June_2

Hainan Xu

02/06/2022

MBP Feature extraction and analysis

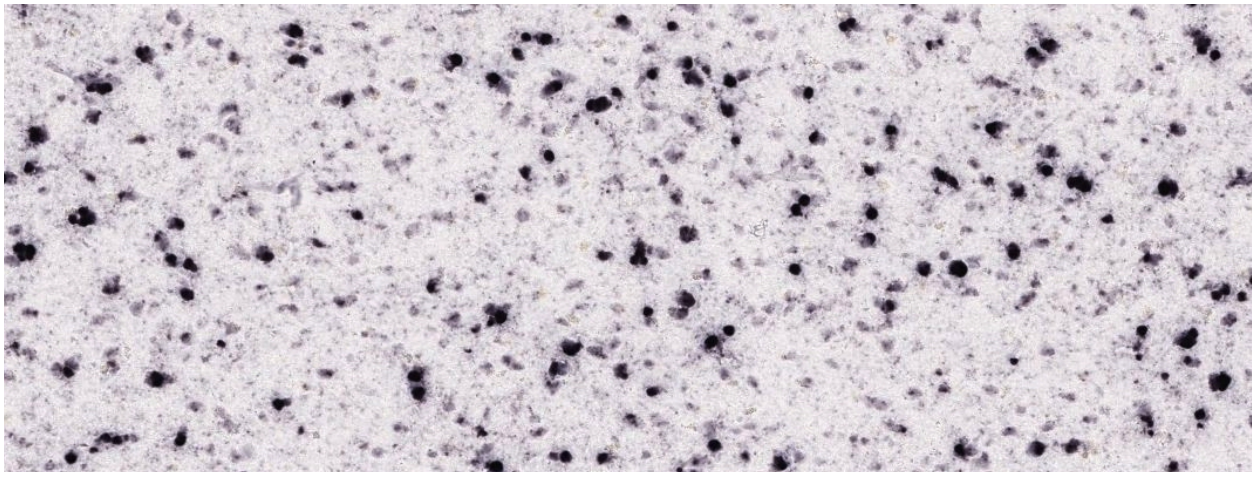
Here I used cropped picture(this is all I have) as an example. There are mainly 2 steps:
1. Threshold the image, then compute the features;
2. Use the features to compute intensity estimation and other graphs. ## Step1: Feature extraction

Load the package and previewthe image.

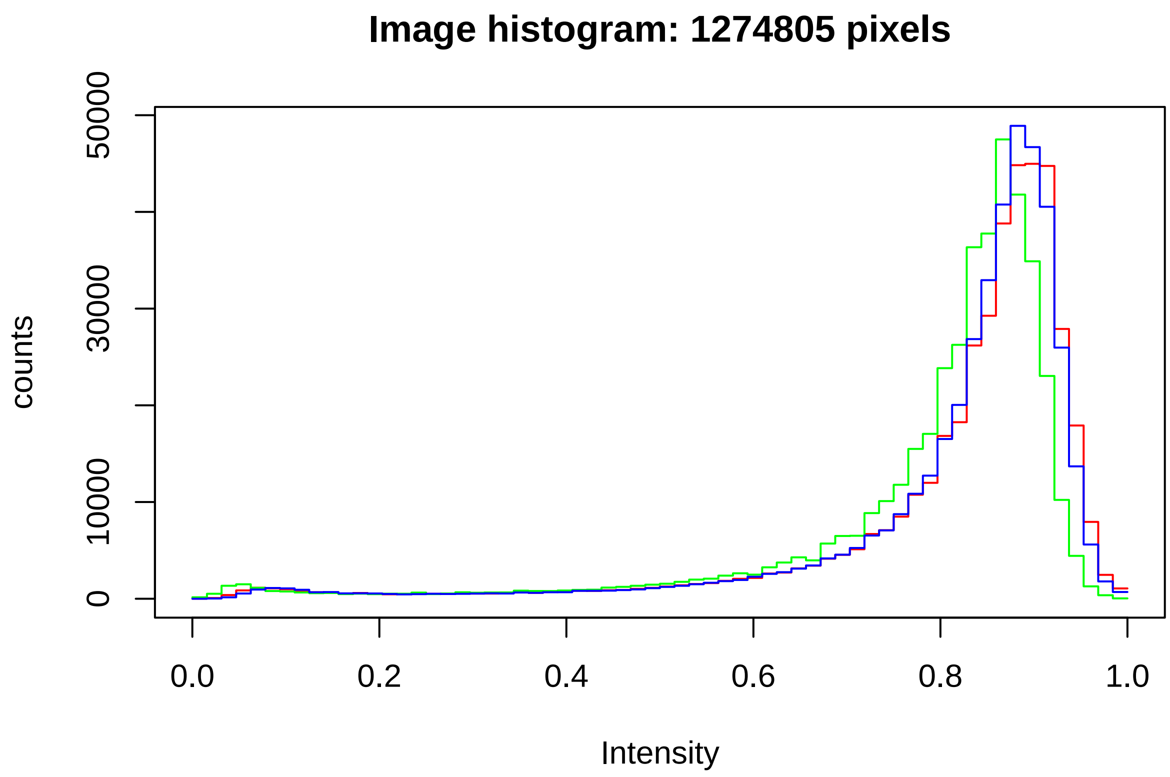
```
library(BiocManager)
```

```
## Bioconductor version '3.14' is out-of-date; the current release version '3.15'
##   is available with R version '4.2'; see https://bioconductor.org/install
```

```
library(EBImage)
im=readImage("/Users/hainanxu/Documents/spatial_visual_cortex/data/im3.jpg")
display(im,method="raster")
```

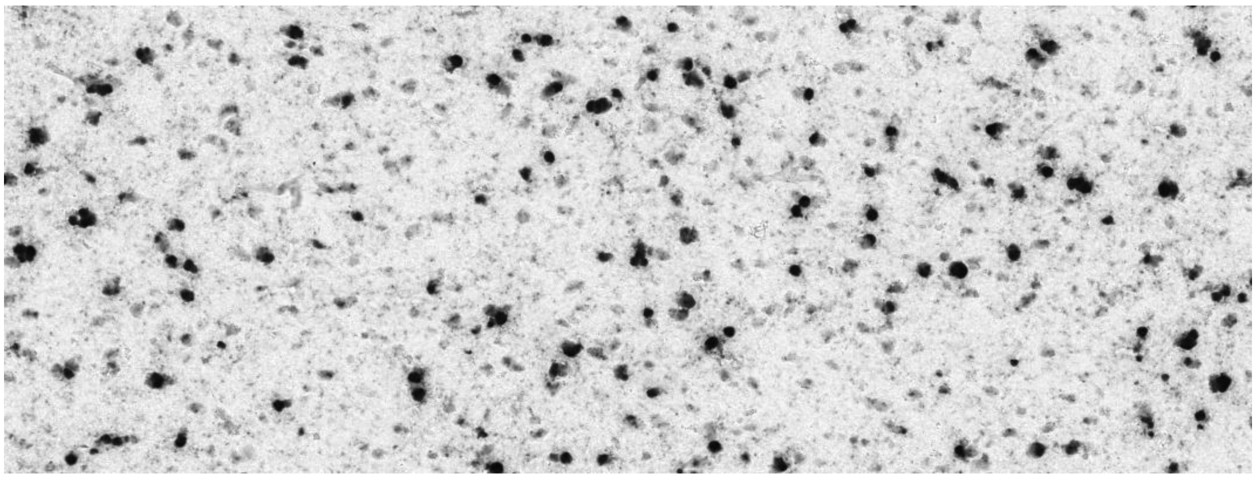


Check the histogram, we can see that there are 3 colors in the histogram, and they are interlaced. To make the segementation easier, we convert them to greyscale(using the red channel).



Take a look at greyscaled `im`.

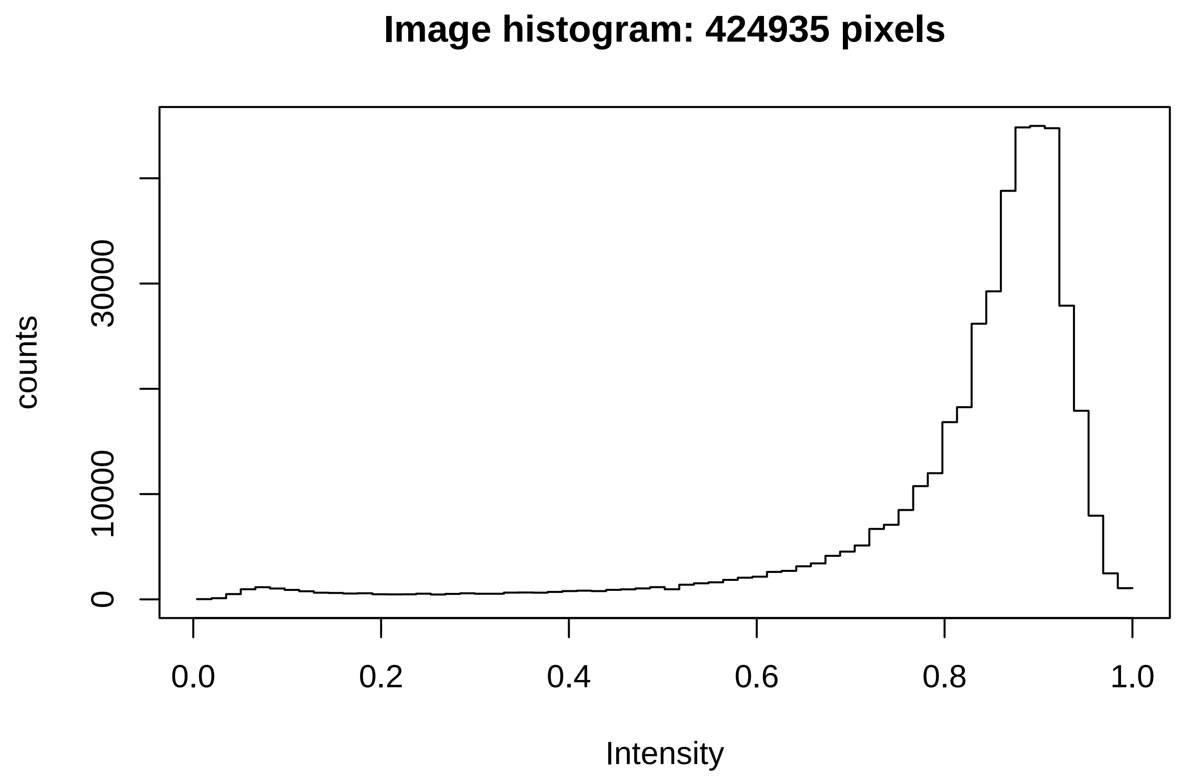
```
display(im,method="raster")
```



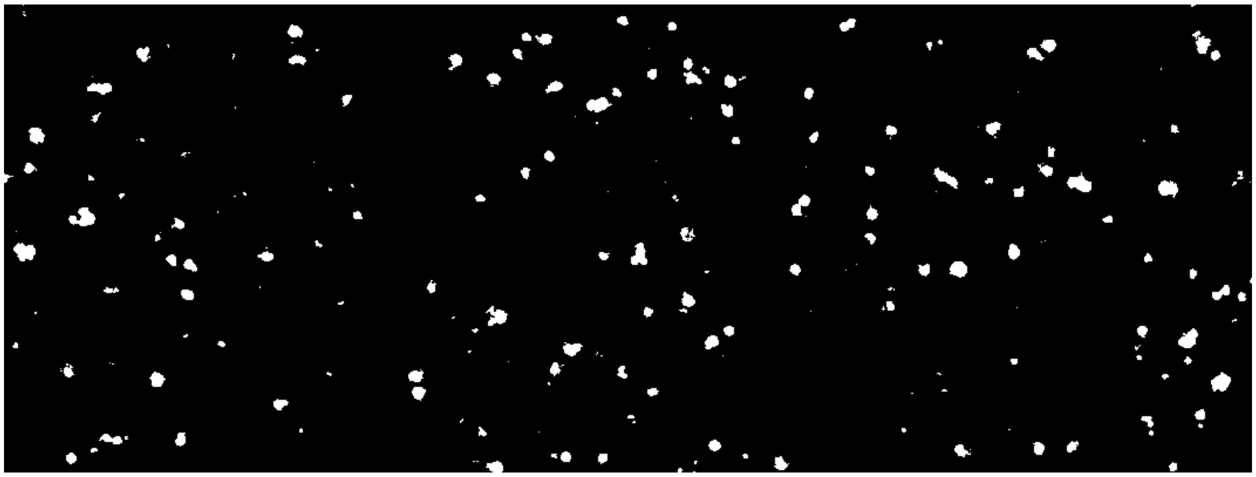
```
#w = makeBrush(size = 30, shape = "gaussian", sigma = 2)
#nucSmooth = filter2(getFrame(r, 1), w)
#display(nucSmooth)
#display(nucSmooth<0.3)
```

After that, we apply a simple method to threshold our greyscale image `im`.

```
hist(im)
```



```
display(im<0.25)
```

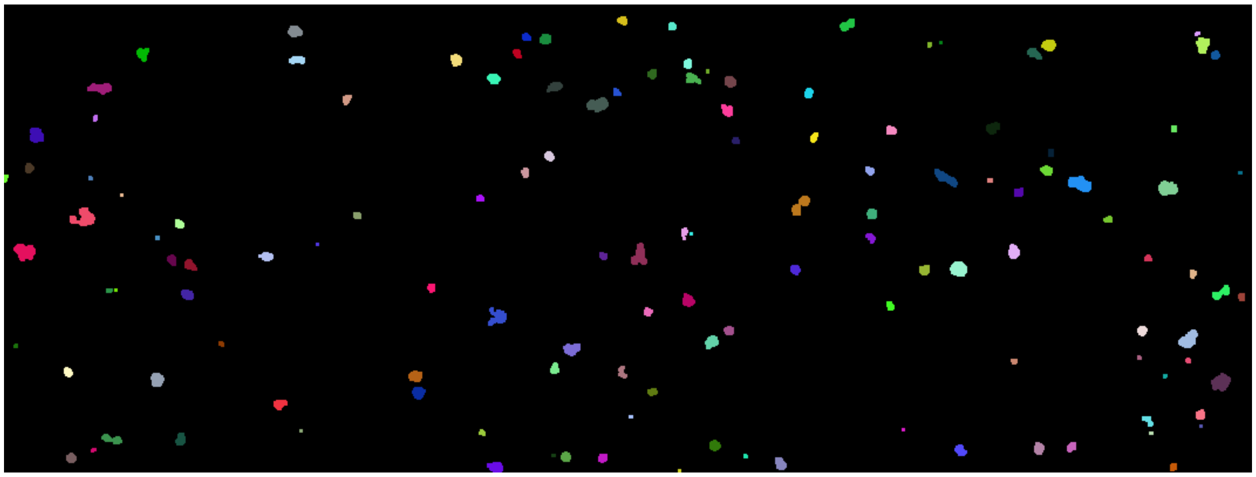


```
rThresh=im<0.25
rOpened = EBImage::opening(rThresh,
                           kern = makeBrush(3, shape = "disc"))
display(rOpened)
```

```r
rRGB=toRGB(rOpened)
display(rRGB)
writeImage(rRGB, "Ahad_June3.tiff", quality = 100)
```

After thresholing our image, we segment it into different objects.

```r
rSeed = bwlabel(rOpened)
display(colorLabels(rSeed),method="raster")
```



Here we computed the features of this slice of image. There are 84 features in total, including shape features and moment features of the MBP cell. We create a new dataframe to store the 3 features that we need right now. The 3 features includes: x coordinate, y coordinate, area of each object. After this, we can convert it to a ppp object.

```
rSeed=bwlabel(rOpened)
table(rSeed)
```

```
## rSeed
##      0      1      2      3      4      5      6      7      8      9     10
## 416219     56     97     44    102     17     47     79    126    101      9
##     11     12     13     14     15     16     17     18     19     20     21
##     19     95     87     50     52     87     83     55     59     12     92
##     22     23     24     25     26     27     28     29     30     31     32
##     85     84     91    146     40     59     55    175     86     22     98
##     33     34     35     36     37     38     39     40     41     42     43
##     62     30    134     48     37     35     61     57     75     50     51
##     44     45     46     47     48     49     50     51     52     53     54
##    156     12     23     15    201     20    171     58      9     37    144
##     55     56     57     58     59     60     61     62     63     64     65
##    216     71     40     40     54     48      9     54     16      9    172
##     66     67     68     69     70     71     72     73     74     75     76
##    206    101     77     42     63     35     79    157     64     71     40
##     77     78     79     80     81     82     83     84     85     86     87
##     47    107     23      9     83     96     39     47    164     49     61
##     88     89     90     91     92     93     94     95     96     97     98
##     63    176    104     22     15    124     15     22     27     62     60
##     99    100    101    102    103    104    105    106    107    108    109
##     55     98    115    188     15    100     50     82     62     12     57
##    110    111    112    113    114    115    116    117    118    119    120
##      9      9      9     28     12     79    100     76     80     58     78
##    121    122    123    124    125    126    127    128    129    130
##     17     66     64     12     56     15     87    104     39      9
```

```
F1 = computeFeatures(rSeed,im, xname = "r",
                     refnames = "r")
```

Step2: Intensity estimation and other graphs

First, create a new dataframe MBP. We can explore the distributions of the cells by computing the marginal distribution.

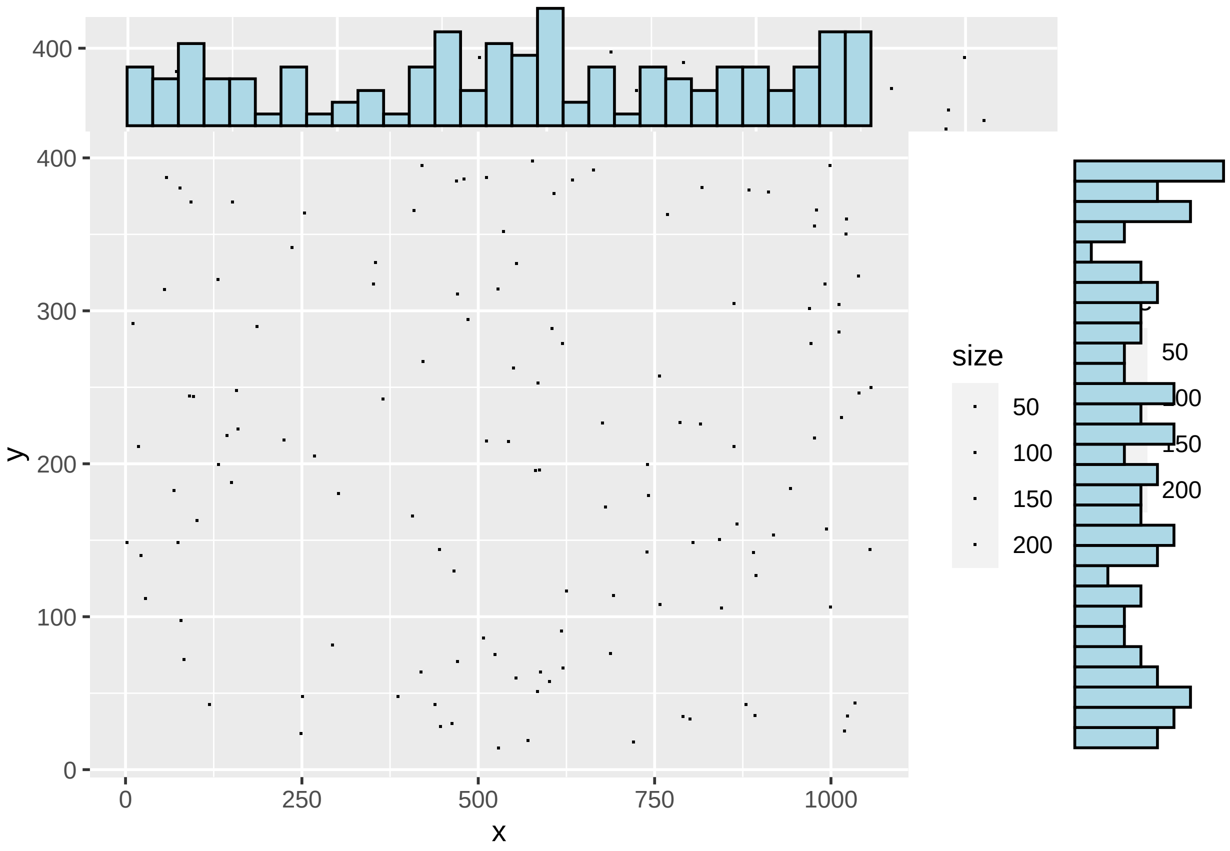
```
library(ggplot2)
library(ggExtra)
x_coord<-F1[,1]
y_coord<-F1[,2]
size<-F1[,6]

View(F1)

 MBP<-data.frame(x=x_coord,
             y=y_coord,
             size=size
          )

range(size)
```

```
## [1]   9 216
```

```
head(MBP)
```

```
##           x        y size
## 1  528.4286 14.30357   56
## 2  720.3608 18.02062   97
## 3  570.8182 19.18182   44
## 4  248.9020 23.55882  102
## 5 1018.9412 25.41176   17
## 6  446.3404 28.36170   47
```

```
a<-ggplot(MBP,
   aes(x = x, y = y,size=size)) + geom_point(shape = ".",aes(size=size))
a
ggMarginal(a,type="histogram",fill = "lightblue")
```

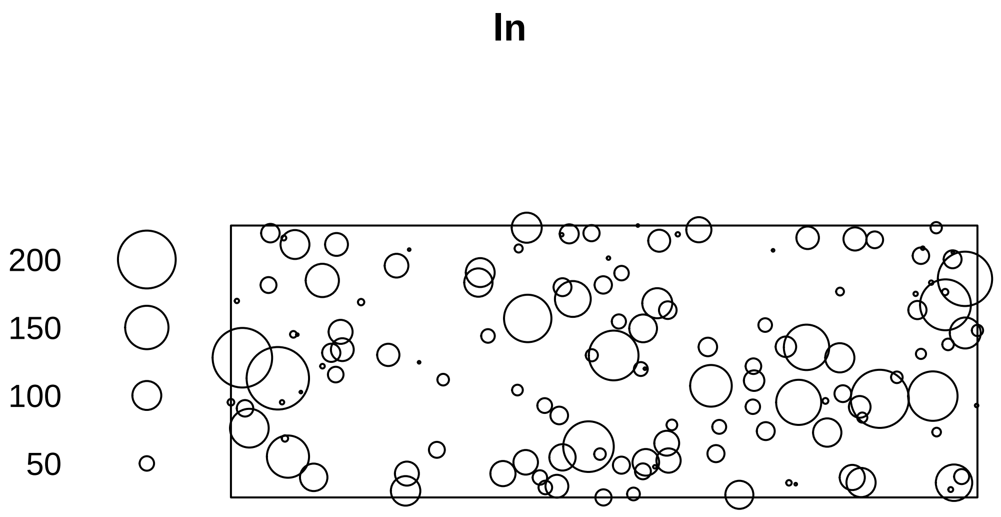


Secondly, convert MBP as a porint process object. Here each circle indicates a cell, and the size of the cell is corresponding to the area each cell as occupied in pixel.

```
library("spatstat")
```

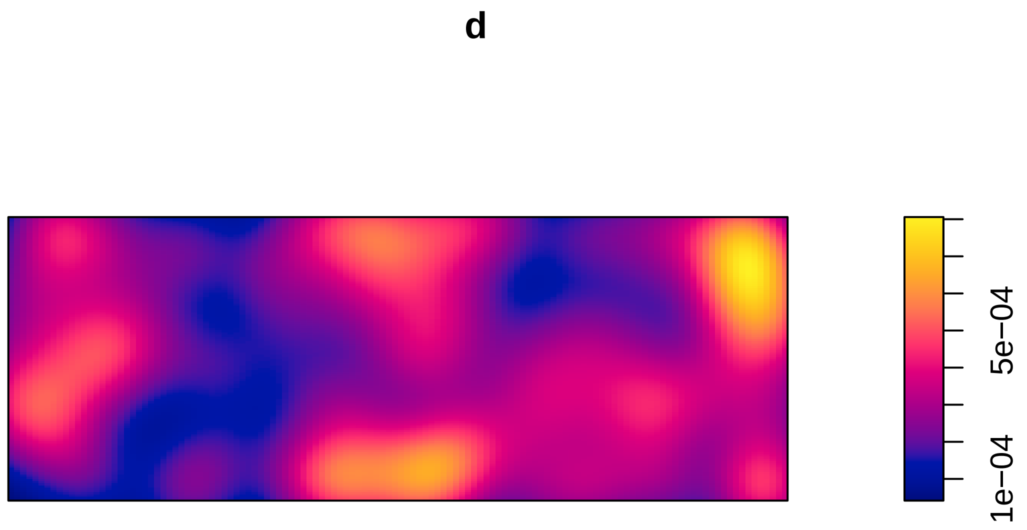
```
## Loading required package: spatstat.data

## Loading required package: spatstat.geom

## spatstat.geom 2.4-0

##
## Attaching package: 'spatstat.geom'

## The following objects are masked from 'package:EBImage':
##
##     affine, closing, distmap, opening, rotate

## Loading required package: spatstat.random

## spatstat.random 2.2-0

## Loading required package: spatstat.core

## Loading required package: nlme
## Loading required package: rpart

## spatstat.core 2.4-4

##
## Attaching package: 'spatstat.core'

## The following object is masked from 'package:BiocManager':
##
##     valid

## Loading required package: spatstat.linnet

## spatstat.linnet 2.3-2

##
## spatstat 2.3-4       (nickname: 'Watch this space')
## For an introduction to spatstat, type 'beginner'
```

```
ln = with(MBP,
  ppp(x = x, y = y, marks = size, xrange = range(x), yrange = range(y)))
plot(ln)
```



Thirdly, we can explore the first order effect: estimate the intensity of this particular region.

```
d = density(subset(ln), edge=TRUE, diggle=TRUE)
plot(d)
```



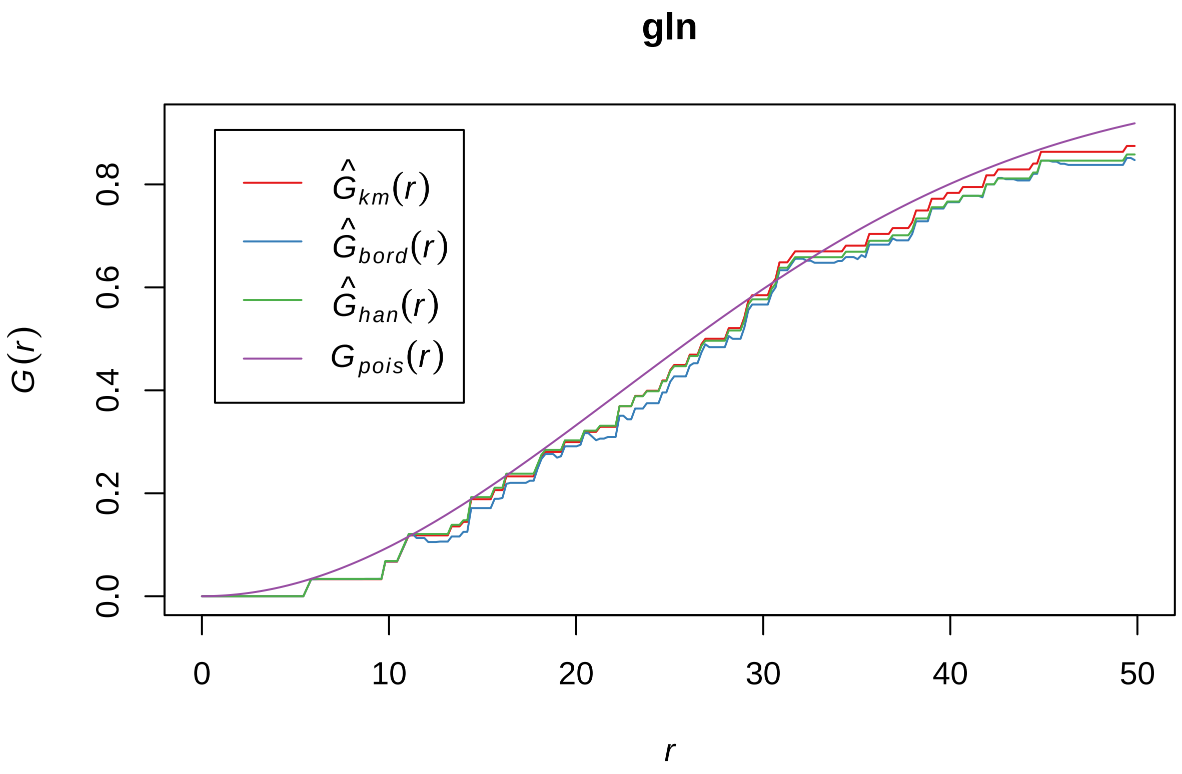
Fourthly, we can explore the second order effect: randomly pick a point, what is the distance to it's nearest neigbor?

G function: the cumulative distribution function of the distance. Here the purple line indicates the possion process. The second graph is the zoomed in version of the first graph.

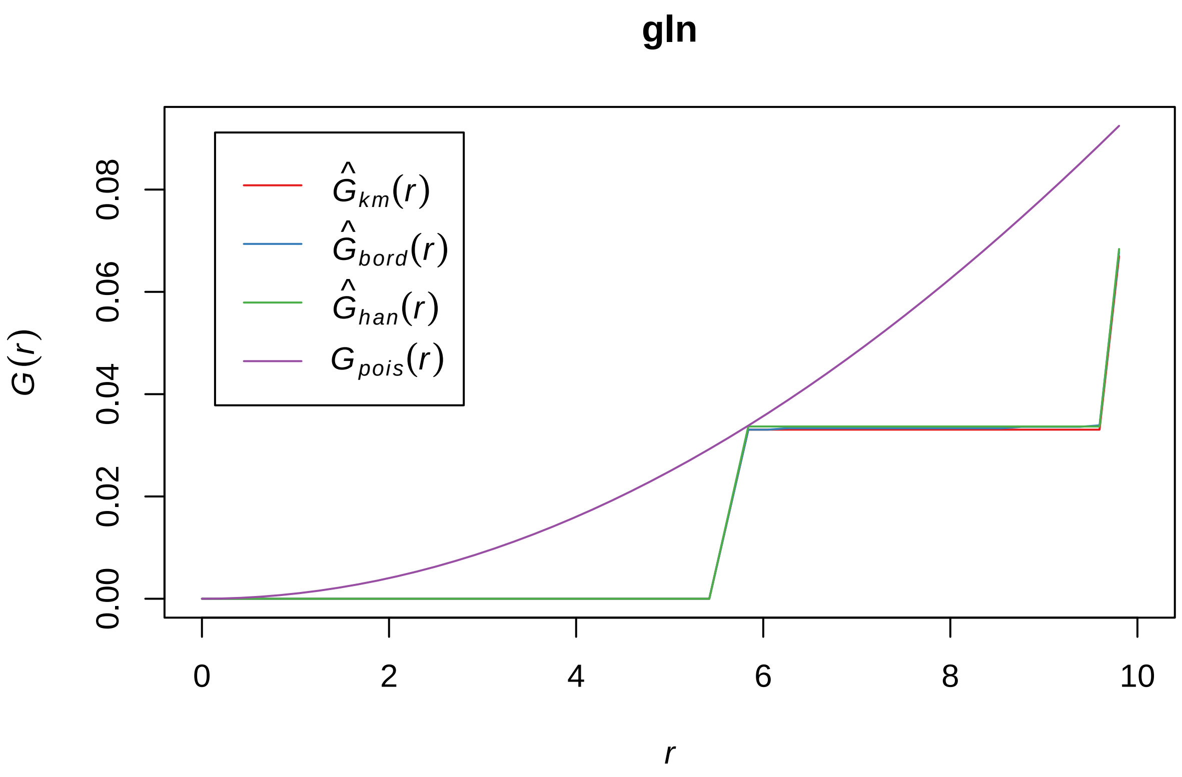
```
gln = Gest(ln)
gln
```

```
## Function value object (class 'fv')
## for the function r -> G(r)
## .....................................................................
##          Math.label      Description
## r        r               distance argument r
## theo     G[pois](r)      theoretical Poisson G(r)
## han      hat(G)[han](r)  Hanisch estimate of G(r)
## rs       hat(G)[bord](r) border corrected estimate of G(r)
## km       hat(G)[km](r)   Kaplan-Meier estimate of G(r)
## hazard   hat(h)[km](r)   Kaplan-Meier estimate of hazard function h(r)
## theohaz  h[pois](r)      theoretical Poisson hazard function h(r)
## .....................................................................
## Default plot formula:  .~r
## where "." stands for 'km', 'rs', 'han', 'theo'
## Recommended range of argument r: [0, 50.895]
## Available range of argument r: [0, 106.8]
```

```
library("RColorBrewer")
plot(gln, xlim = c(0, 50), lty = 1, col = brewer.pal(4, "Set1"))
```

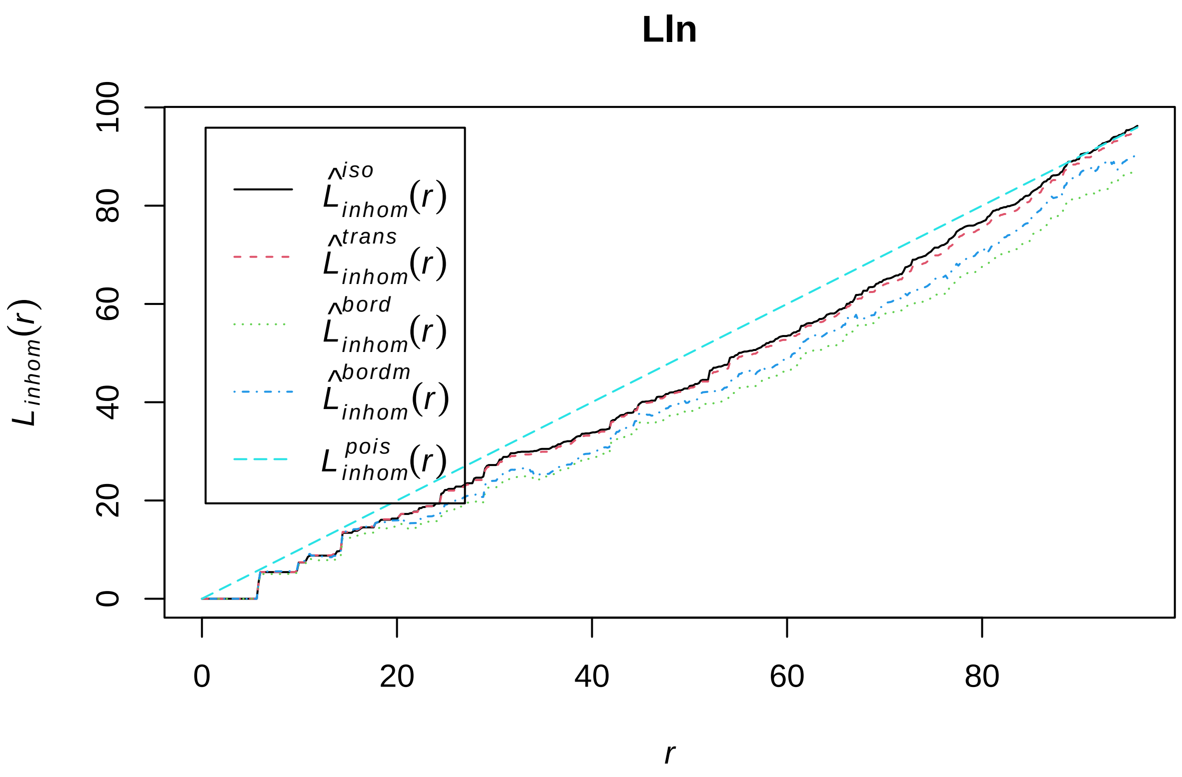


```
plot(gln, xlim = c(0, 10), lty = 1, col = brewer.pal(4, "Set1"))
```



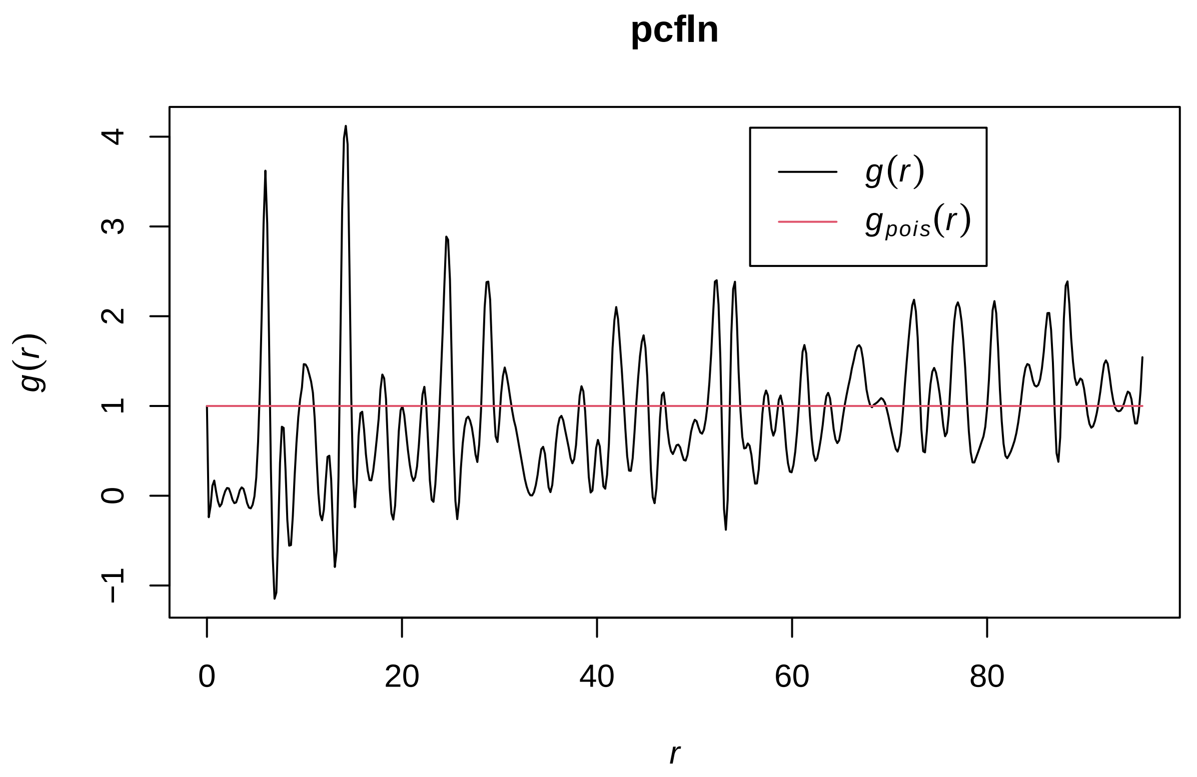
Ripley's K function: for estimation and visualization, transfer K function as L function.

```
Lln = Linhom(ln)
plot(Lln)
```

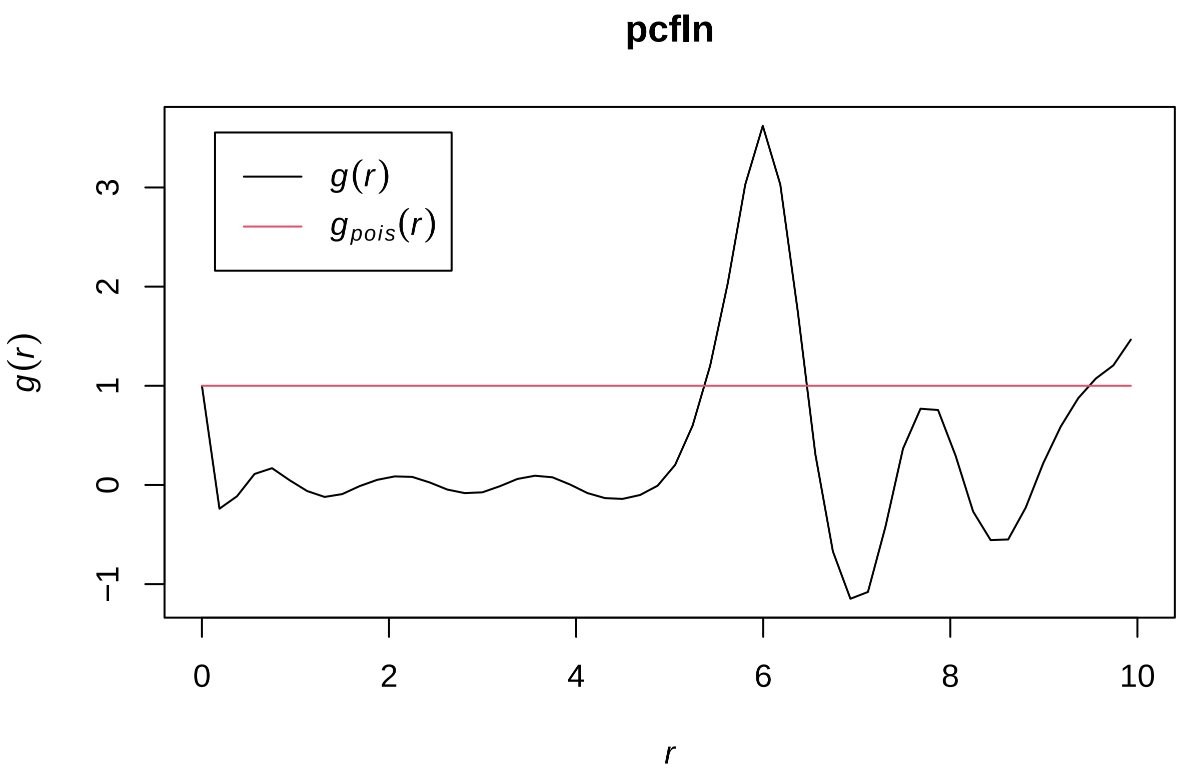


Paired correlation function:

```
pcfln = pcf(Kinhom(ln))
plot(pcfln, lty = 1)
```



```
plot(pcfln, lty = 1, xlim = c(0, 10))
```


pcfln
$g(r)$
$g_{pois}(r)$
$g(r)$
r
−1
0
1
2
3
0
2
4
6
8
10