# `RapidPolygonLookup`: An R package for polygon lookup using kd trees

Markus Loecher     Madhav Kumar

**Abstract**

Coordinate level spatial data need to be frequently aggregated to higher geographical identities like census blocks, ZIP codes or police district boundaries for analysis. This process requires mapping each point in the given data set to an individual element of the desired geographical hierarchy. Unless efficient data structures are used, this can be a daunting task. The operation point.in.polygon() from the package sp Pebesma and Bivand (2005) is computationally expensive. Here, we exploit kd-trees as an efficient nearest neighbor search algorithm to dramatically reduce the effective number of polygons being searched and expedite the lookup process.

**Keywords:** polygon lookup, kd-trees, nearest neighbour, spatial, census, R

## 1 Introduction

We are often faced with the task of aggregating spatially dispersed data points to higher geographical entities such as ZIP codes, census blocks, or districts with boundaries that are not simply rectangular. Associating individual points to larger spatial authorities often makes the analyses more representative and revelant helping derive meaningful insights from the data. However, mapping points to polygons is a computationally expensive exercise that typically involves looking up each data point in multiple spatial polygons until the search criteria are met. This can become a daunting task if the number points to be mapped, which we denote by $N$, go beyond a few thousand. In particular, the overall computational cost is expected to be of $O(N \cdot M)$ for $M$ polygons with a very high multiplicative constant.

In this package we try to expedite the search process through efficient data structures by exploiting kd-trees as a fast approximate nearest neighbour algorithm to dramatically reduce the number of polygons being searched.

This paper is organized as follows: In the next section we describe the basic idea behind our approach. Section 3 gives a brief introduction to the data used in the paper. Section 4 provides code and exmaples for the package and section 5 concludes the paper.

## 2 Approach

Our approach can be broadly categorized into the four steps described below:

1. Compute the centroids for the $M$ polygons and represent them in a kd-tree data structure using the package RANN Arya, Mount, Kemp, and Jefferis (2013).

2. Find the $n$ nearest centroids for each data point using the function nn2() from the package RANN.

3. Execute point.in.polygon() on this ordered list to find a match.

4. For those points that remain unmapped, a new search strategy is executed by excluding those polygons as candidates whose bounding box does not contain the current point.

## 3 Data

The data used for examples in this paper and the package are crime incidents from 2012 for the city of San Francisco. Since the examples here are for illustration, we have subset the data in terms of number of rows and columns. The original data set contains information for about 120,000 on time, date, day of week, category of crime, text based description, geographical coordinates, and the police district. In the package, we have included randomly selected 20,000 from the data set along with the following three columns – date, x coordinate, and y coordinate. In addition, we have included one more column indicating whether the crime was violent or not. Incidents from the following categories have been defined as violent – assault, robbery, rape, kidnapping, and purse snatching.

Fig 1 visualizes the crime incidents on the map of San Francisco. The map data are fetched through a query to the Google server for static maps using the GetMap() function from the RgoogleMaps Loecher (2013) package. The points are overlayed on the map by using it as a background image.

```
> library(RgoogleMaps)
> library(RapidPolygonLookup)
> data(sf.crime.2012)
> if (0) {
+   sf.map <- GetMap(center=c(lat = 37.77173, lon = -122.4306), destfile = "sanFrancisco.png",
+                    size=c(550,550), GRAYSCALE = FALSE, zoom = 13, verbose= FALSE)
+ } else {
+   sf.map = ReadMapTile("SanFrancisco.png")
+ }
> color <- ifelse(sf.crime.2012$violent == "TRUE", "red", "green4")
> PlotOnStaticMap(MyMap= sf.map, lat= sf.crime.2012$Y, lon= sf.crime.2012$X,
+                 col= color, cex= 0.35, pch= 20)
```

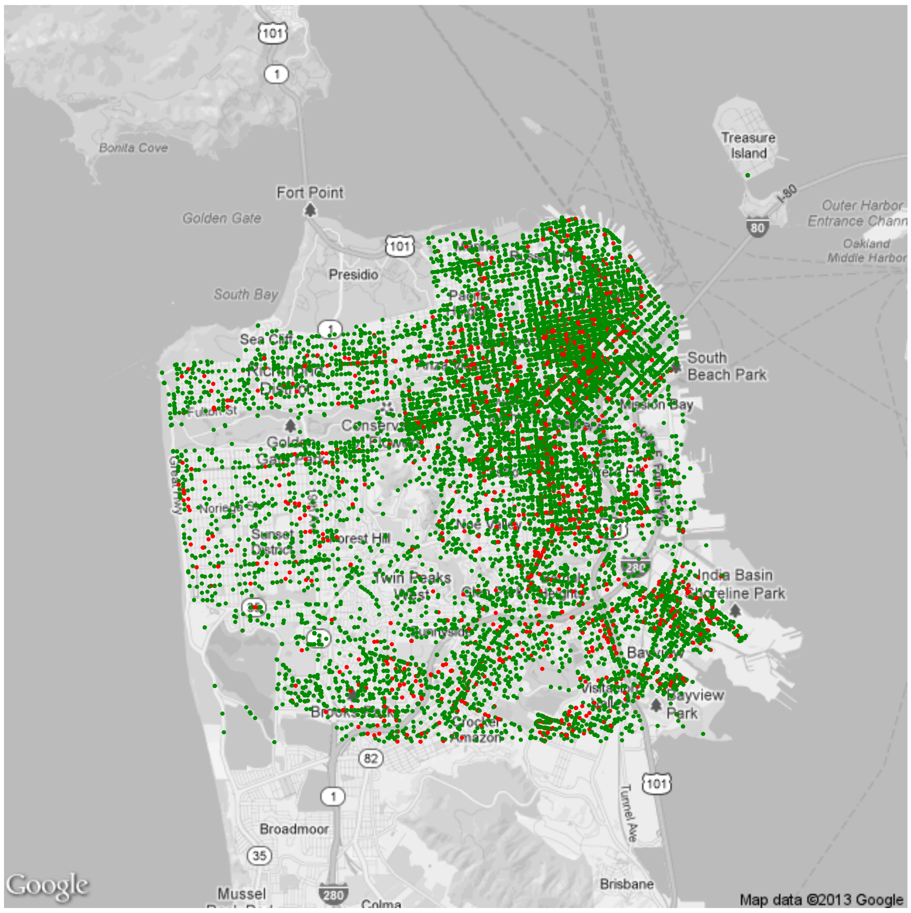

Figure 1: Crime in San Francisco classified as violent (red) and non-violent (green)

## 4 Polygon Lookup

We use crime incidents in San Francisco as the geographical points that need to be associated to a certain set of spatial polygons. As an example, we take the Census tracts from the 2010 US Census as the polygons. This specific example can be easily generalized to other spatial hierarchies as well.

The Census tracts are obtained from the UScensus2010 Almquist (2010) package. The tracts are provided as objects of class SpatialPolygonsDataFrame. To initiate the lookup process, we first change the structure of the spatial polygons (from the sp Pebesma and Bivand (2005) package) to the one supported by PBSmapping Schnute, Boers, Haigh, Grandin, Johnson, Wessel, and Antonio (2013) and then clip the polygons to retain the tracts from San Francisco only. We then calculate the centroids of the polygons. All these actions are completed using the function CropSpatialPolygonsDataFrame(). In order to keep the size of the package managable, we have provided pre-cropped tracts that include San Francisco and areas nearby it. Fig 2 displays the pre-cropped polygons for California and Fig 3 provides a view of transformmed polygons for San Francisco only.

```
> data(california.tract10)
> plot(california.tract10)
```

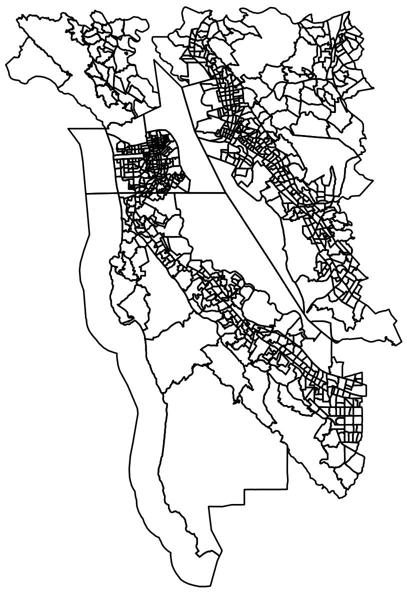

Figure 2: Subset of Census 2010 tracts from the state of California

```
> sf.polys <- CropSpatialPolygonsDataFrame(x= california.tract10,
+                                          bb= data.frame(X=c(-122.5132, -122.37),
+                                                         Y= c(37.70760, 37.81849)))
> str(sf.polys, max.level =1)

List of 3
 $ data        :'data.frame':        204 obs. of  461 variables:
 $ polys       :Classes ‘PolySet’ and 'data.frame':        12855 obs. of  5 variables:
 $ poly.centers:Classes ‘PolyData’ and 'data.frame':        204 obs. of  3 variables:

> plotPolys(sf.polys$polys)
```

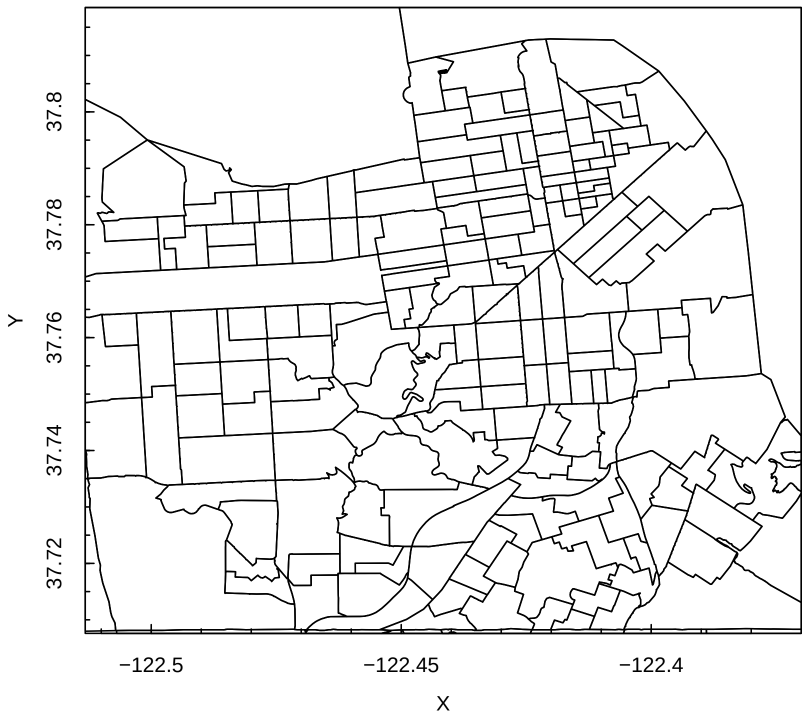

Figure 3: Census tracts cropped around San Francisco

After reorganizing the polygon structure, we calculate $n$ neareast neighbours for all the points in the data using the nn2() function from the RANN Arya et al. (2013) package. The neighbours are computed using the coordinates of the point and the centroids of the polygon. The nn2() function uses kd-trees to find the requisite number of neighbours.

```
> XY.kdtree <- RapidPolygonLookup(sf.crime.2012[,c("X","Y")], poly.list= sf.polys,
+                                  k= 10, N= 5000,
+                                  poly.id= "fips", poly.id.colname= "census.block",
+                                  keep.data= TRUE, verbose= FALSE)
> table(XY.kdtree$XY$rank, useNA= "always")

   1    2    3    4    5    6    7    8    9   10 <NA>
3383  930  333  134   90   50   36   11   25    8    0

> hist(XY.kdtree$XY$rank, xlab = "Rank of neighbor",
+      main= "Histogram of number of polygons searched")
```

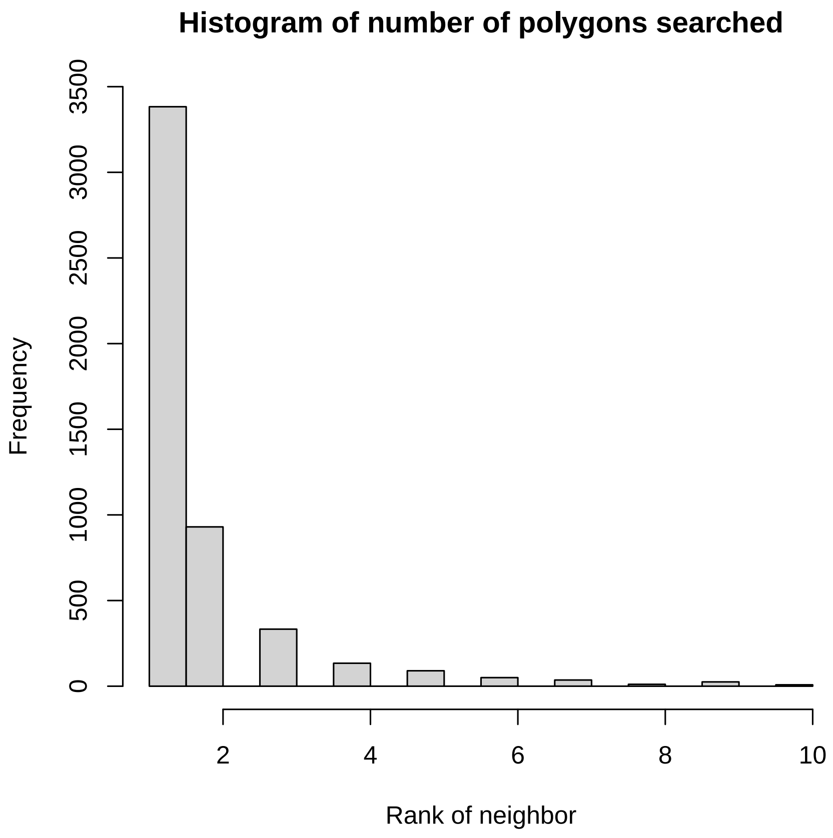

Figure 4: No. of polygons searched before a match was found

In the example above, we use the nn2() function to find $n = 10$ approximate nearest neighbours and then run point.in.polygon() from the sp package on this ordered list of polygons. Since the nearest neighbour calculation is approximate, it is possible that a data point may not belong to any of the short-listed polygons. This is especially true for points that lie on the edges of the polygon. To account for this possibility, we perform an alternative search across the polygon space only for the points that were left unmapped which excludes those polygons as candidates whose bounding box does not contain the current point. This process is built into the RapidPolygonLookup() function and does not need to be executed separately. There possibility to optimize the tradeoff between the two search strategies by a careful choice of $n$ does exist.

The output of the RapidPolygonLookup() function is a list whose first element is a data frame with the original data set, the ID of the corresponding polygon to which the point was mapped, and the number of polygons searched before a match was found for the point. Fig 4 shows the historgram of the number of polygons that had to be searched for mapping each crime incident to a particular Census tract. The other elements of list contain the call information from the function.

Once the points are mapped to the Census tracts, demographic information from the corresponding tract can be fetched using the data component of the *sf.polys* object. Such information can then be used to contextualize the crime incidents better by making references to the area, population, and other demographic metrics. In addition, various demographic and socio-economic variables can be used as features for spatial models of crime.

## 5 Conclusion

In this package, we use efficient data structures to aggregate spatially dispersed points to higher geographical entities. The process is expedited by exploiting kd-trees as an approximate nearest neighbours algorithm to considerably reduce the number of polygons searched. Using polygon centroids and coordinate points, we use the nn2() function from the RANN Arya et al. (2013) package to determine the nearest neighbours and then run point.in.poylgon() from the sp Pebesma and Bivand (2005) package on these ordered polygons.

## References

Z. W. Almquist. Us census spatial and demographic data in R: TheUScensus2000 suite of packages. *Journal of Statistical Software*, 37(6):1–31, 2010. URL `http://www.jstatsoft.org/v37/i06/`.

S. Arya, D. Mount, S. E. Kemp, and G. Jefferis. *RANN: Fast Nearest Neighbour Search (wraps Arya and Mount's ANN library)*, 2013. URL `http://CRAN.R-project.org/package=RANN`. R package version 2.3.0.

M. Loecher. *RgoogleMaps: Overlays on Google map tiles in R*, 2013. URL `http://CRAN.R-project.org/package=RgoogleMaps`. R package version 1.2.0.5.

E. J. Pebesma and R. S. Bivand. Classes and methods for spatial data in R. *R News*, 5(2):9–13, November 2005. URL `http://CRAN.R-project.org/doc/Rnews/`.

J. T. Schnute, N. Boers, R. Haigh, C. Grandin, A. Johnson, P. Wessel, and F. Antonio. *PBSmapping: Mapping Fisheries Data and Spatial Analysis Tools*, 2013. URL `http://CRAN.R-project.org/package=PBSmapping`. R package version 2.66.53.