Final Project

CS110: Computation: Solving Problems with Algorithms

Kalia Barkai

December 2017

R-Trees: Introduction

R-Trees are a type of data structure used to store and search spatial indices. The advantage of searching through R-Trees for values, is that it is a multi-dimensional data structure and therefore, can find the values within a certain spatial area - and the values near it.

It does this by storing the key, along with it's tuple coordinates, in rectangles (i.e. areas), which are child nodes of greater rectangles (i.e. the encompassing areas). The tree structure is similar to that of a B+-Tree, where it is balanced by having equal lengths of the paths from the root to all leaf nodes, and each node can have 0 to N children nodes. N is the decided upon max number of sibling nodes and this applies to the root level too.

The following are the invariants of an R-Tree:

1. Every node (non-leaf) has between n and N children (except for root), where $n \leq \frac{N}{2}$
2. The root has at least two children, unless it is a leaf
3. Each node rectangle is the smallest rectangle that contains its children
4. All leaves are on the same level (balanced tree)

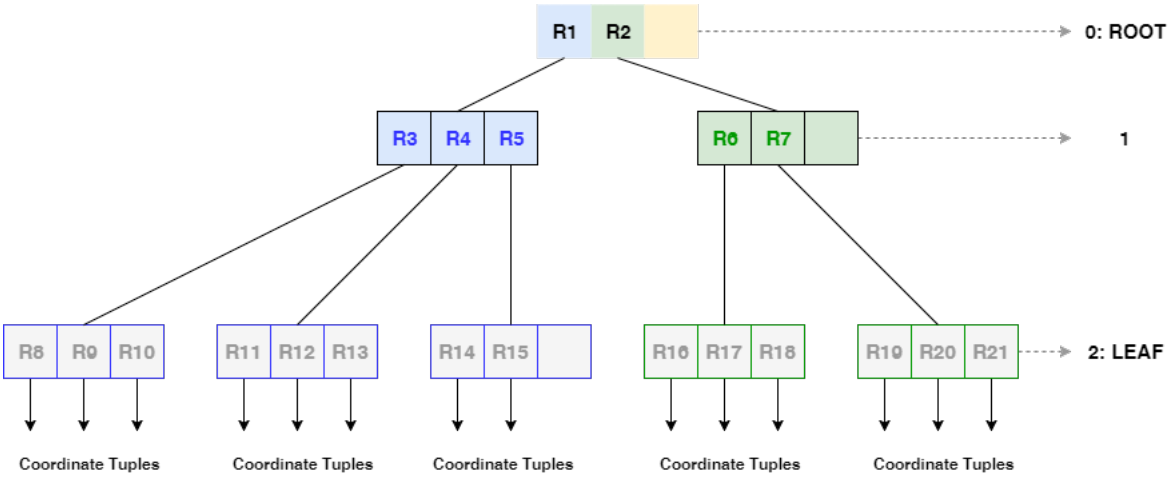


Figure 1. R-Tree example with 21 rectangles. N = 3

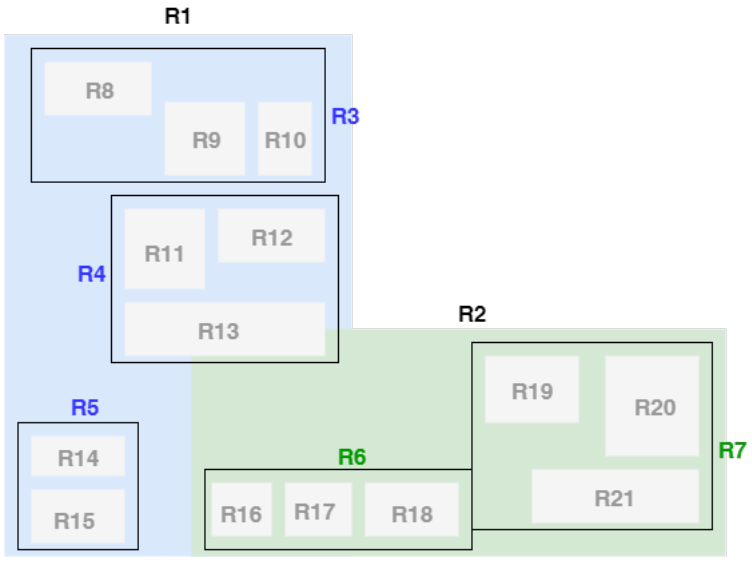


Figure 2. R-Tree rectangles example from R-Tree in Figure 1.

The R-Tree datastructure supports the following operations: insert, search and deletion. Search can be implemented in two more ways: range search or priority search. Range search finds all the values within a rectangle/area, while priority search finds a specific value. Priority search might be used for a nearest neighbour search for example.

Range search happens recursively where any rectangle that intersects with the search rectangle is searched until it reaches the leaf nodes, which are then checked against the main search rectangle.

Insertion also happens recursively from the root node. Each rectangle is checked by seeing which rectangle would need the least enlargement for the insertion to take place, it then goes into this rectangle checking the child nodes in the same way until it reaches a leaf node where, if there is space, the value is inserted in the leaf node, and if there is no space, then the leaf node is split.

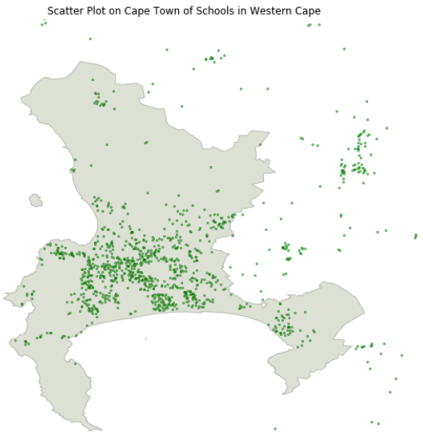
Deletion either updates the bounding boxes of parent rectangles in order to rebalance, however if the R-Tree is not full, then it will be deleted and each value will be reinserted into the tree.

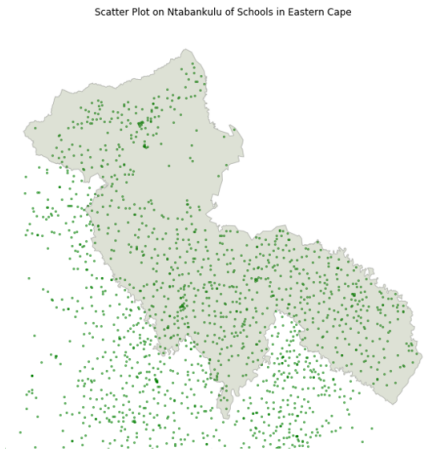
The average case time complexity for search in an R-Tree is $T(n) = O(\log n)$, where n refers to the number of rectangles/nodes. This assumes that there is not too much overlap in the tree and only a minimal amount of paths must be followed for the search.

R-Trees: Application

Using the schools database from the Department of Basic Education of the Republic of South Africa which is divided up by provinces, Cape Town and Ntabankulu student to teacher ratios at each school were compared.[1]

Since the full dataset is divided by provinces and not by cities, but includes coordinates for each school, it is possible to use spatial indexing in R-Trees to find which schools fall within chosen cities.





Figures 3. and 4. Scatter plot of schools in Cape Town and in Ntabankulu and surrounding area. Created using Geopandas and Shapely libraries in Python. Schools database from the Department of Basic Education, RSA (2016).

Listing 1. below shows the code which implements search in the R-Tree of the Western Cape school nodes.[2]

```
# network of schools coordinates
gdf_nodes = gpd.GeoDataFrame(data={'x':x, 'y':y})
# system of starting coordinates (dict)
gdf_nodes.crs = ct.crs
```

[1] Cape Town is in the top 3 richest cities in South Africa while Ntabankulu (part of the Alfred Nzo Municipality) is in the poorest 3 cities.

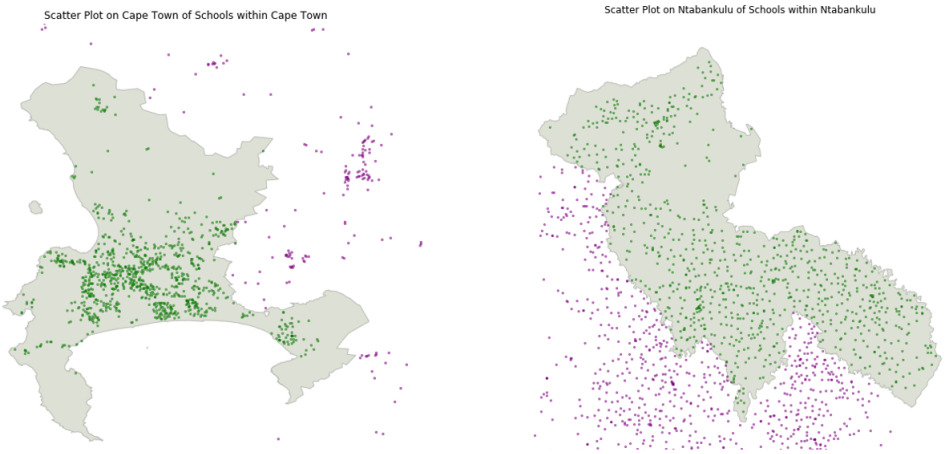
[2] The rest of the code can be found in the Appendix.

```
# value is the ratio of learners per educator
gdf_nodes.name = lpe
# create 'geometry' col with coordinates
gdf_nodes['geometry'] = gdf_nodes.apply(lambda row:
    Point((row['x'], row['y'])), axis=1)

spatial_index = gdf_nodes.sindex # spatial index
# check which spatial indices intersect with our
    bounding box of the polygon (our city)
# possible, because there are no false negatives,
    but there can be false positives
# this happens when the bounding box in the R-Tree
    of the polygon has bounds greater than the
    polygon bounds
# in this case points outside the city but in this
    bounding box are false positives
possible_index = list(spatial_index.intersection(
    geometry.bounds))
# fetch nodes of possible schools
possible_schools = gdf_nodes.iloc[possible_index]
# check if the possible schools intersect with our
    actual polygon
# this gives us the actual schools within
schools_ct = possible_schools[possible_schools.
    intersects(geometry)]
```

Listing 1: Code which implements spatial indexing with R-Trees using the Geopandas Python library.

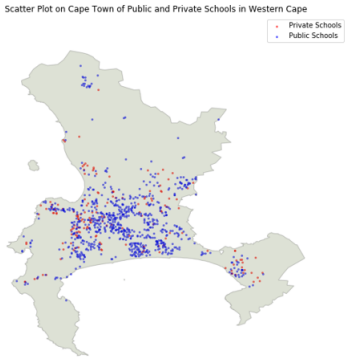
Figures 5. and 6. show the visualisations created by finding which schools lie within the cities and which do not.

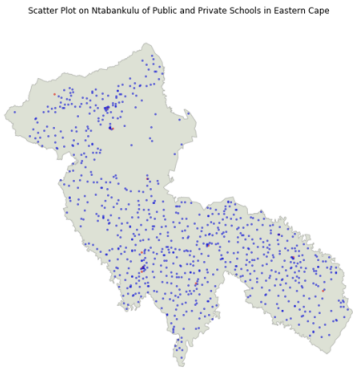


Figures 5. and 6. Scatter plot of schools within Cape Town and in Ntabankulu and surrounding area. R-Tree spatial indexing used to find points within the cities. Created using Geopandas and Shapely libraries in Python. Schools database from the Department of Basic Education, RSA (2016).

Using the values of the schools within the cities, we can calculate whether there is a significant difference between the amount of students per educators at each school. First, the mean for schools in Cape Town was calculated to be 30.055, while the mean for schools in Ntabankulu was calculated to be 32.622. However, schools in Cape Town have a range of 3 to 67 students per educator, while schools in Ntabankulu have a range of 5 to 83 students per educator. To calculate whether there is actually a significant difference, it is important to look at the p-value between the two cities. Using scipy.stats library in Python, the p-value = 0.00000043. Since this is ≤ 0.05, there is a significant difference. Therefore, only looking at the mean of this data, might be misleading.

For further analysis we can also look at a visualisation of the private and public schools within each city. This can be seen in Figures 7. and 8. below.





Figures 7. and 8. Scatter plot of public and private schools in Cape Town and in Ntabankulu and surrounding area. Created using Geopandas and Shapely libraries in Python. Schools database from the Department of Basic Education, RSA (2016).

As seen in Figures 7. and 8., Cape Town has more private schools. This is just one further analysis of the two regions. More analysis on the populations of the areas, the schools budget and resources and the socio-economic disposition of the communities surrounding the schools would be relevant to further understand the reason for the significant difference between students per educator in each region.

Appendix

Notebook of all code available on here.

```
import osmnx as ox, matplotlib.pyplot as plt, pandas as pd, geopandas as gpd
from descartes import PolygonPatch
from shapely.geometry import Point, Polygon, MultiPolygon
%matplotlib inline
# for high quality visualisations:
%config InlineBackend.figure_format = 'svg'

# boundary of Cape Town
ct = ox.gdf_from_place('City of Cape Town, South Africa')

# x and y bounds
west, south, east, north = ct.unary_union.buffer(0.1).bounds

# geometry into polygon
geometry = ct['geometry'].iloc[0]
if isinstance(geometry, Polygon):
    geometry = MultiPolygon([geometry])

# Western Cape school database
wc_schools = pd.read_excel("wc.xlsx")
```

```
21
22 # Latitude coordinate of school
23 y = wc_schools["GIS_Latitude"]
24 # Longitude coordinate of school
25 x = wc_schools["GIS_Longitude"]
26
27 # amount of learners in school
28 learners = wc_schools["Learners_2016"]
29 # amount of educators in school
30 educators = wc_schools["Educators_2016"]
31 lpe = [] # learners per educator
32 for i in range(len(learners)):
33     lpe.append(learners[i]/educators[i])
34
35 gdf_nodes = gpd.GeoDataFrame(data={'x':x, 'y':y})
36 gdf_nodes.crs = ct.crs
37 gdf_nodes.name = lpe
38 gdf_nodes['geometry'] = gdf_nodes.apply(lambda row:
     Point((row['x'], row['y'])), axis=1)
39
40 fig, ax = plt.subplots(figsize=(10,10))
41 for polygon in geometry:
42     patch = PolygonPatch(polygon, fc='#556B2F', ec='
    k', alpha=0.2, zorder=2)
43     ax.add_patch(patch)
44
```

```
plt.title("Scatter Plot on Cape Town of Schools in
    Western Cape")
ax.scatter(x=gdf_nodes['x'], y=gdf_nodes['y'], s=5,
    c='g', zorder=1, alpha = 0.5) # plotting schools
ax.set_xlim(west, east)
ax.set_ylim(south, north)
ax.axis('off')
plt.savefig("ct_1.png")
plt.show()
```

Listing 2: Code used to create Figure 3

```
# Compare to Ntabankulu in Eastern Cape
# repeat process
nt = ox.gdf_from_place('Alfred Nzo District
    Municipality, South Africa')

# Eastern Cape schools database
ec = pd.read_excel("ec.xlsx")

y_ec = ec["GIS_Latitude"]
x_ec = ec["GIS_Longitude"]

learners_ec = ec["Learners_2016"]
educators_ec = ec["Educators_2016"]
lpe_ec = []
for i in range(len(learners_ec)):
    lpe_ec.append(learners_ec[i]/educators_ec[i])
```

```
16
17 geo_nt = nt['geometry'].iloc[0]
18 if isinstance(geo_nt, Polygon):
19     geo_nt = MultiPolygon([geo_nt])
20
21 gdf_nodes_ec = gpd.GeoDataFrame(data={'x':x_ec, 'y':
       y_ec})
22 gdf_nodes_ec.crs = nt.crs
23 # value is the ratio of learners per educator
24 gdf_nodes_ec.name = lpe_ec
25 gdf_nodes_ec['geometry'] = gdf_nodes_ec.apply(lambda
       row: Point((row['x'], row['y'])), axis=1)
26
27 west_nt, south_nt, east_nt, north_nt = nt.
       unary_union.buffer(0.1).bounds
28
29 fig, ax = plt.subplots(figsize=(10,10))
30 for polygon in geo_nt:
31     patch = PolygonPatch(polygon, fc='#556B2F', ec='
       k', alpha=0.2, zorder=2)
32     ax.add_patch(patch)
33
34
35 plt.title("Scatter Plot on Ntabankulu of Schools in
       Eastern Cape")
36 ax.scatter(x=gdf_nodes_ec['x'], y=gdf_nodes_ec['y'],
       s=5, c='g', zorder=1, alpha = 0.5)
```

```
ax.set_xlim(west_nt, east_nt)
ax.set_ylim(south_nt, north_nt)
ax.axis('off')
plt.savefig("nt_1.png")
plt.show()
```

Listing 3: Code used to create Figure 4

```
# schools outside Cape Town
schools_not = gdf_nodes[~gdf_nodes.isin(schools_ct)]

fig, ax = plt.subplots(figsize=(10,10))
for polygon in geometry:
    patch = PolygonPatch(polygon, fc='#556B2F', ec='k', alpha=0.2, zorder=2)
    ax.add_patch(patch)

plt.title("Scatter Plot on Cape Town of Schools within Cape Town")
ax.scatter(x=schools_not['x'], y=schools_not['y'], s=5, c='purple', zorder=1, alpha = 0.5)
ax.scatter(x=schools_ct['x'], y=schools_ct['y'], s=5, c='g', zorder=2, alpha = 0.5)

ax.set_xlim(west, east)
ax.set_ylim(south, north)
ax.axis('off')
plt.savefig("ct_3.png")
```

```
plt.show()
```

Listing 4: Code used to create Figure 5

```
spatial_index_ec = gdf_nodes_ec.sindex
possible_index_ec = list(spatial_index_ec.
    intersection(geo_nt.bounds))
possible_schools_ec = gdf_nodes_ec.iloc[
    possible_index_ec]
schools_nt = possible_schools_ec[possible_schools_ec
    .intersects(geo_nt)]

schools_not_nt = gdf_nodes_ec[~gdf_nodes_ec.isin(
    schools_nt)]

fig, ax = plt.subplots(figsize=(10,10))
for polygon in geo_nt:
    patch = PolygonPatch(polygon, fc='#556B2F', ec='
    k', alpha=0.2, zorder=2)
    ax.add_patch(patch)

ax.scatter(x=schools_not_nt['x'], y=schools_not_nt['
    y'], s=5, c='purple', zorder=1, alpha = 0.5)
ax.scatter(x=schools_nt['x'], y=schools_nt['y'], s
    =5, c='g', zorder=2, alpha = 0.5)

plt.title("Scatter Plot on Ntabankulu of Schools
    within Ntabankulu")
```

```
ax.set_xlim(west_nt, east_nt)
ax.set_ylim(south_nt, north_nt)
ax.axis('off')
plt.savefig("nt_3.png")
plt.show()
```

Listing 5: Code used to create Figure 6

```
# public or independent school
sector = wc_schools["Sector"]

private_lon = []
private_lat =[]
public_lon = []
public_lat = []

for i in schools_ct.index:
    # private school coordinates
    if sector[i] == "INDEPENDENT":
        private_lon.append(x[i])
        private_lat.append(y[i])
    else: # public school coordinates
        public_lon.append(x[i])
        public_lat.append(y[i])


fig, ax = plt.subplots(figsize=(10,10))
for polygon in geometry:
```

```
    patch = PolygonPatch(polygon, fc='#556B2F', ec='k', alpha=0.2, zorder=2)
    ax.add_patch(patch)

plt.title("Scatter Plot on Cape Town of Public and Private Schools in Western Cape")
ax.scatter(private_lon, private_lat, s=5, c='r', zorder=1, alpha = 0.5, label = "Private Schools")
ax.scatter(public_lon, public_lat, s=5, c='b', zorder=1, alpha = 0.5, label = "Public Schools")
ax.legend()

ax.set_xlim(west, east)
ax.set_ylim(south, north)
ax.axis('off')
plt.savefig("ct_2.png")
plt.show()
```

Listing 6: Code used to create Figure 7

```
# public or independent school
sector_ec = ec["Sector"]

private_lon_ec = []
private_lat_ec =[]
public_lon_ec = []
public_lat_ec = []

```

```
for i in schools_nt.index:
    if sector_ec[i] == "INDEPENDENT":
        private_lon_ec.append(x_ec[i])
        private_lat_ec.append(y_ec[i])
    else:
        public_lon_ec.append(x_ec[i])
        public_lat_ec.append(y_ec[i])


fig, ax = plt.subplots(figsize=(10,10))
for polygon in geo_nt:
    patch = PolygonPatch(polygon, fc='#556B2F', ec='k', alpha=0.2, zorder=2)
    ax.add_patch(patch)

plt.title("Scatter Plot on Ntabankulu of Public and Private Schools in Eastern Cape")
ax.scatter(private_lon_ec, private_lat_ec, s=5, c='r', zorder=1, alpha = 0.5, label = "Private Schools")
ax.scatter(public_lon_ec, public_lat_ec, s=5, c='b', zorder=1, alpha = 0.5, label = "Public Schools")
ax.legend()

ax.set_xlim(west_nt, east_nt)
ax.set_ylim(south_nt, north_nt)
ax.axis('off')
```

```
plt.savefig("nt_2.png")
plt.show()
```

Listing 7: Code used to create Figure 8

```
# empty list of teacher-student ratio for schools in
    Cape Town
schools_tsr = []
for i in range(len(schools_ct)):
    index = schools_ct.index[i]
    schools_tsr.append(gdf_nodes.name[index])

schools_tsr_nt = []
for i in range(len(schools_nt)):
    index = schools_nt.index[i]
    schools_tsr_nt.append(gdf_nodes_ec.name[index])

from scipy.stats import ttest_ind
t, p = ttest_ind(tsr, tsr_nt)
print p
>> 4.27283063584e-07
```

Listing 8: Code used to calculate p-value of ratio of students per educator in Cape Town versus Ntabankulu.

References

"A Short Tutorial On B+ Tree". 2015. Sketching Dream. http://sketchingdream.com/blog/b-plus-tree-tutorial/. "B+ Tree". 2017. Wikipedia. https://en.wikipedia.org/wiki/B%2B_tree.

Boeing, Geoff. 2016. "R-Tree Spatial Indexing With Python - Geoff Boeing". Geoff Boeing. http://geoffboeing.com/2016/10/r-tree-spatial-index-python/.

Boeing, Geoff. 2017. "Rtree-Spatial-Indexing". Github. https://github.com/gboeing/urban-data-science/blob/master/19-Spatial-Analysis-and-Cartography/rtree-spatial-indexing.ipynb.

"EMIS Downloads". 2017. Education.Gov.Za. https://www.education.gov.za/Programmes/EMIS/EMISDownloads.aspx. Guttman, Antonin. 1984. "R-Trees". ACM SIGMOD Record 14 (2): 47. doi:10.1145/971697.602266.

"R-Tree". 2017. Wikipedia. https://en.wikipedia.org/wiki/R-tree. R-Tree

Chapter 1. 2008. Ebook. Itoma. http://www.bowdoin.edu/ ltoma/teaching/cs340/spring08/Papers/Rtree-chap1.pdf.