Topic 1: Working with Big Spatial Data

This topic discusses Big Data challenges in spatial analysis and demonstrate some applications of the studied spatial methods to real estate data.

Analysis of Big Spatial Data

The main challenges in the analysis of big spatial data:

- Collection of Big Spatial Data
- Quality Assessment
- Structuring and Representation
- Parallel and Cloud Programming
- Visualisation
- Spatial Data Mining
- Data Modeling and Analysis

Approaches in Modeling and Analysis of Big Spatial Data:

- Moving windows to deal with heterogeneity (relationships that vary over space and time)
- Non-parametric statistics
- Multi-resolution approaches (wavelets)
- Feature extraction to reduce dimensions
- Low-rank approximation (spatial process is approximated by a linear combination of a relatively small set of basis functions. Often it misses small-scale variations)
- Sparse approximation (spatial partitioning based on the assumption of independence between observations across subregions)
- Spatial sampling (efficient selection of a set of representative data objects)
- Bayesian modeling

Analysis of Irish Property Prices by using kriging

For this example we use the data from the website `www.shanelynn.ie/the-irish-property-price-register-geocoded-to-small-areas`

These are comprehensive data with GPS coordinates of the Property Price Register data from years 2012-2017 (approx 220k property sales).

We will use the following R libraries:

```
> library(sp)
> library(geoR)
> library(gstat)
> library(spatstat)
```

The data set is available in the LMS folder Data as the csv flie: `ppr_data_encoded.csv`

First, we import the data in R:

```
> MyData <- read.csv("ppr_data_encoded.csv", header=TRUE)
> str(MyData)
'data.frame': 217231 obs. of  24 variables:
$ X                : int  1 2 3 4 5 6 7 8 9 10 ...
$ year             : int  2012 2012 2012 2012  ...
$ input_string     : chr  "' St. Martins', Ightermurragh, ...
$ sale_date        : chr  "18/09/2012" "16/01/2012" ...
$ address          : chr  "' St. Martins', Ightermurragh, ...
$ postal_code      : chr  "" "" "" "" ...
$ ppr_county       : chr  "Cork" "Tipperary" "Kildare" ...
$ price            : num  118000 210000 220000 496089  ...
...
```

Then we remove the cases with missing information:

```
> MyData <- MyData[complete.cases(MyData),]
```

From all fields we select only geographic coordinates and sale prices for the following analysis.

```
> MyData1 <- MyData[,c("latitude", "longitude","price")]
```

There are some cases with duplicated locations. They may impact our analysis and those methods that assume different spatial points. There are several strategies how to deal with such cases. One of them is to add small amount of random noise to coordinates by using the function **jitter**. It will displace duplicated points.

```
> str(MyData1[duplicated(MyData1[1:2]),])
'data.frame': 72826 obs. of  3 variables:
$ latitude : num  52.7 52.7 52.7 53.4 53.4 ...
$ longitude: num  -6.29 -6.29 -6.29 -6.38 -6.38 ...
$ price    : num  280000 260000 185000 180000 180000 ...
> MyData10 <- MyData1
```

```
> MyData10$latitude <- jitter(MyData10$latitude,1)
> str(MyData10[duplicated(MyData10[1:2]),])
'data.frame': 0 obs. of  3 variables:
$ latitude : num
$ longitude: num
$ price    : num
```

If one believes that such points were introduced by error, they can be removed by deleting duplicated coordinates.

```
> MyData1 <- MyData1[!duplicated(MyData1[1:2]),]
> str(MyData1[duplicated(MyData10[1:2]),])
'data.frame': 0 obs. of  3 variables:
$ latitude : num
$ longitude: num
$ price    : num
```

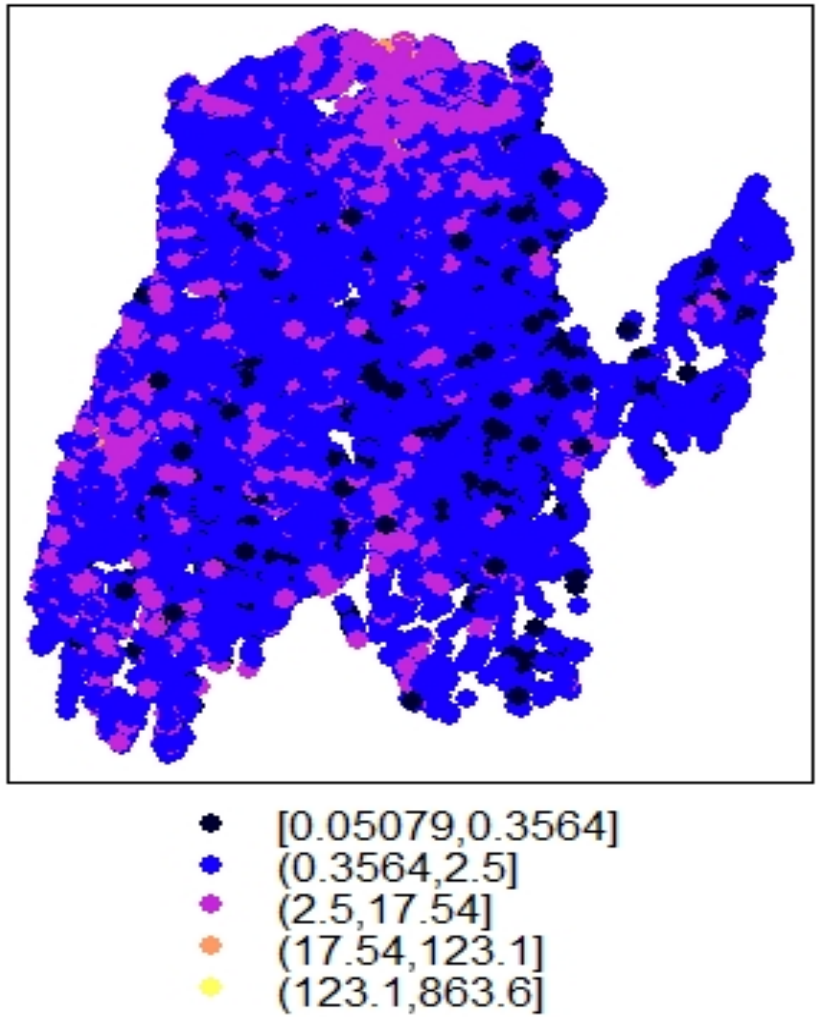
Then we rescale coordinates as their differences are very small and prices because their values are too large. We transform the obtained object to the class 'SpatialPointsDataFrame':

```
> MyData1$price <- MyData1$price/100000
> MyData1$latitude <- MyData1$latitude*100
> MyData1$longitude <- MyData1$longitude*100
> coordinates(MyData1) <- c("latitude", "longitude")
```

```
> str(MyData1)
Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots
..@ data       :'data.frame': 129568 obs. of  1 variable:
.. ..$ price: num [1:129568] 1.18 2.1 2.2 4.96 2.4 ...
..@ coords.nrs : int [1:2] 1 2
..@ coords     : num [1:129568, 1:2] 5190 5244 5318 5345 5188 ...
.. ..- attr(*, "dimnames")=List of 2
.. .. ..$ : chr [1:129568] "1" "2" "3" "4" ...
.. .. ..$ : chr [1:2] "latitude" "longitude"
..@ bbox       : num [1:2, 1:2] 5144 -1046 5537 -601
```

Finally, we visualise the spatial object MyData1

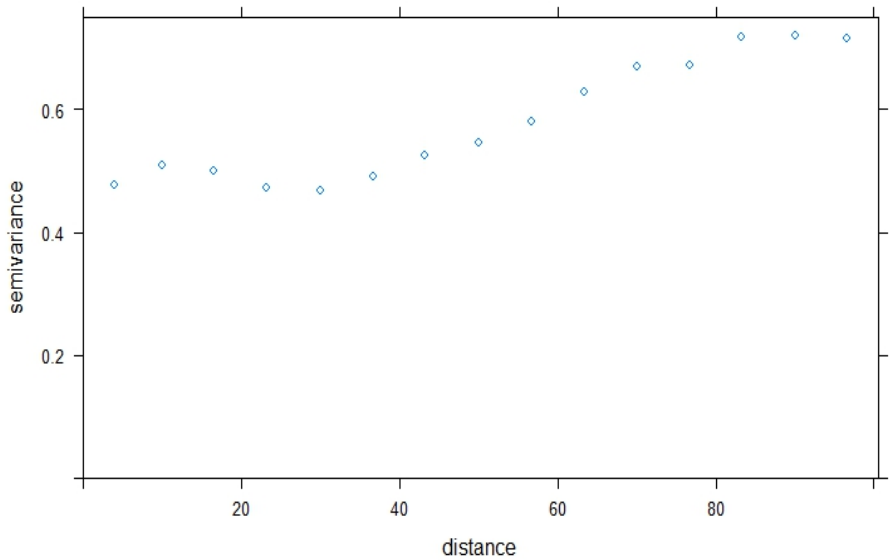
```
> spplot(MyData1,"price", do.log = T)
```



The option do.log = T was used to plot prices on the log-linear scale for better color representation.

As the plot shows, there are many spatial locations which can slow computations. For example, try to compute the sample covariance function (a rather lengthy process):

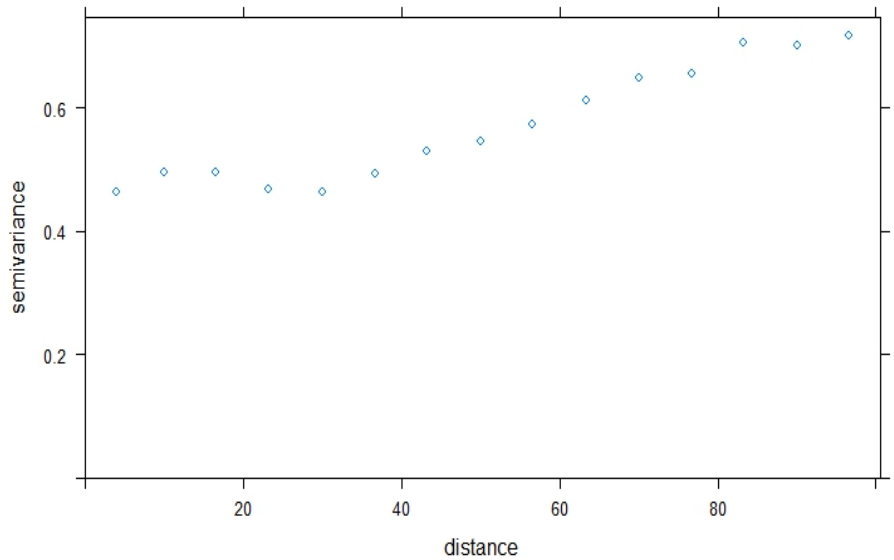
```
> length(MyData1)
[1] 129568
> v  <- variogram(log(price) ~ 1, MyData1, cutoff=100)
> plot(v)
```



One of possible ways to deal with such situations is to consider a representative sub-sample of a smaller size. For example,

```
> MyData0 <- MyData1[sample(1:length(MyData1),10000),]
> v  <- variogram(log(price) ~ 1, MyData0, cutoff=100)
> plot(v)
```

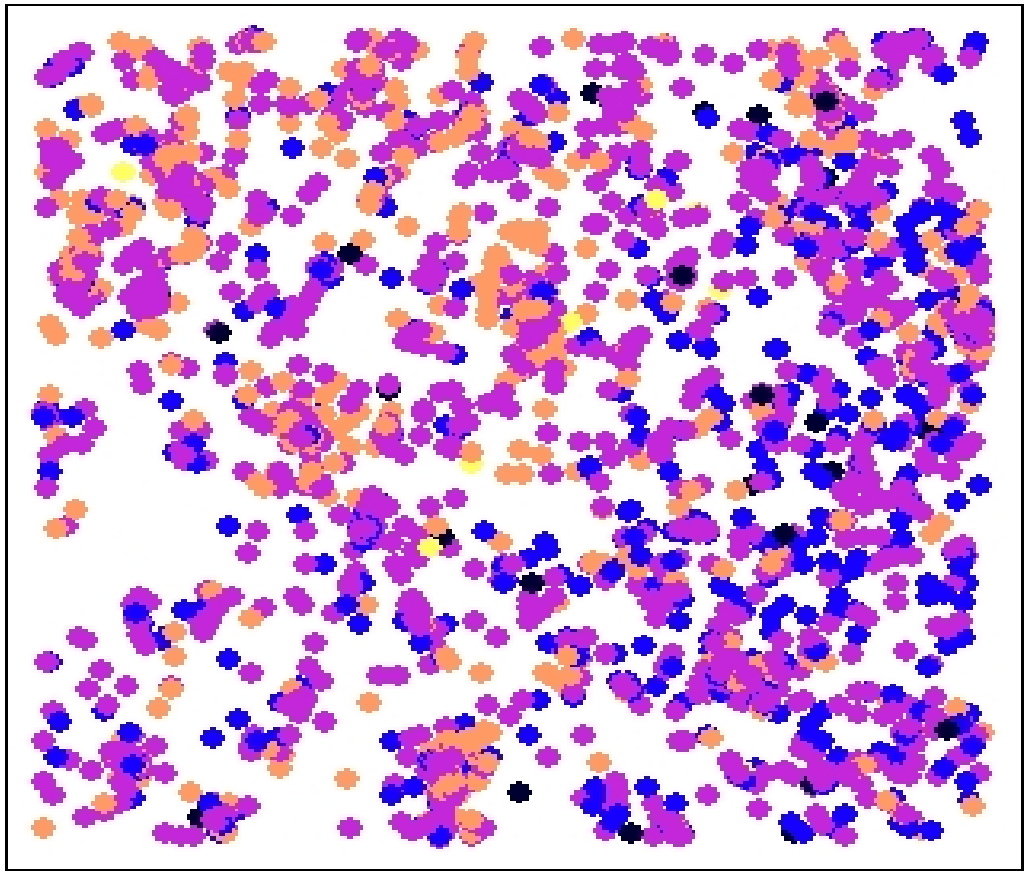
This much smaller sub-sample gives the sample covariance function which is rather similar to the corresponding function for all locations.



Also, as the data can be non-stationary applying the same model for all regions can lead to an inadequate model. One of strategies to deal with such situations is to split the area in smaller sub-areas to analyse them separately. For example, we select a smaller sub-area (window), which let us better visualize results:

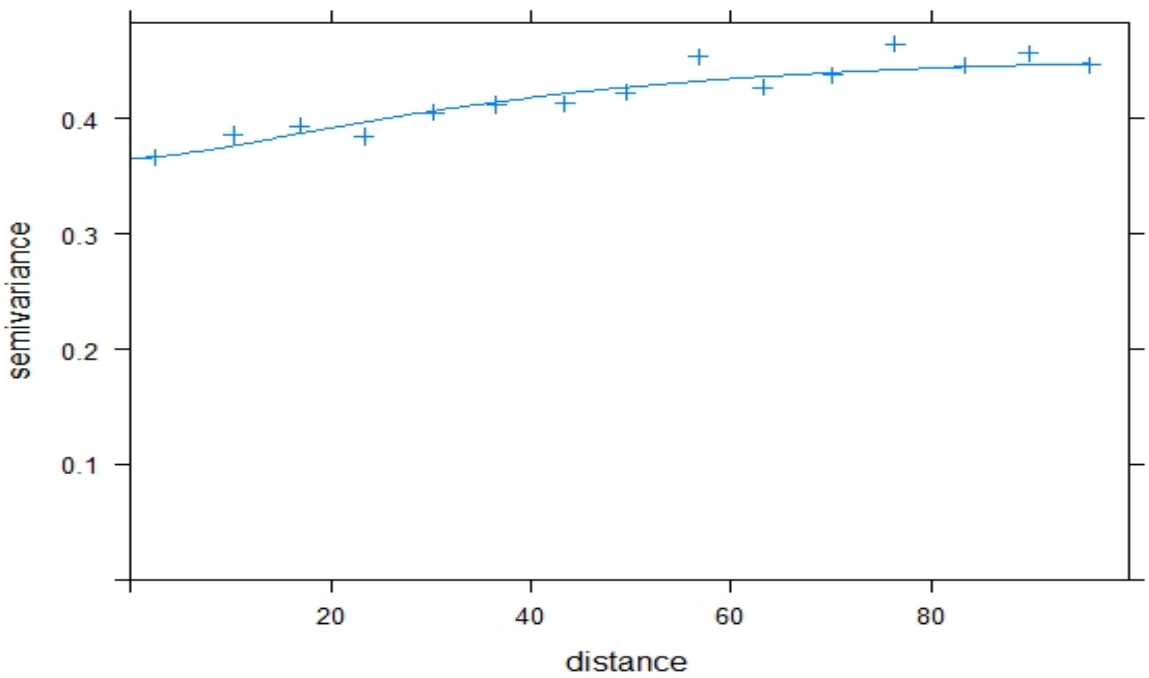
```
> MyData2 <- MyData1[(MyData1$latitude<=5400)&
+ (MyData1$latitude>=5300)& (MyData1$longitude<=-700)&
+ (MyData1$longitude>=-800),]
> spplot(MyData2,"price", do.log = T)
```

This process can be repeated for other windows (moving windows approach) to get the full region coverage.


[0.065,0.1965]
(0.1965,0.594]
(0.594,1.795]
(1.795,5.427]
(5.427,16.41]

For the selected window and data subset we estimate the variogram and fit the Bessel covariance model:

```
> v  <- variogram(log(price) ~ 1, MyData2, cutoff=100)
> plot(v)
> v.fit <- fit.variogram(v, vgm(0.4, "Bes", 1, 1))
> plot(v, v.fit, pch = 3)
```



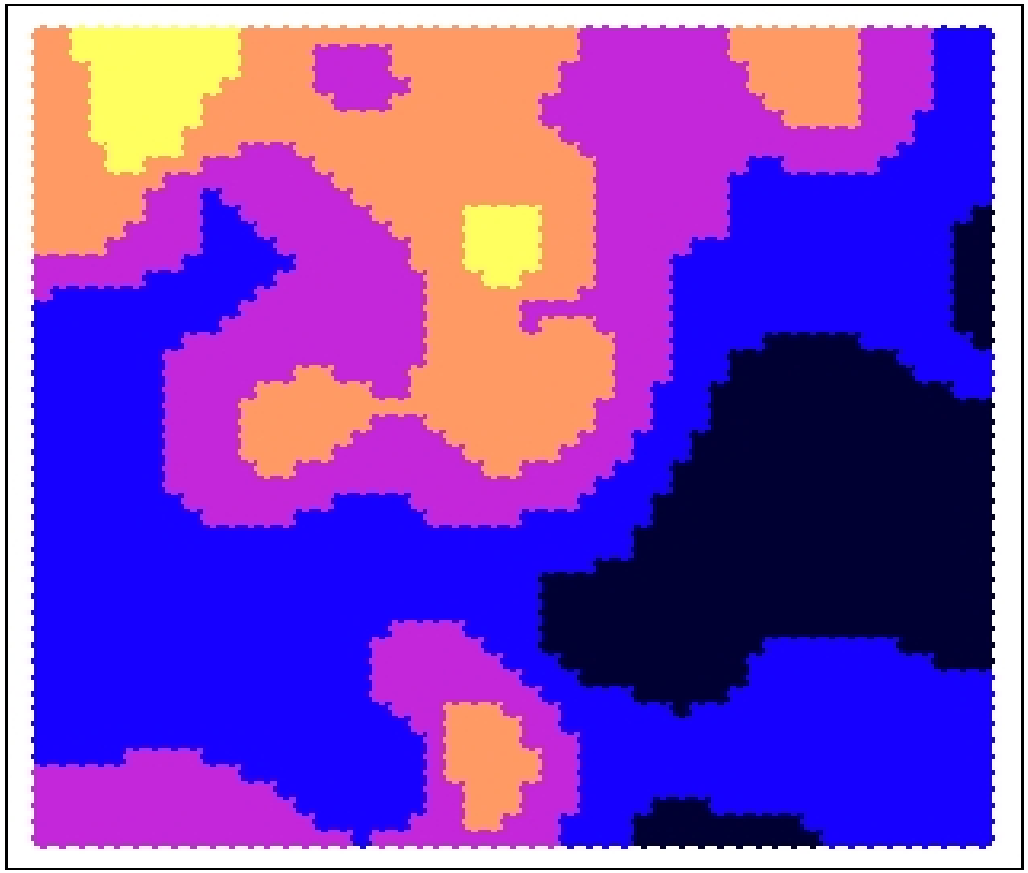
To spatially predict prices we cover the selected area by a grid 50x50 points and perform the ordinary kriging (!!!time lengthy computations).

```
> pred.grid <-  expand.grid(seq(5300,5400, l=51),
+ seq(-800,-700, l=51))
> colnames(pred.grid) <- c("latitude", "longitude")
> coordinates(pred.grid) <- ~latitude+longitude
> lz.uk <- krige(log(price)~1, MyData2, pred.grid, v.fit)
```

Finally we rescale the estimated prices to the original scale and plot a map with the obtained results.

```
> lz.uk$var1.pred <- exp(lz.uk$var1.pred)*100000
> spplot(lz.uk, c("var1.pred"))
```

The plot clearly shows areas with expensive and cheap houses.


[5.076e+04,6.906e+04]
(6.906e+04,8.736e+04]
(8.736e+04,1.057e+05]
(1.057e+05,1.24e+05]
(1.24e+05,1.423e+05]

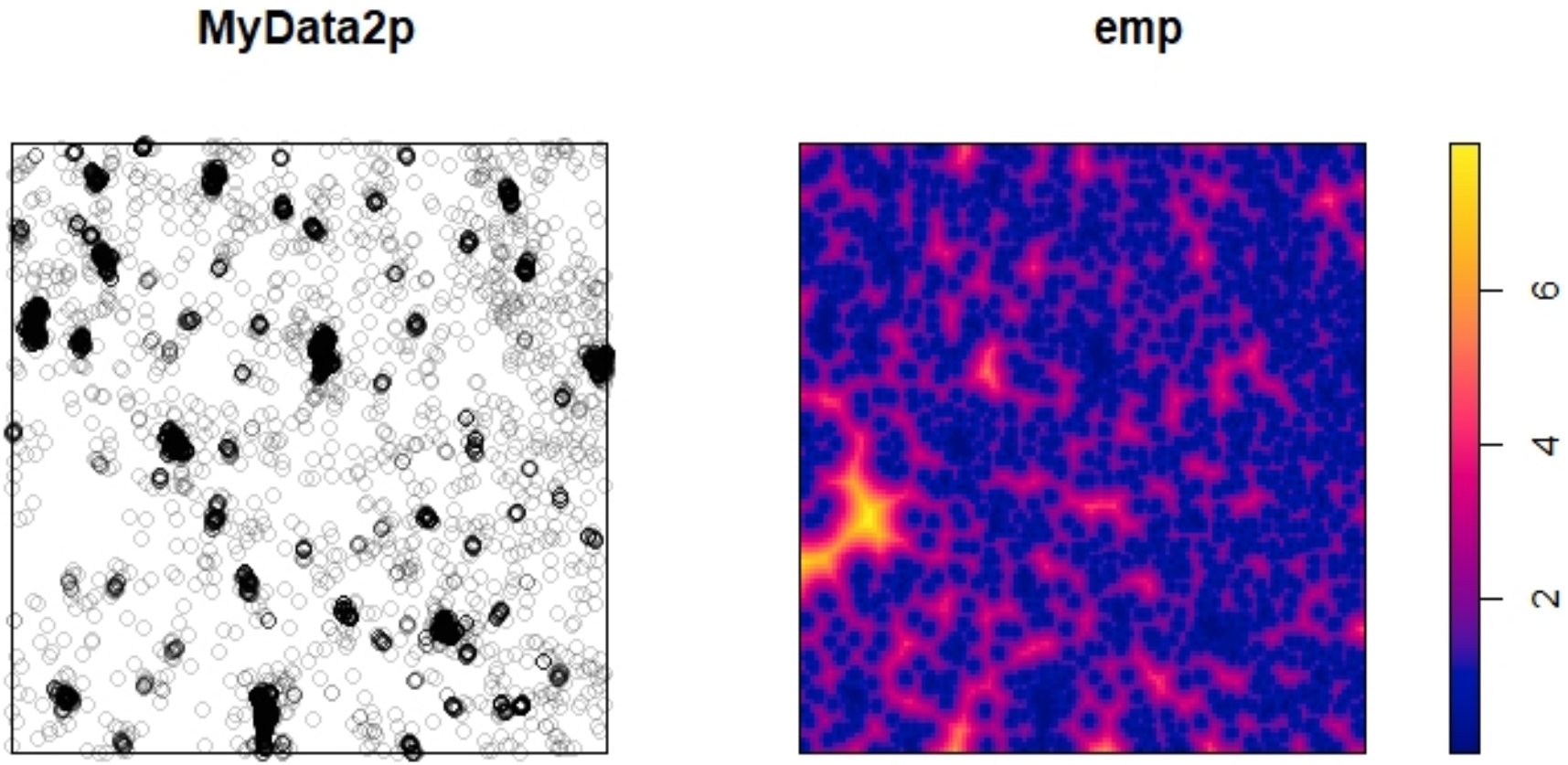
Analysis of spatial distribution of Irish sold properties

Now we continue the analysis of the selected subarea and investigate spatial pattern of the sold properties.
First we transform the data to the PPP format and remove all information except the coordinates.

```
> library(maptools)
> MyData2p<-as(MyData2,"ppp")
> MyData2p <- unmark(MyData2p)
```

Then we compute the estimated intensity of houses and visualise the distribution of their locations.

```
> summary(MyData2p)$intensity
[1] 0.827476
> plot(MyData2p)
> emp <- distmap(MyData2p)
> plot(emp)
```


MyData2p
emp
2
4
6

Finally we produce plots of the estimated F and G functions

```
> plot(Fest(MyData2p))
> plot(Gest(MyData2p))
```

The plots suggest that the locations of the sold properties have a cluster structure.