_1, M_2 are the marks attached to two points of the process separated by a distance h, while $M, M^{'}$ are independent realizations of the marginal distribution of marks. Again, I take this to mean that M and $M^{'}$ in the denominator are marks associated with randomly chosen points. The function f is defined as above in the description of `nncorr`. $\rho_{f(h)} \geq 0$ and will take on the value of 1 if the marks attached to points are independently and identically distributed. The description of the function notes that the interpretation of values above or below 1 depend on f. One key assumption to note is that the results are valid under an assumption of a homogeneous process which may limit its usefulness. The numerator and denominator of ρ are estimated using density estimation methodology. See the description of the function in R for details.

Example: For the `betacells` *data set the* `markcorr` *function results are*

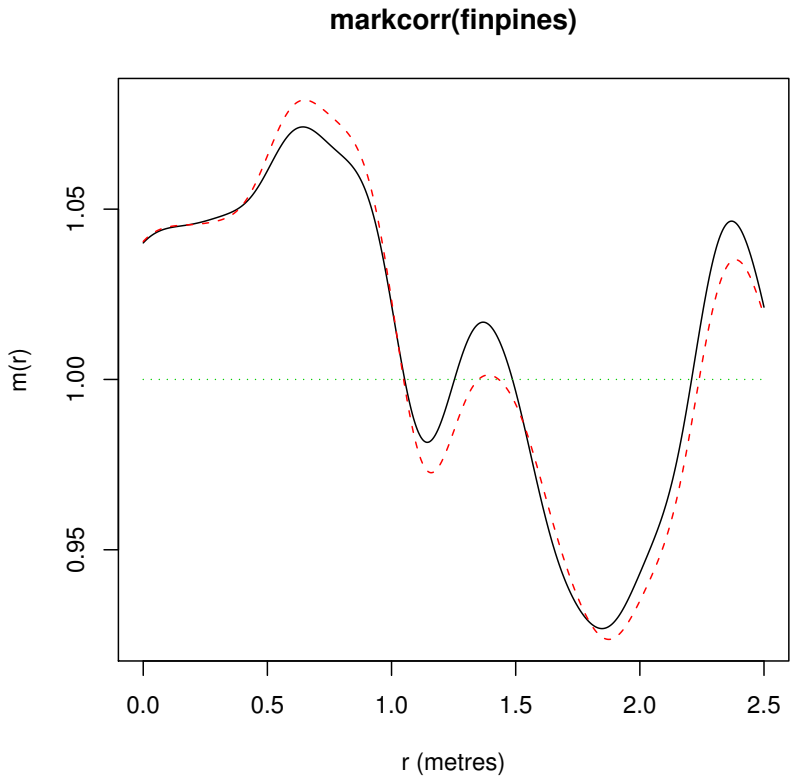
```
plot(markcorr(betacells))
     lty  col
iso    1    1
trans  2    2
theo   3    3
```



The horizontal line is the theoretical line under an assumption of independent and identically distributed values. The interpretation is a bit tough but recall that `nncorr` *indicated that only about 0.04 of the nearest neighbors had the same mark so it appears that neighbors 75 microns or less apart tend to have different marks.*

For the `finpines` *data set we get*

```
plot(markcorr(finpines))
```



Recalling that nearest neighbor heights were negatively correlated above we see indications of negative correlation up to 1 meter followed by possibly positive correlations. However, given the size of the correlation we saw in `nncorr` *it is not clear that these results are different from what would be expected by chance. Also, a note of caution: don't forget to keep an eye on the y-axis. The plot suggests a lot of variability in* $\rho_{f(h)}$ *but the values only range between 0.95 and 1.05. Changing the scale on the y-axis would give a very different picture.*

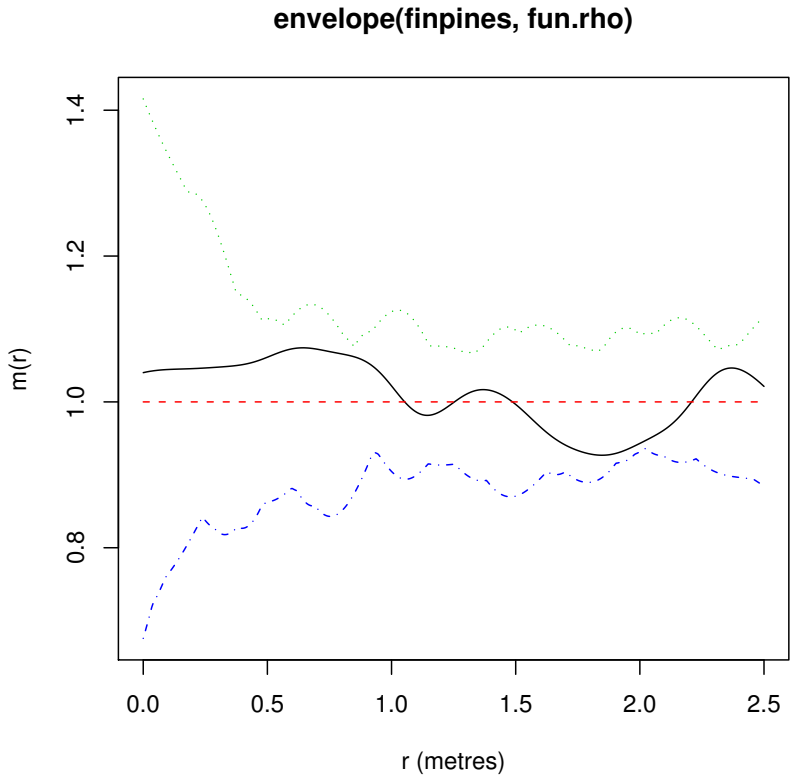
Obviously simulation envelopes would help. We can get those as follows. The description of the `envelope` *function helps. We can define our own function as described in the documentation:*

```
The statistic fun can also be a user-supplied function; if
so, then it must have arguments X and r like those in the
functions listed above, and it must return an object of class
"fv".
```

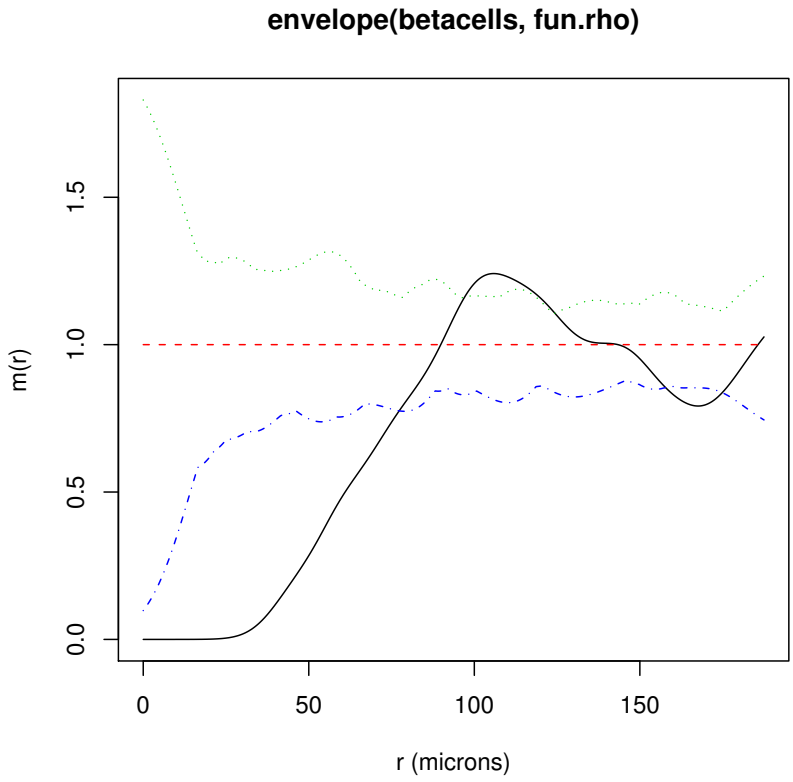
I defined a function that will work in the `envelope` *function as follows which produced the plot below.*

```
fun.rho<-function(X,r=NULL){markcorr(X)}
plot(envelope(finpines,fun.rho))
```

The plot certainly suggests that there is little evidence against the null of independent and identically distributed heights over the area of interest.



The envelope plot for the `betacells` *(below) suggests a real effect.*

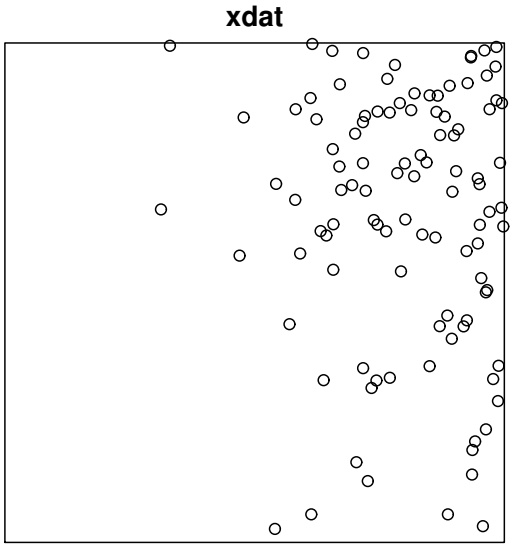


- *More on Heterogeneous Processes*: Recall from above that we defined a heterogeneous process to be one in which the intensity function varied over space. There are any number of ways to model such processes. A simple way is to assume that the intensity is a deterministic function of $\mathbf{s} = (x, y)$. Problem 3.7 on 130 gives us a simple example of such a process. We have a inhomogeneous Poisson process with a deterministic intensity function

$$\lambda(x, y) = \exp(5x + 2y).$$

The problem goes on and asks us to divide the unit square into a 4×4 grid and perform a Chi-square test for CSR based on observed quadrat counts. Here is how we might do that in R.

```
# generate the data
simdat<-rpoispp(function(x,y) exp(5*x + 2*y),1000)
plot(simdat)
```



Note that the data are most definitely not CSR. We can carry out the chi-square goodness-of-fit test described on page 91 in the text (the one based on the index of dispersion) as follows:

```
quadrat.test(simdat,nx=4)
        Chi-squared test of CSR using quadrat counts

data:  simdat X-squared = 162.4757, df = 15, p-value < 2.2e-16
```

Clearly we do not have CSR. The indication is that the points are clustered.

What is the overall mean of this process? If we denote the unit square on which we are working as the area of interest A then

$$\mu(A) = \int_0^1 \int_0^1 \exp(5x + 2y)\, dxdy = 94.18.$$

In the above example we had a deterministic intensity function. Another way of generating heterogeneous processes is to let $\lambda(\mathbf{s})$ be a random variable. Problem 3.6

in the text is a simple example. Suppose that we have 2 random variables X and Λ. Conditional on $\Lambda = \lambda$ we assume

$$X|\lambda \sim \text{Poi}(\lambda)$$

and we assume that $\Lambda \sim \text{Gamma}(\alpha, \beta)$ where

$$g(\lambda) = \frac{1}{\Gamma(\alpha)\beta^{\alpha}} \lambda^{\alpha-1} \exp\left(-\lambda/\beta\right).$$

Assuming that α is an integer, it can be shown that the marginal (unconditional) distribution of X is Negative Binomial with parameters α and $p = 1/(1+\beta)$. Note that $E(\Lambda) = \alpha\beta, \text{Var}(\Lambda) = \alpha\beta^2, E(X) = \alpha\beta, \text{Var}(X) = \alpha\beta(1+\beta)$. Note that a homogeneous Poisson process with intensity $\lambda = \alpha\beta$ has variance less than the heterogeneous process of Negative Binomial counts. This is the simplest example of what are known as Cox Processes.

Example: We will work in a simple setting of a gridded region with 100 equal size grids of unit area. Suppose that $\Lambda \sim$ *Gamma*$(2, 3)$*. I drew* 100 *realizations of* Λ

```
lambda.vec<-rgamma(100,shape=2,scale=3)
```

The counts in each grid were assumed to be Poisson with intensity equal to one of the 100 randomly chosen intensities.

```
grid.count<-rep(0,100)
for(i in 1:100) grid.count[i]<-rpois(1,lambda.vec[i])
```

Below is a table of the resulting counts. Within each grid we have a homogeneous Poisson process but each grid has a different intensity. This results in an obvious clustering of events.

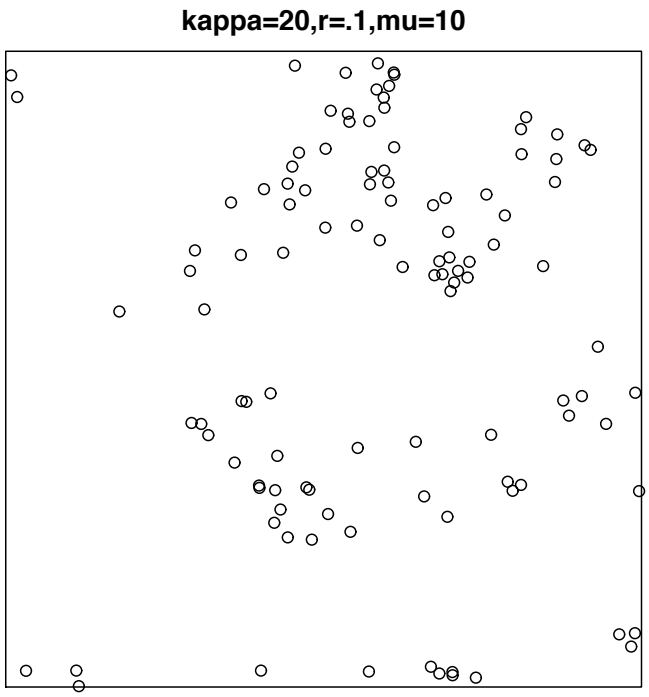
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
1	*4*	*3*	*11*	*2*	*4*	*10*	*10*	*2*	*4*	*5*
2	*6*	*0*	*7*	*9*	*11*	*6*	*4*	*9*	*10*	*11*
3	*5*	*7*	*4*	*1*	*8*	*3*	*10*	*7*	*11*	*1*
4	*1*	*9*	*2*	*2*	*5*	*0*	*4*	*3*	*8*	*17*
5	*2*	*10*	*4*	*4*	*18*	*8*	*6*	*5*	*2*	*2*
6	*2*	*8*	*6*	*11*	*5*	*6*	*11*	*3*	*15*	*4*
7	*3*	*6*	*20*	*1*	*11*	*0*	*5*	*4*	*1*	*2*
8	*2*	*6*	*6*	*6*	*3*	*1*	*13*	*4*	*6*	*4*
9	*4*	*9*	*11*	*6*	*12*	*1*	*13*	*15*	*8*	*5*
10	*13*	*4*	*4*	*5*	*10*	*3*	*3*	*8*	*4*	*7*

The mean of the counts is 6.17 *and the variance is* 17.68. *The counts are said to be "over-dispersed". A chi-square test of CSR leads to a very low p-value*

```
1-pchisq(99*17.68/6.17,99)
  0
```

There are other examples of Cox processes. The Matern process is a cluster process in which parent points are drawn from a homogeneous process with a constant intensity. Each of the parents then gives rise to offspring which are independently and uniformly distributed in a circle of specified radius around the parents, (which are removed). The number of offspring is assumed to also be Poisson. The `spatstat` function `rMatClust` simulates data from a Matern processes.

```
plot(rMatClust(kappa=20,r=5,mu=10))
```



We have a realization of a process with 20 parents, each centered in a circle of radius 0.1, generating an average of 10 children uniformly and independently distributed in the circle.

The Thomas process is similar except that the children are distributed around each parent according to a bivariate normal distribution. The function `rThomas` will generate data from this process.

Both of the above are examples of Neyman-Scott processes which are Cox processes with arbitrary clustering mechanisms. As already indicated the basic idea is simple. A point process produces event locations for parents. Each parent then spawns a number of offspring according to another process. The offspring event locations are determined by a function f. The parents are then removed and the result is a clustered process.

As mentioned in class K functions for other spatial processes than CSR can be derived. Neyman-Scott processes with homogeneous Poisson parent processes and

radially symmetric f's have

$$K(h) = \pi h^2 + \frac{E\left(N\left(N-1\right)\right)}{\lambda E\left(N\right)^2} F(h)$$

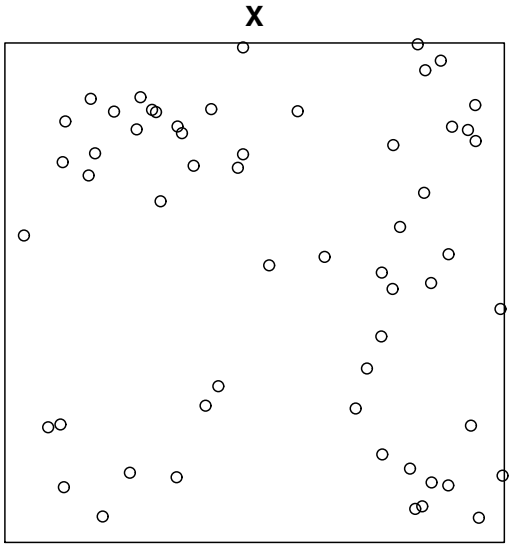
where h is the Euclidean distance between 2 arbitrary event locations in the same cluster, N is the number of points (offspring), and $F(h)$ is the cdf of h. For example, the Thomas model assumes:

1. The parent process is a homogeneous Poisson process with intensity λ.
2. Each parent produces N offspring with $N \sim \text{Poi}(\mu)$.
3. The position of the offspring relative to their parent is determined by bivariate Gaussian density
$$N\left(0, \sigma^2 I\right).$$

The resulting process has intensity $\lambda\mu$ and a K function equal to

$$K(h) = \pi h^2 + \frac{1}{\lambda}\left(1 - \exp\left\{-\frac{h^2}{4\sigma^2}\right\}\right).$$

Note that the K function does not depend on μ. The `spatstat` library has the capability of simulating and modeling data from Neyman-Scott processes. A typical realization of such a process is shown below. This is actually a realization of a Thomas process.



The `spatstat` library does not appear to have the ability to generate K functions for Neyman-Scott processes. But one can use empirical K functions to "fit" Neyman-Scott processes.

Diggle suggested using nonlinear least squares to estimate the parameters in the K function that holds assuming a given Neyman-Scott model and $\widehat{K}$, an estimate based on observed data. The objective function is

$$D(\theta) = \int_0^{h_0} \left\{ \left(\widehat{K}(h) \right)^c - (K(h;\theta))^c \right\}^2 dh$$

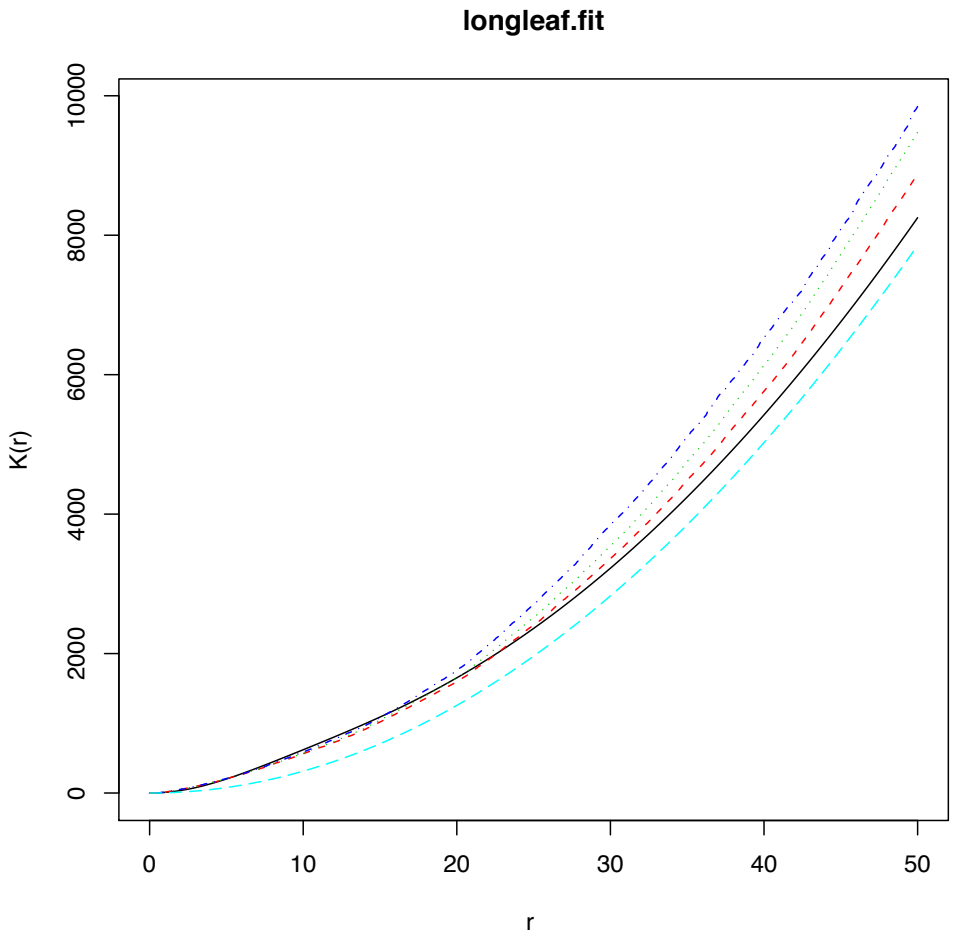
where c is a "tuning constant" and θ is vector of parameters to be estimated. This method is referred to as the method of minimum contrast.

We will look at an example from Cressie's text in class as applied to the longleaf pine data set.

There is a function in `spatstat` called `thomas.estK` which will "fit" the Thomas process K function by the minimum contrast method.

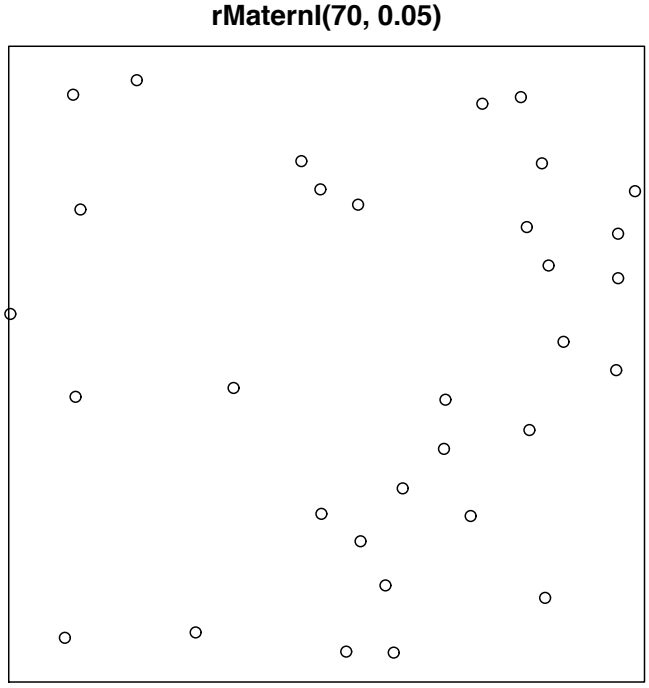
```
longleaf.fit
```

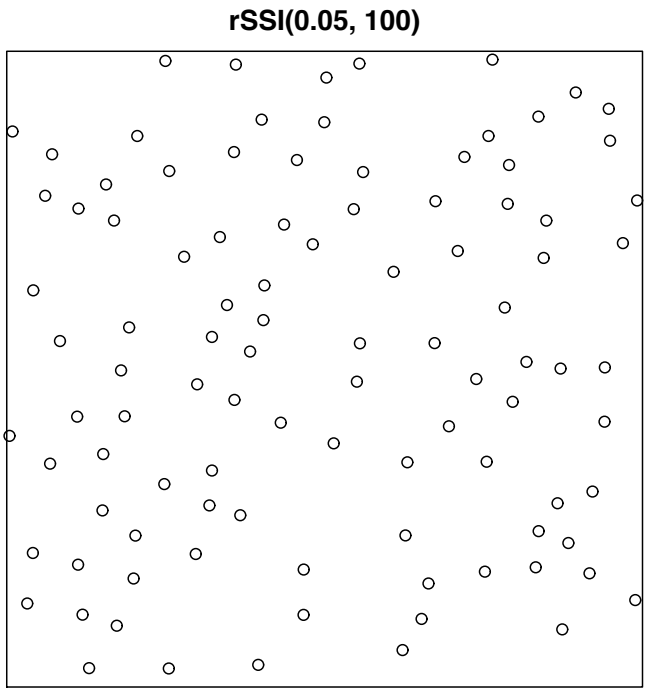
```
Minimum contrast fit (object of class minconfit)
Model: Thomas process
Fitted by matching theoretical K function to Kest(longleaf)
Parameters fitted by minimum contrast ($par):
       kappa       sigma2
 0.002518169 16.896668204
Derived parameters of Thomas process ($modelpar):
      kappa       sigma          mu
0.002518169 4.110555705 5.797863132
Converged successfully after 69 iterations.
Domain of integration: [ 0 , 50 ]
Exponents: p= 2, q= 0.25
plot(longleaf.fit)
       lty col
fit      1   1 Black
iso      2   2 Red
trans    3   3 Green
border   4   4 Dark Blue
theo     5   5 Light Blue
```



Regular processes can simulated using some kind of inhibition process. One approach is to simulate a process, say a CSR process and then thin it by deleting points that lie too close to one another. Another approach is to simulate points sequentially and independently of one another. A point that lies too close to an earlier generated point is deleted and another point is generated. The simulations end when it is not possible to add another point.

Below are plots of data simulated from the first approach using `rMatternI` function for the Type I Matern model and `rSSI` for the sequential inhibition process. You can look at the help pages for these functions to see how they work.


rMaternI(70, 0.05)



The `rSSI` function can produce very regular patterns with a bit of tweaking.

The `spatstat` package has a number of different functions for exploring and modeling nonhomogeneous processes. The function `Kinhom` will estimate the K function for a nonhomogeneous point process. Recall the brief mention of the `interaction` component of the `ppm` function. That component describes the stochastic dependence structure among the points. The default is `Poisson` which is what is assumed for nonhomogeneous Poisson processes. There are 16 other choices for other non-Poisson nonhomogeneous processes. If you are interested you will have to pursue this further on your own.