Read Population Data

1 Read Data

We read the population data downloaded from Statistisches Amt für Hamburg und Schleswig-Holstein. (2015, July 14). Interaktive Karte für Hamburg zum Zensus 2011. https://www.statistik-nord.de/fileadmin/maps/zensus2011_hh/index.html. We skip the first two rows because this is not data yet, but explanatory rows. We also get rid of the last two rows, since they're only indicating the source of the file and are not part of the dataset.

```
library(readxl)
library(spatstat)
```

```
population_data <- read_excel("./Data/Population Data/Zensus2011_Ergebnisse_nach_Gitterzellen_fuer_Hamburg
```

1.1 Data Cleaning

We remove all rows where the number of inhabitants is equal to "-1" because it means that the cell is either uninhabited or the number of inhabitants has to be kept secret.

```
population_data <- population_data[population_data$Einwohner != "-1", ]
```

1.2 Data Output

The attributes "x_mp_100m" and "y_mp_100m" are the geographical longitude and latitude of the centroid of the cell.

```
head(population_data)
```

```
## # A tibble: 6 x 6
##   OBJECTID CellCode          Gitter_ID_100m    x_mp_100m y_mp_100m Einwohner
##      <dbl> <chr>             <chr>                 <dbl>     <dbl>     <dbl>
## 1       90 100mN33727E43082 100mN33727E43082   4308250   3372750        23
## 2       91 100mN33727E43083 100mN33727E43083   4308350   3372750        19
## 3       92 100mN33727E43084 100mN33727E43084   4308450   3372750         5
## 4       93 100mN33727E43085 100mN33727E43085   4308550   3372750         3
## 5      147 100mN33730E43082 100mN33730E43082   4308250   3373050        14
## 6      165 100mN33731E43081 100mN33731E43081   4308150   3373150         3
```

2 Transform Data

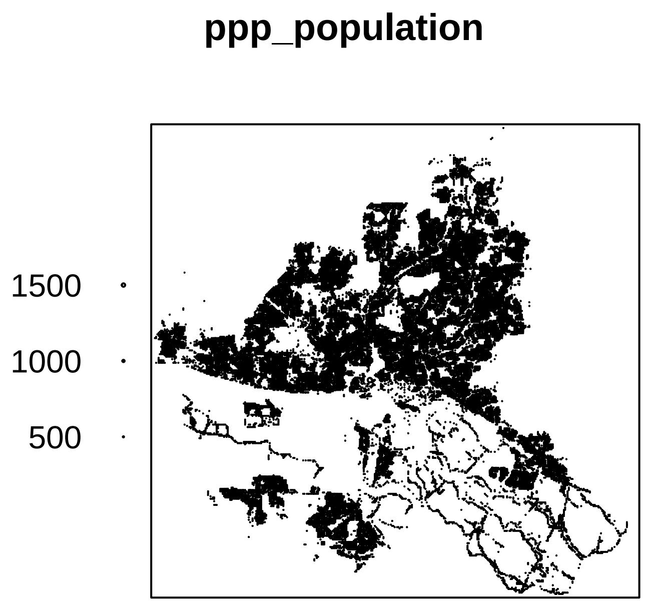
Now, we have an object of the class tbl_df, tbl, data.frame. To use it within spatstat, we first need to define the bounding window with the minimum and maximum x and y values. The values for the window object are taken from the full dataset also including the cell without inhabitants.

```
window <- owin(c(4303150,4342650), c(3365250,3403550))
```

Now, we can transform the data into a ppp-object (planar point pattern)

```
ppp_population <- ppp(x=population_data$x_mp_100m, y=population_data$y_mp_100m, window = window, marks= po
```

```
plot(ppp_population)
```



```
summary(ppp_population)
```

```
## Marked planar point pattern:  29094 points
## Average intensity 1.923125e-05 points per square unit
##
## Coordinates are multiples of 10 units
##
## marks are numeric, of type 'double'
## Summary:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    3.00   16.00   35.00   58.83   76.00 1969.00
##
## Window: rectangle = [4303150, 4342650] x [3365250, 3403550] units
##                     (39500 x 38300 units)
## Window area = 1512850000 square units
```

3 Smooth Data

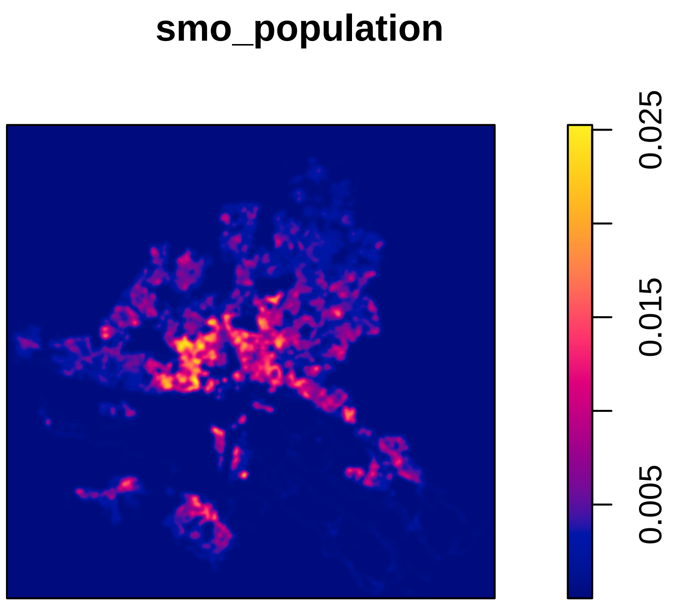
```
smo_population <- density(ppp_population, sigma = 150, positive = TRUE,  eps = 100, weights= marks(ppp_pop
```

For further explanation of the density function see: density.ppp function - RDocumentation

- `Sigma`: smoothing bandwidth

    - Standard deviation of isotropic smoothing kernel. Either a numerical value, or a function that computes an appropriate value of sigma.
    - this is just an informed pick for the bandwidth, there are techniques for automatic bandwidth selection that we could use.
- `eps`: The pixels are eps=100 wide
- `weights`: Optional weights to be attached to the points. A numeric vector, numeric matrix, an expression, or a pixel image.

```
plot(smo_population)
```



```
smo_population
```

```
## real-valued pixel image
## 383 x 395 pixel array (ny, nx)
## enclosing rectangle: [4303200, 4342600] x [3365200, 3403600] units
```

3.1 Save the data

```
saveRDS(smo_population,"./Data/Population Data/im_population.rds")
```