Resample-smoothing of Voronoi intensity estimators

Ege Rubak

useR!2019: 11 July 2019

Introduction

The presentation is based on joint work with these great guys:

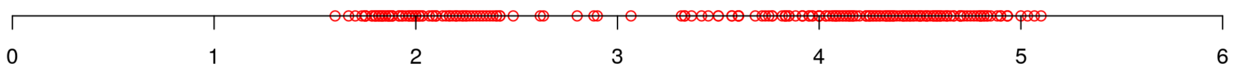
- M. Mehdi Moradi (unfortunately denied entrance to France for useR!2019)
- Ottmar Cronie
- Raphael Lachieze-Rey
- Jorge Mateu
- Adrian Baddeley

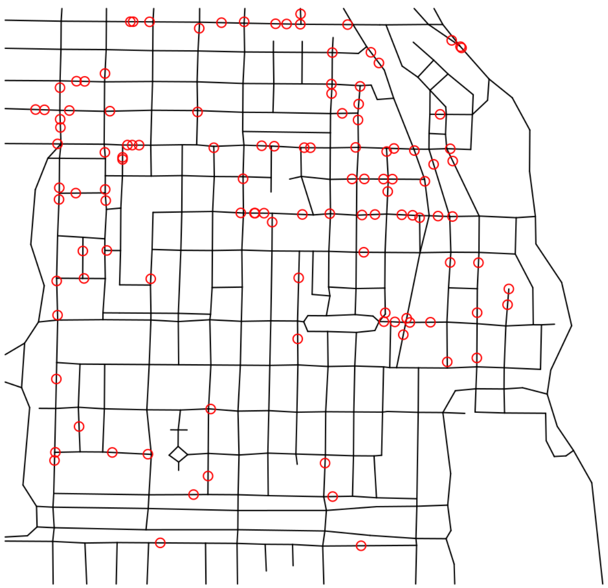
The results are published in:

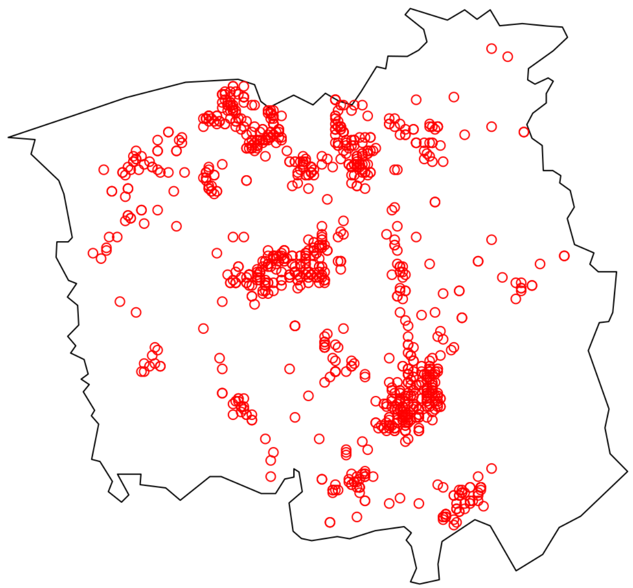
M. M. Moradi, et al. (2019) Resample-smoothing of Voronoi intensity estimators, Statistics and Computing, pp. 1-16. DOI: 10.1007/s11222-018-09850-0

Examples of point patterns

Random locations on e.g. the time line (Old Faithful Geyser eruptions), a network (street crimes around University of Chicago), or a planar region (cancer cases in part of Lancashire).

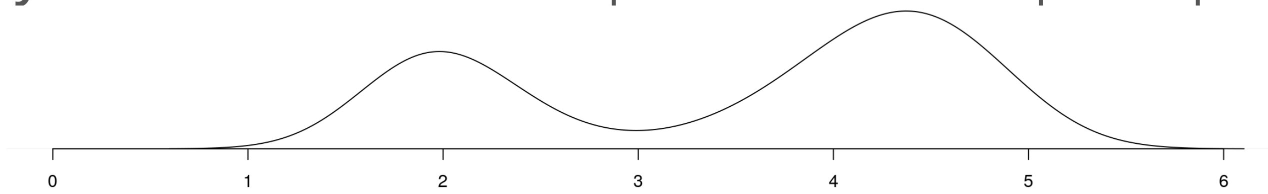


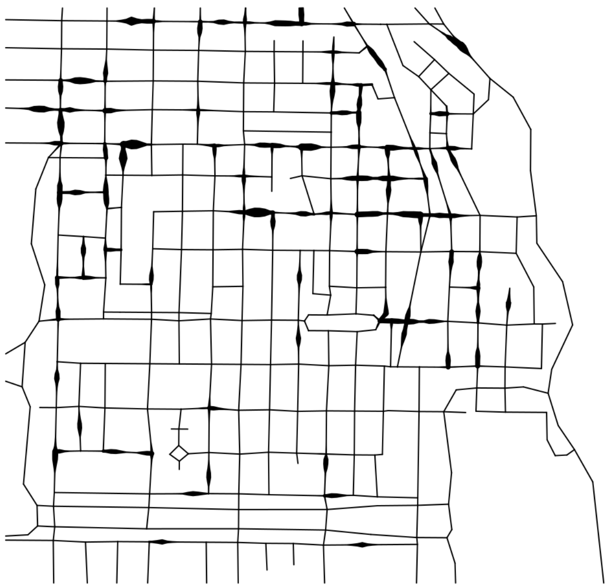




The intensity function

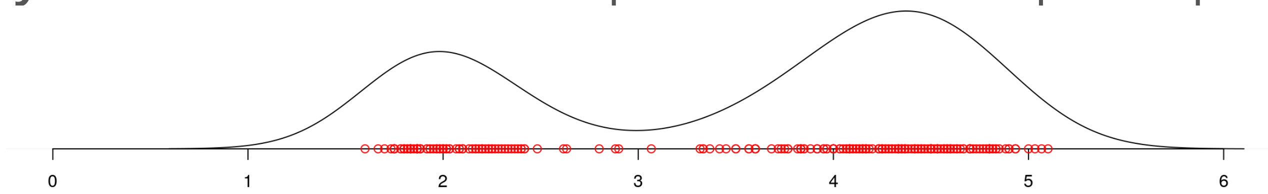
- The **intensity function** controls the expected number of points per unit area.

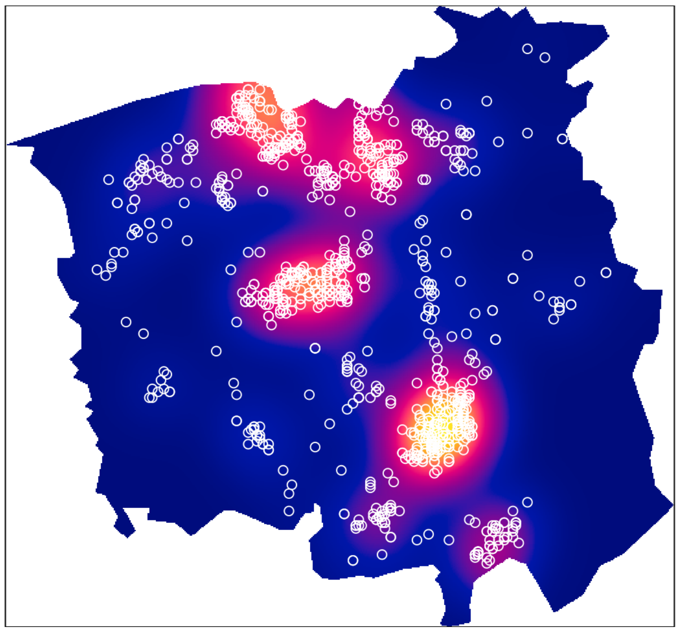




The intensity function

- The **intensity function** controls the expected number of points per unit area.



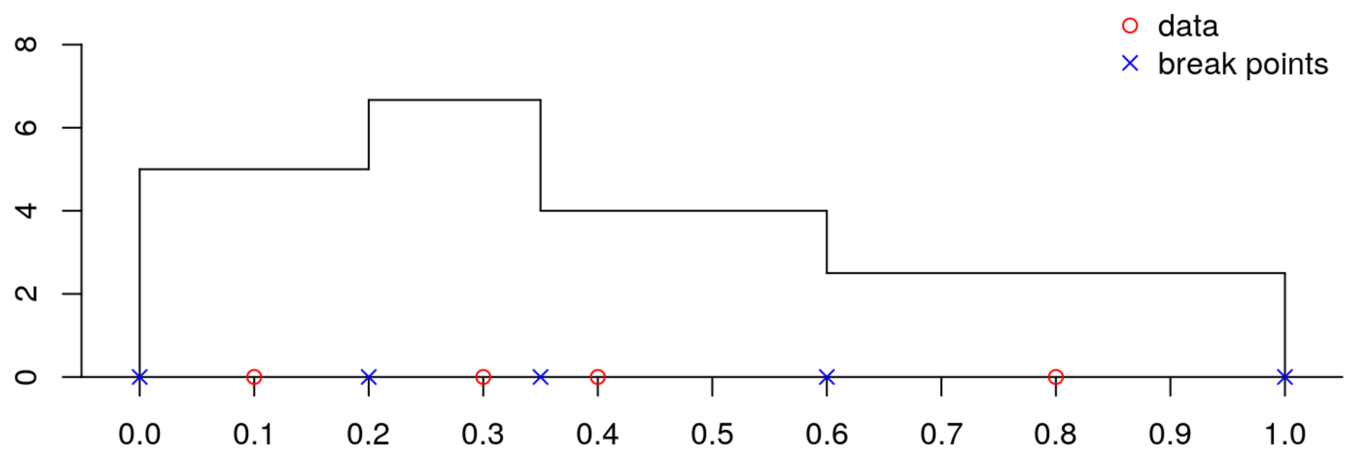


Intensity estimation

- The intensity is like a unnormalized density. The integral is the expected number of points rather than the probabily of outcomes.
- All the classical density estimation techniques apply.
- Most common non-parametric method is kernel smoothing.
- Our starting points is the Voronoi estimator, which is closely related to nearest neighbour density estimation.

Voronoi intensity estimator

- Example with points on the unit interval [0,1].
- Cut [0,1] into Voronoi regions closest to each data point and use reciprocal region size as piecewise constant intensity estimate (think histogram with varying bin size).

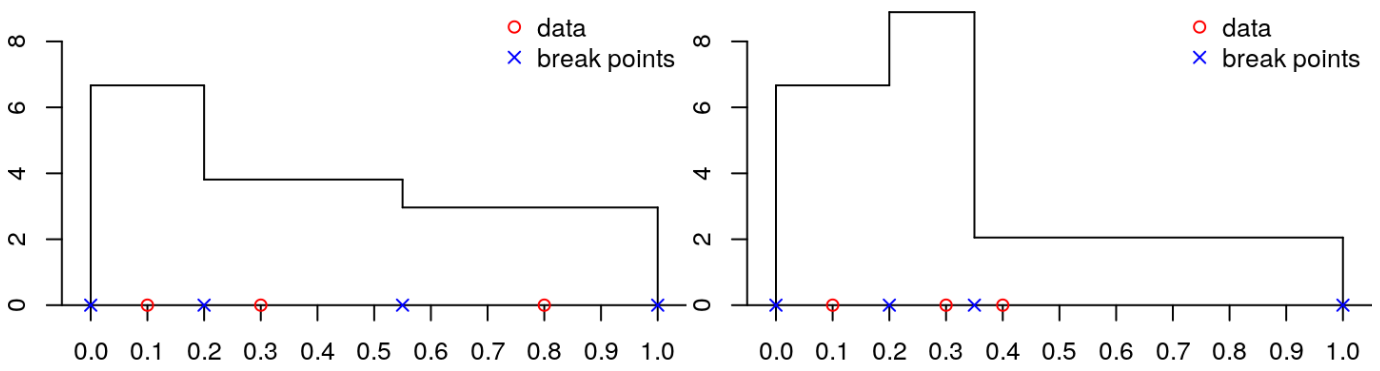


Issues with the Voronoi estimator

- The Voronoi estimator is approximately unbiased (apart from edge effects), but the variance is huge.
- We propose to reduce the variance (and introduce a bit of bias), by subsampling.
- We first illustrate ideas with deterministic subsampling, but for real data we use random subsampling.

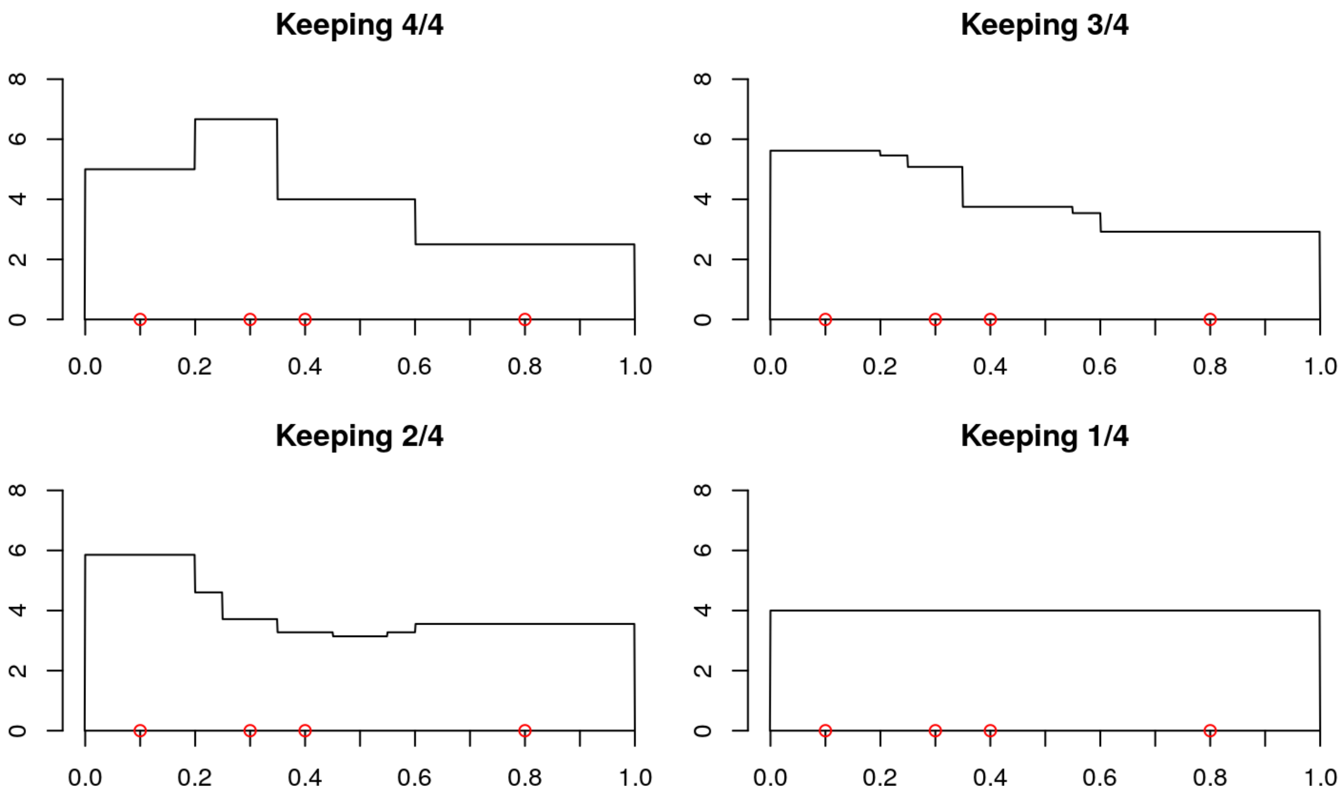
Subsampling

- Leave out one data point at the time and rescale by n/(n-1).

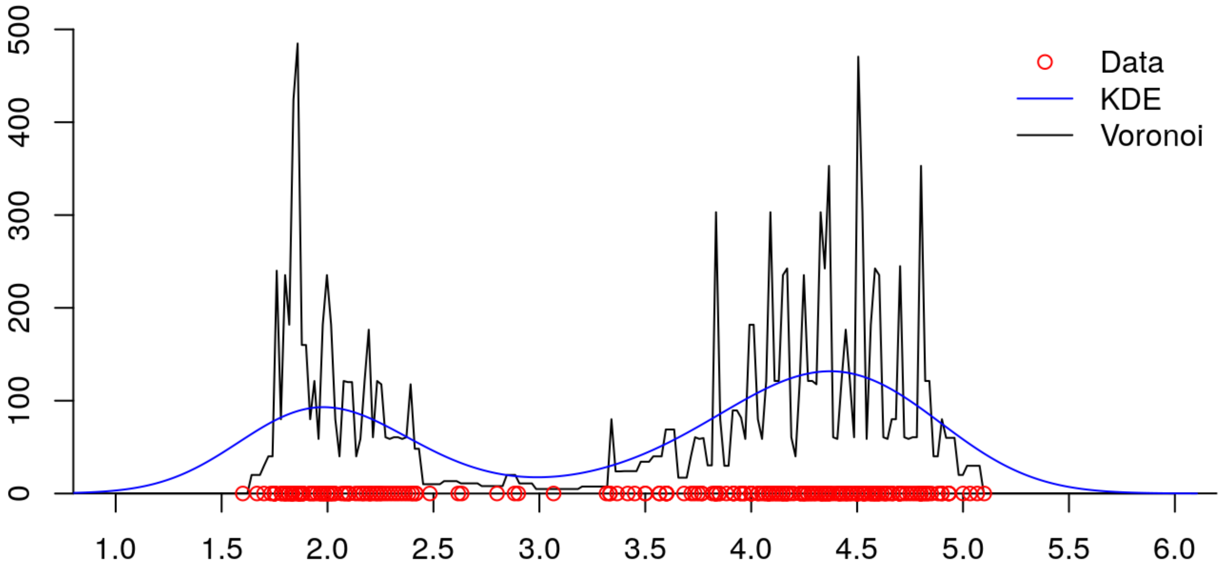


Different levels of subsampling

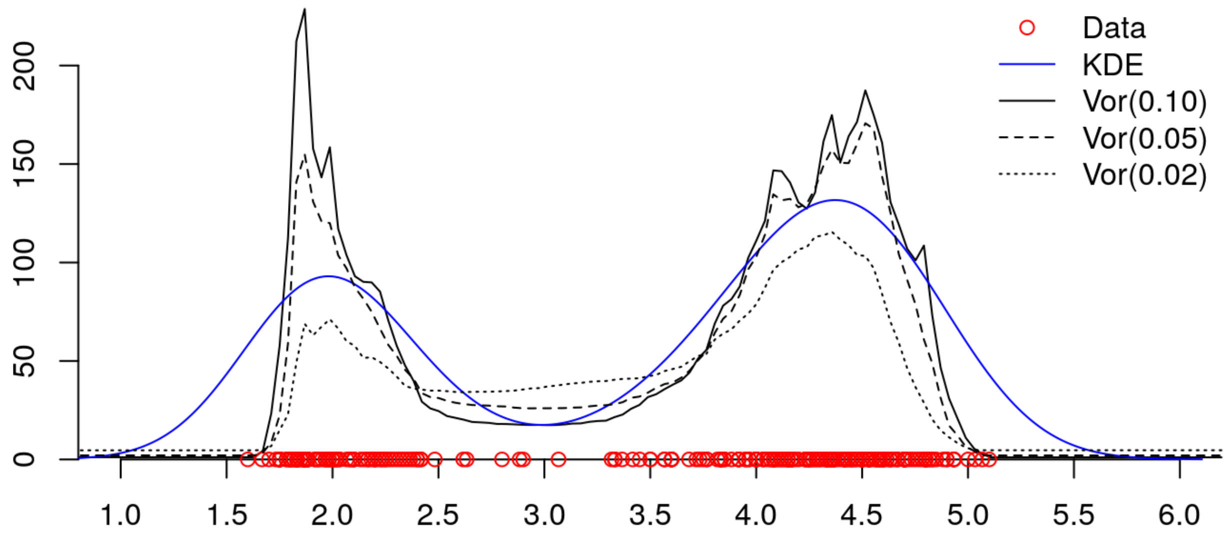
- Final estimate at a given level of subsampling is the average of corresponding subsampling estimates.



Voronoi intensity estimate of eruptions



Subsampling smoothed version



Ties

- This is the point where you say: “Wait a minute, you must be cheating! I clearly remeber the eruption durations in `datasets::faithful` has ties!”
- In case of ties we simply scale the estimate by the number of identical data points, i.e., we replace 1/size by n/size.

Simulation study

- As for almost any method, it is possible to construct a simulation study where smoothed Voronoi outperforms other methods and vice versa.
- I will not do that here :-)
- Instead we will look a bit at the detail of the implementation.

Implementation in spatstat package

- The starting point of `spatstat` 20 years ago was 2D planar point patterns (class `ppp`) in a given region (observation window – `owin`).
- Data can be imported directly from csv-files etc. or converted from other formats using `sf`, `sp` and `maptools`.
- Here we use a built-in dataset to avoid technicalities.

```
library(spatstat)
X <- unmark(chorley) # Cancer data without types
print(X)
```

```
## Planar point pattern: 1036 points
## window: polygonal boundary
## enclosing rectangle: [343.45, 366.45] x [410.41, 431.79] km
```

Planar point patterns in spatstat

```
methods(class = "ppp")
```

```
##   [1] [                  [<-               affine           anyDuplicated
##   [5] as.data.frame      as.im             as.layered       as.owin
##   [9] as.ppp             auc               berman.test      boundingbox
##  [13] boundingcentre     boundingcircle    boundingradius   by
##  [17] cdf.test           circumradius      closepairs       closing
##  [21] connected          coords            coords<-         crossdist
##  [25] crosspairs         cut               density          densityAdaptiveKern
##  [29] densityfun         densityVoronoi    dilation         distfun
##  [33] distmap            domain            duplicated       edit
##  [37] envelope           erosion           fardist          flipxy
##  [41] Frame<-            has.close         head             identify
##  [45] intensity          iplot             is.connected     is.empty
##  [49] is.marked          is.multitype      kppm             lurking
##  [53] markformat         marks             marks<-          multiplicity
##  [57] nnclean            nncross           nndensity        nndist
##  [61] nnfun              nnwhich           nobjects         npoints
##  [65] opening            pairdist          pcf              periodify
##  [69] pixellate          plot              ppm              print
```

Tesselations in spatstat

- We use `deldir` to calculate tesselations which have class `tess` in `spatstat`:

```
tes <- dirichlet(X)
print(tes)
```

```
## Tessellation
## Tiles are irregular polygons
## 706 tiles (irregular windows)
## window: polygonal boundary
## enclosing rectangle: [343.45, 366.45] x [410.41, 431.79] km
```

```
methods(class = "tess")
```

```
##  [1] [             [<-           affine        as.data.frame as.function   as.im
##  [7] as.owin       as.tess       connected     domain        flipxy        head
## [13] marks         marks<-       nobjects      plot          print         reflect
## [19] rotate        scalardilate  shift         tail          unitname      unitname<-
## [25] unmark        unstack       Window
## see '?methods' for accessing help and source code
```

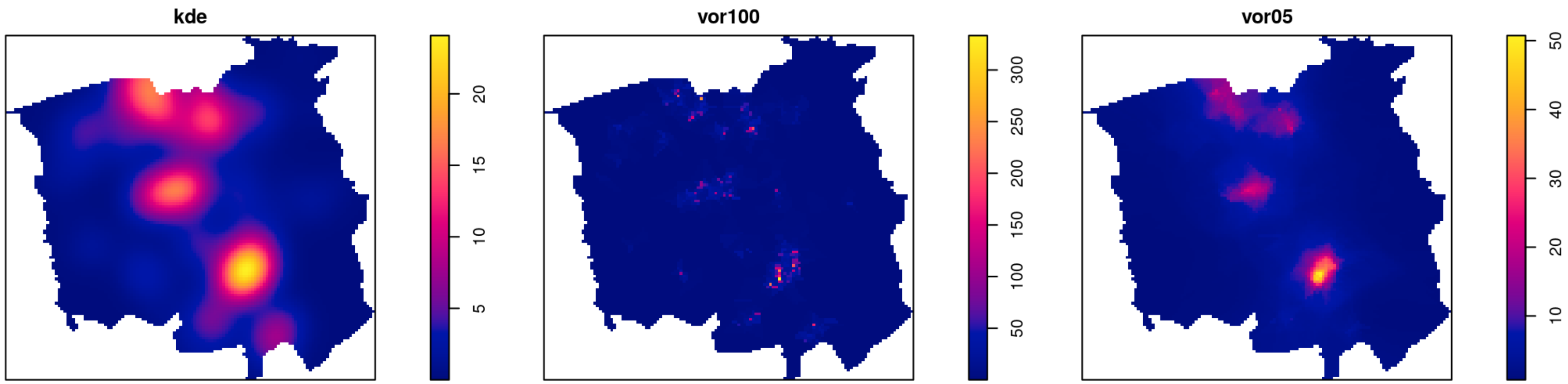
Tesselations in spatstat

```
plot(tes, main = "")
```

Intensity estimation in spatstat

- Intensity estimation by kernel smoothing is the default in `spatstat`. It is a S3 method for `density` in line with base R, while the new method has its own generic function `densityVoronoi`:

```
kde <- density(X, sigma = 1)
vor100 <- densityVoronoi(X, f = 1.0)
vor05 <- densityVoronoi(X, f = 0.05, nrep = 100, verbose = FALSE)
```



Linear networks in spatstat

- More recently support for linear networks have been added to `spatstat`.
- Linear networks are like graphs, but less general, since they are embedded in space and positions of vertices have meaning.

```
X <- unmark(chicago) ## Chicago street crime without types
X
```

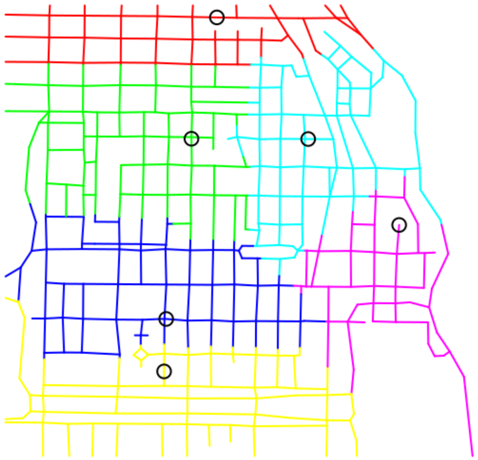
```
## Point pattern on linear network
## 116 points
## Linear network with 338 vertices and 503 lines
## Enclosing window: rectangle = [0.3894, 1281.9863] x [153.1035, 1276.5602] feet
```

Linear networks in spatstat

```
##  [1] [               affine         as.linnet      as.owin        as.ppp         as.psp
##  [7] auc             berman.test    boundingbox    cdf.test       connected      crossdist
## [13] cut             deletebranch   density        densityVoronoi distfun        domain
## [19] envelope        extractbranch  identify       intensity      iplot          is.multityp
## [25] lppm            marks<-        nncross        nndist         nnfun          nnwhich
## [31] nsegments       pairdist       plot           points         print          rescale
## [37] rhohat          roc            rotate         scalardilate   shift          subset
## [43] summary         superimpose    text           uniquemap      unitname       unitname<-
## [49] unmark          unstack        Window         Window<-
## see '?methods' for accessing help and source code
```

Linear network tesselations in spatstat

```
Xsmall <- X[sample(npoints(X), size = 6)]
tes <- lineardirichlet(Xsmall)
plot(tes, main = "")
plot(Xsmall, add = TRUE, show.network = FALSE)
```



Voronoi estimation on linear networks

```
est <- densityVoronoi(X, f = 0.05, nrep = 500)
plot(est, style = "width", main = "")
```