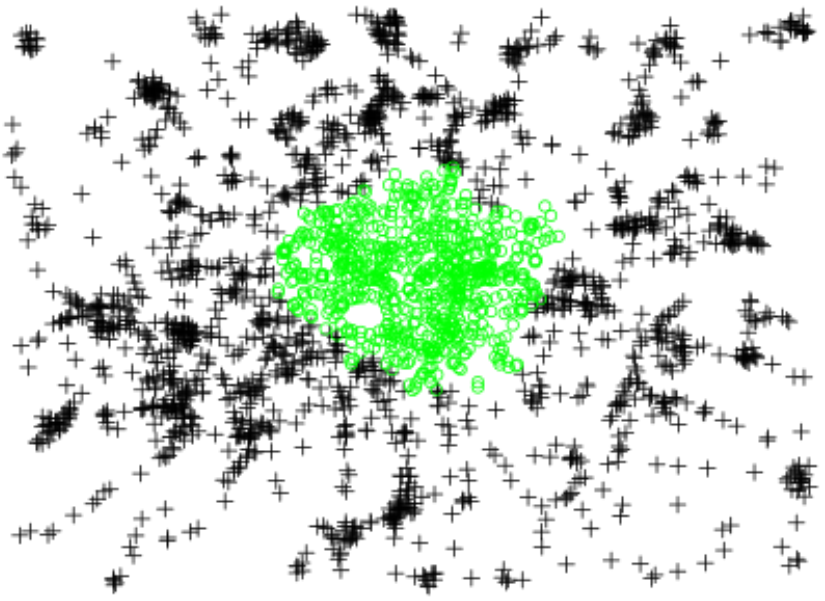

```
stations <- stations.cl
rm(stations.cl)  # tidy up: we're only interested in clipped
ones
```

The first line instructs R to look at each column (`MARGIN = 2`, we would use `MARGIN = 1` for row-by-row analysis) and report back whether `all` of the values are false. This creates the inverse selection that we want, hence the use of `!` to invert it. We test that the function works on a new object (often a good idea, to avoid overwriting useful data) with plots and, once content that the clip has worked, save the sample of points to our main `stations` object and remove the now duplicated `stations.cl` object.

Aggregating the data to complete the spatial join

Now that we know how `gIntersects` works in general terms and for clipping, let's use it to allocate a borough to each of our station points, which we will then aggregate up. Data from these points (e.g. counts, averages in each area etc.) can then be transferred to the main polygons table: the essence of a spatial join. Again, `apply` is our friend in this instance, meaning we can avoid `for` loops: