CS540 Project Presentation

Kevin Dumitrescu
For Professor Steven Lehr, CS540
ERAU Daytona Beach, Spring 2021

https://github.com/Dumitrek/CS540_flood_zone_project

Preliminary thoughts

- Due to recent climate changes and current average inches of rain statistics in Volusia county, the higher tier of flood zone you are close to decreases your property value
  - However, there's a contradiction when it comes to beachfront properties due to premiums associated to proximity to coastal waters.
- For this project, I have used scripting within Python
  - Rather straightforward. Very similar to the PgAdmin queries
- The flood zone distance and flood zone id value belong in the parcel table
  - Though I also included them in the sales_analysis table for the project as well

Step 1: Acquiring the Data

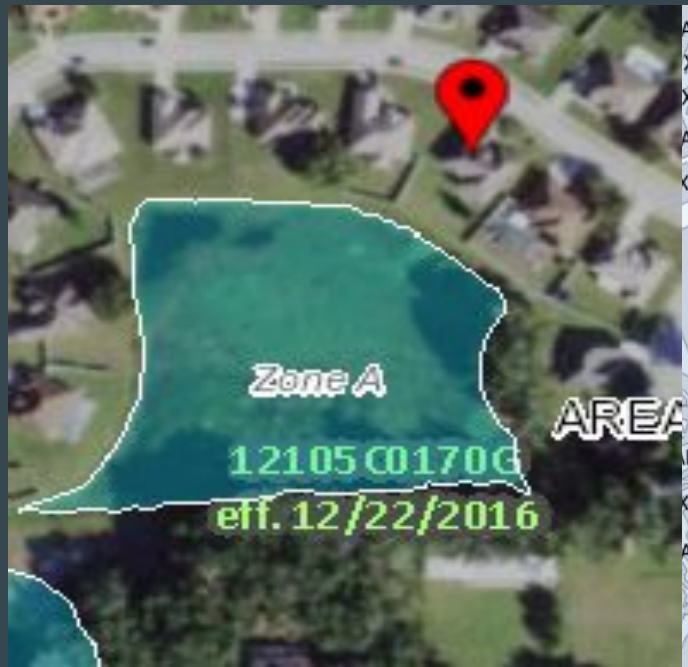
- Flood zone maps showcase flood zone data in a means of highlighting a body of water and labeling it with it's flood zone id.
  - However in my QGIS picture I made it similar to that of contour data on a contour map that uses a color grading scale.
  - Both pictures are compared side by side in the next slide.
- Download to Volusia County flood zone map:

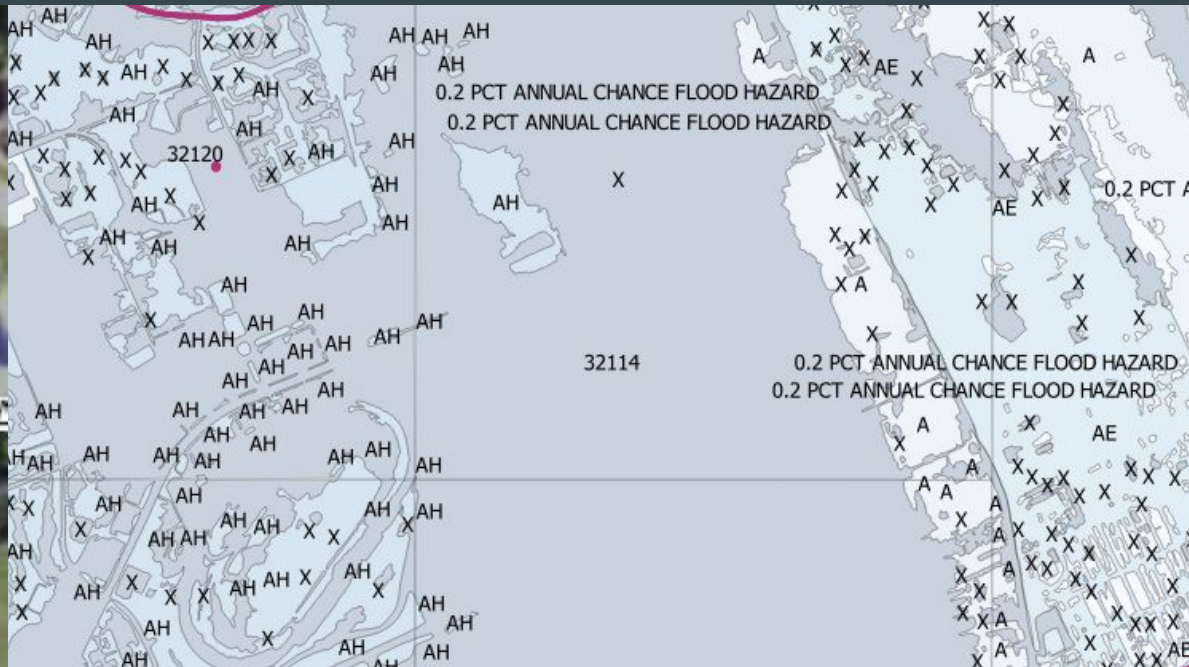
  http://maps.vcgov.org/gis/data/firmvc.htm

- Putting .shp file into QGIS (see slide 5)

Step 1: Acquiring the Data Cont.

- Left image: fema.gov flood zone map image
- Right image: QGIS flood zone map image





Step 1: Acquiring the Data Cont.

- In order to get the map loaded onto QGIS, these are the following steps:
- Download the zip file and unzip into your data folder
- Add the .shp file to QGIS like Professor Lehr showed us in class
  - Layer > Add Layer > New Vector Layer > Source > .shp file (Type: SHP file)
- Open Properties
  - Right click file > Properties
- Select Symbology
  - Categorized > Value: FLD_ZONE > Classify > Symbol: (your choice) > Color Ramp: (your choice) > Apply
- Select Labels
  - Single Labels > Value: FLD_ZONE > Apply

Step 1: Acquiring the Data Cont.

- In order to get the map loaded onto PgAdmin, these are the following steps:
- Open command prompt
- Change directory to where you unzipped the Volusia flood zone files
- Enter:
  - “c:\School\Postgres\bin\shp2pgsql.exe” -d -I -s 2236 -W “latin1” -g geom firmvc_poly.shp volusia.flood_zones > create_volusia_flood_zones.sql
  - “c:\School\Postgres\bin\psql.exe” -U postgres -d spatial -f create_volusia_flood_zones.sql
- Open or refresh the volusia schema and you should see it there

Step 2: Queries

- So how do we find the closest flood zone and distance to it for a each parcel in our parcel table?
- Answer KNN: “K Nearest Neighbour”
- What’s KNN?
    - According to PostGis’ official website, “KNN is a pure index based nearest neighbour search. By walking up and down the index, the search can find the nearest candidate geometries without using any magical search radius numbers...”[1].
- Since parcel id’s are “located” within flood zone “boxes” similar to that in the KNN documentation, this approach seems like the most apt to solving this project problem.

Step 2: Queries

- So first, we want to add give our parcel table a geometric aspect to it so that we can represent it on QGIS.
- These two queries add a geometry column to the parcel table and then updates the table using the already imported volusia.gis_parcels table we already have.

```
SELECT AddGeometryColumn ('volusia','parcel','geom',2236,'MULTIPOLYGON',2);
update volusia.parcel a set geom = p.geom from volusia.gis_parcels p where a.parid=p.altkey;
```

Step 2: Queries

- This is the query that we use to check the closest flood zone to a given parcel.
- It uses the ST_Distance to find distance from the parcel to the flood zone.
  - When we write the python script, the parid will automatically be filled in as it filters through the table.

```
--closest body of water to a random parcel
select p.gid, p.geom, p.fld_zone, ST_Distance(p.geom,(select p2.geom from volusia.parcel p2 where p2.parid=5213005))/5280
from volusia.fl_volusia_floodzones p
order by p.geom <->(select p2.geom from volusia.parcel p2 where p2.parid=5213005)
limit 1;
```

Step 2: Queries

- Next step is to alter the parcel table to that each parcel in the table can then hold a distance and nearest flood zone id value.
- We do this by using the alter table query to add a column of text for flood zone id (fzid) and a column of double precision for distance to said flood zone (fzdistance) to our parcel table.
- You will have to also do so for the geom column.

```
alter table volusia.parcel add column fzdistance double precision;
alter table volusia.parcel add columm fzid text;
```

```
SELECT AddGeometryColumn ('volusia','parcel','geom',2236,'MULTIPOLYGON',2);
update volusia.parcel a set geom = p.geom from volusia.gis_parcels p where a.parid=p.altkey;
```

Step 2: Queries

- The next queries is to now update our newly created columns with the data we have extrapolated.
- The first query updates the distance from the parcel to the flood zone, while the second updates the column to showcase the nearest flood zone's id.
    - As previously stated, when we write the python script the parid will automatically be filled in as we traverse through the parcel table.

```
update volusia.parcel p1 set fzdistance = ST_Distance(p1.geom, p2.geom)/5280 from volusia.fl_volusia_floodzones p2 where p1.parid=3324188
update volusia.parcel p1 set fzid =(p2.fld_zone) from volusia.fl_volusia_floodzones p2 where p1.parid=3324188
```

Step 2: Queries

- The final queries is to index our parcel table so that updating each parcel with the python script doesn't take that long.
  - Speed of updating is also dependent on your machine.

```
create index idx_parcel on volusia.parcel (parid);
create index idx_parcel_fzid on volusia.parcel(fzid);
create index idx_parcel_fzdistance on volusia.parcel(fzdistance);
```

- We also want to index our geometry.

```
CREATE INDEX parcel_geom_idx
  ON volusia.parcel
  USING GIST (geom);
```

Step 3: Python Script

- Now we move onto the Python Script in which the script will automatically go through the parcel table and update each parcel row with a flood zone id and it's proximity.
- It is to be noted that if you don't have pandas or psycopg2 installed onto your machine, you can do so by running these commands into your command prompt:
  - pip install psycopg2
  - pip install pandas

Step 3: Python Script

- In order to work with our database, we first have to connect to it and create our cursors.

```
# connection to database:
try:
    conn = psycopg2.connect("dbname='spatial' user='postgres' host='localhost' password='XXXX'")
except:
    print("cant connect to the database")

cur = conn.cursor()
cur2 = conn.cursor()
cur3 = conn.cursor()
cur4 = conn.cursor()
```

Step 3: Python Script

- Afterwards we then create our first sequel statement where we select our parid from our parcel table where they have a geometry associated to them.

```
sql = "select parid from volusia.parcel p where geom is not null " #limit 342930"

print('SQL: ', sql)
cur.execute(sql)
```

Step 3: Python Script

- Next we create a loop that iterates through all parid's in the parcel table.
- We also implement the fetchone() method so that we can get each following row, one after another.

```
i=0
row = cur.fetchone()
while row is not None:
    i = i + 1
    parid = str(row[0])#str(3324188)
```

Step 3: Python Script

- We then issue our first update for flood zone distance into the parid's fzdistance column.

```
fld_distance = row2[2]
gid = str(row2[0])
sql3 = "update volusia.parcel p1 set fzdistance = ST_Distance(p1.geom, p2.geom)/5280 from volusia.fl_volusia_floodzones p2 where p1.parid="+ parid +";"
cur3.execute(sql3)
print(sql3)
```

Step 3: Python Script

- We then issue our second update for flood zone id into the parid's fzid column.

```
fld_zone = row2[1]
sql4 ="update volusia.parcel p1 set fzid = (p2.fld_zone) from volusia.fl_volusia_floodzones p2 where p1.parid=" +parid + ";"
cur4.execute(sql4)
print(sql4)
```

Step 3: Python Script

- We then set a condition for our iteration for our while loop and when the condition is met, we close our connection to server.

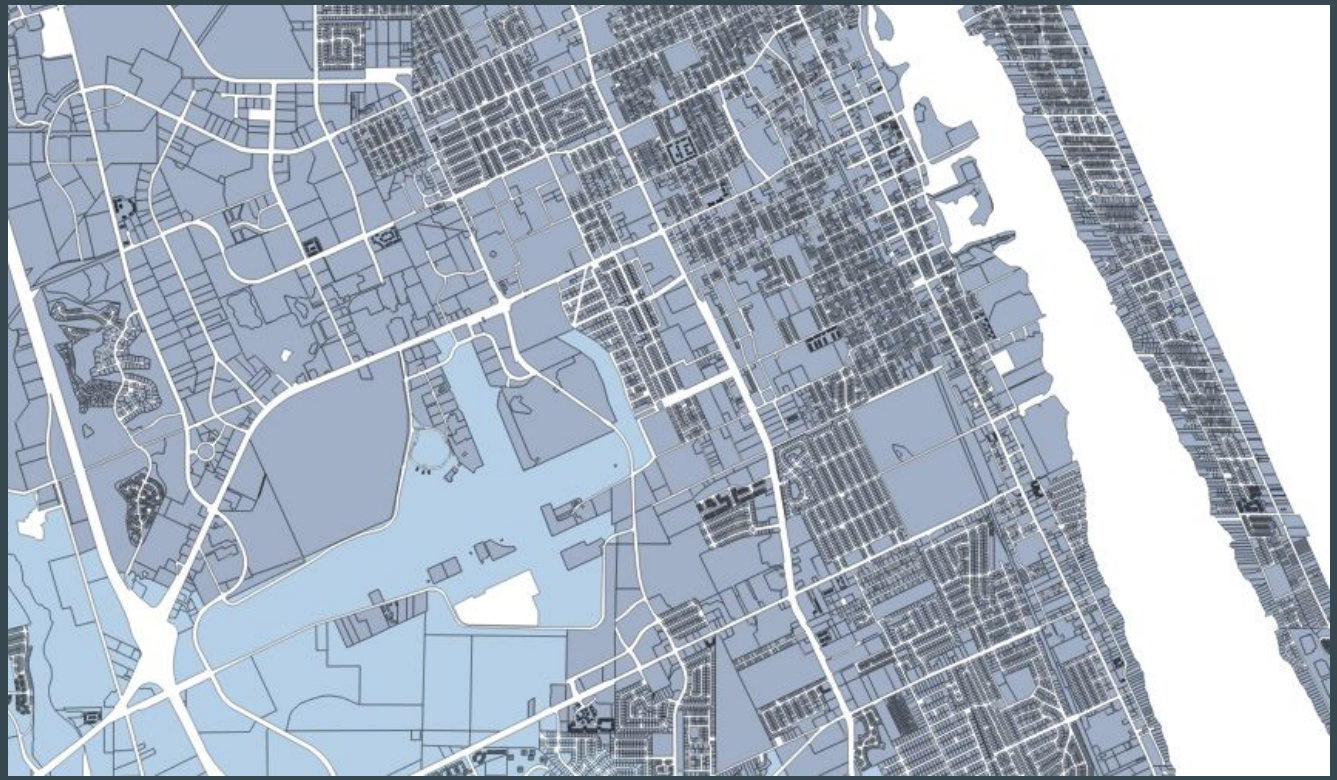
```
    if i%10000 == 0:
        print(i)
        conn.commit()
    row = cur.fetchone()
    print("done")

#df = pd.
conn.commit()
```

Step 4: QGIS Visualization

- Since we added our geom column to the parcel table, we can now see the fruits of our labour.
- First reopen QGIS and add the layer
  - Click Layer > Add Layer > Add PostGIS Layer
- Connect to the server and add the table as a layer
  - Click the dropdown menu > Select Spatial > Click Connect > Click the volusia dropdown arrow > select parcel > Click Add > Click Close
- Right click the layer and go to Properties
  - Click Symbology > Click the dropdown menu >Select Graduated > Value: fzdistance > Color Ramp: (your choice) > Click Classify> Click Apply > Click Ok

Step 4: QGIS Visualization



Conclusion

- Thank you for reading
- For more information visit the github below:

https://github.com/Dumitrek/CS540_flood_zone_project

Sources

[1] PostGis. (n.d.). *Introduction to PostGis*. 27. Nearest-Neighbour Searching - Introduction to PostGIS. https://postgis.net/workshops/postgis-intro/knn.html.