Volusia County Project
Finding Parcel Elevation

Tyler Procko
under Professor Steven Lehr, CS540
ERAU Daytona, Spring 2021

https://github.com/Psychobagger/CS540_Project

Assumptions

- Higher elevation should correlate with higher property values
  - Except for beachfront properties, as these are at/below sea-level and generally cost quite a lot
- No scripting or looping should be necessary outside of SQL
  - It should be pretty straightforward
- The parcel elevation value belongs in the parcel table

Step 1: Getting the Data

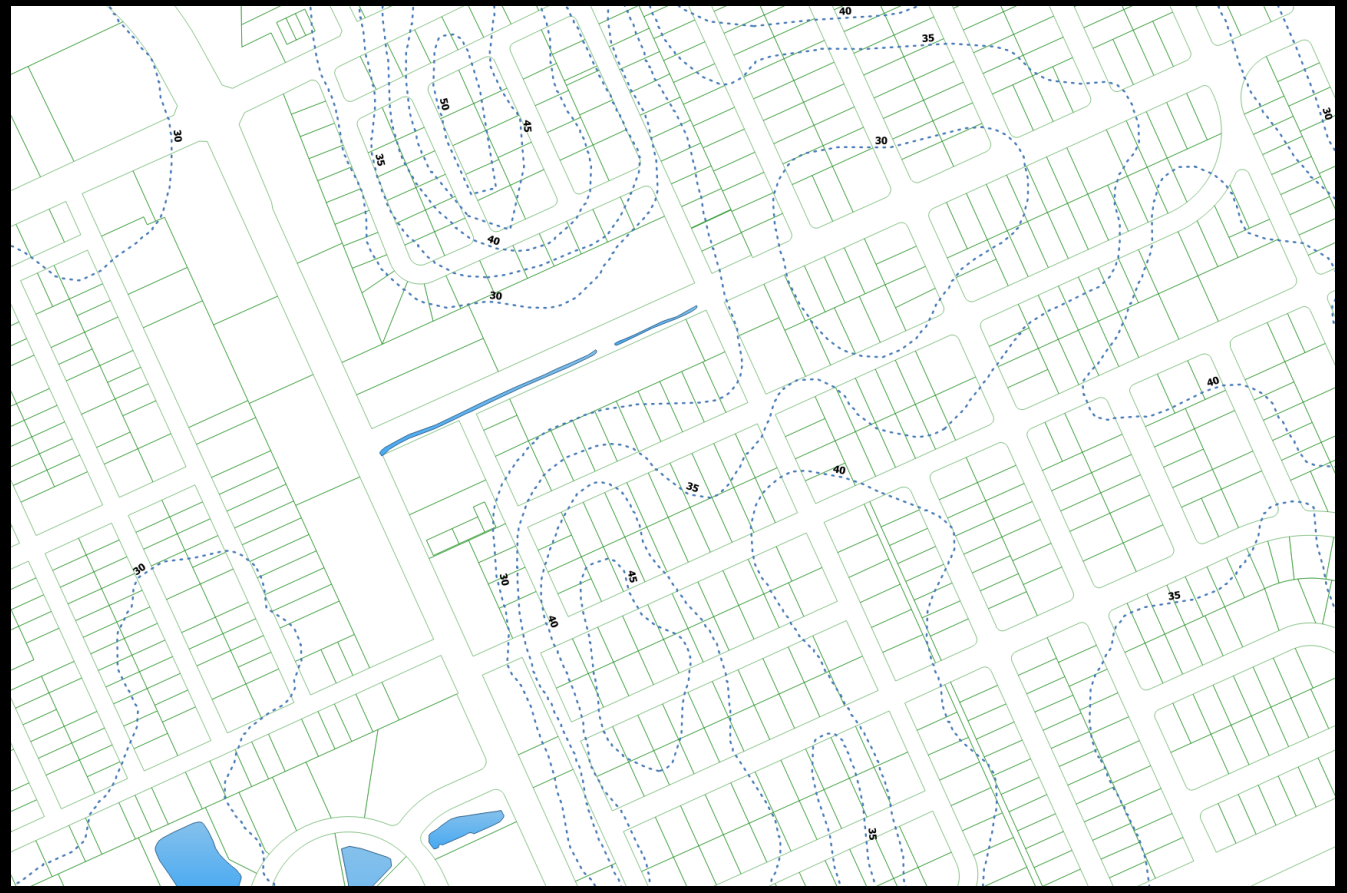
- Elevation values come attached to contour data, which are lines on a map meant to represent the changes in physical terrain
  - It's just 2D lines with numbers associated with them
- Downloaded http://maps.vcgov.org/gis/download/shpfiles/contours.zip
- Put the .shp file into QGIS (see next slide)

Step 1: Getting the Data Cont.

- Add the .shp file to QGIS like we are taught (Layer > Add Layer > New Vector Layer)
- View it, check it out, verify that there are now lines on the map
- Click the Processing tab up top
- Click Toolbox
- A menu will pop up on the right of QGIS
- Click Database
- Click Export to PostgreSQL
- Select the contours layer (should already be selected)
- Type in your login info and the schema you're adding to (volusia)
- Leave the table name blank, it will default to "contours"
- Don't touch anything else
- Scroll down and click Run
- Go to PGAdmin and refresh the volusia schema; the contours table should be there

Step 1: Getting the Data Cont.

- This is what contours look like
- Notice the numbers by each one
  - These are relatively accurate elevation levels



Step 2: Queries

- How do we find the elevation of a parcel?
- The general idea is to find the contour line closest to a parcel, and assume that this is its relative elevation
- Initial work was very complex, involving intercepts, interpolation, projecting along the cardinal directions… it would have worked but just doing this in one direction took several dozen lines of SQL and quintuple-nested GIS function calls
- So, it was simplified…

Step 2: Queries Cont.

- This is the original query just to check to the right (+x) of a parcel for intercepted contour lines
- Casts a line 10000 units to the right of the parcel centroid, sees if it intersects any contour lines
- COMPLICATED!
- And needed to do this *in every direction*... too much code

```
select ST_Intersects(
        ST_MakeLine(
          ST_Centroid(
              ST_Transform(
                  ST_SetSRID(s.geom, 2236), 2236)
              ),
          ST_SetSRID(
              ST_MakePoint(
                  ST_X(
                      ST_Centroid(
                          ST_Transform(
                              ST_SetSRID(s.geom, 2236), 2236)
                          )
                      )+10000,
                  ST_Y(
                      ST_Centroid(
                          ST_Transform(
                              ST_SetSRID(s.geom, 2236), 2236)
                          )
                      )
                  ),
            2236)),
          ST_Transform(
              ST_SetSRID(c.geom, 2236), 2236)
        ) as intersects,
c.objectid, c.elev
from volusia.sales_analysis s, volusia.contours c
where c.geom is not null and parid=4811435 and c.objectid > 11000;
```

Step 2: Queries Cont.

- The new version of the query
- Uses ST_Distance to find the distance from each parcel centroid to *every* contour line
- Puts it into a new table
- Takes 1 hour to run for just two zip codes!

```
select gp.altkey as parid,
ST_Distance(
  ST_Centroid(
    ST_Transform(
      gp.geom, 2236
    )
  ),
  ST_Transform(
    ST_SetSRID(c.geom, 2236), 2236
  )
) as parcel_elevation_contour_distance,
c.elev
into volusia.contours_analysis
from volusia.gis_parcels gp, volusia.contours c, volusia.situs s
where s.parid = gp.altkey and (s.zip1 ilike '32114' or s.zip1 ilike '32118');
```

Step 2: Queries Cont.

- The second necessary query
- Finds the contour line with the *smallest* distance to the parcel, and assumes that this is the parcel elevation (elegant?)
- Reduces all of the work to a two-column table ->
- Takes about 15 minutes to run

```
select ca.parid, ca.elev
into volusia.contours_analysis2
from volusia.contours_analysis ca inner join
(
  select parid, min(parcel_elevation_contour_distance) as min_distance
  from volusia.contours_analysis
  group by parid
) t
on ca.parid = t.parid and ca.parcel_elevation_contour_distance = t.min_distance;
```

	parid double precision	elev integer
1	3410271	20
2	3409728	20
3	3410017	20
4	3409744	15
5	3410289	15
6	3410297	20
7	3410301	10
8	3409752	15
9	3409752	15
10	3410025	20
11	3410319	15
12	3409761	15
13	3410327	15
14	3409779	15
15	3410335	10

Step 2: Queries Cont.

- The third necessary query
- Appends an *elevation* column to the parcel table
- Very easy, takes milliseconds

```
alter table volusia.parcel add column parcel_elevation integer;
```

Step 2: Queries Cont.

- The last necessary query
- Dumps all of the elevation values into the parcel table (SQL UPDATE)

- Done!

```
update volusia.parcel_elev p
set parcel_elevation = c.elev
from volusia.contours_analysis2 c
where p.parid = c.parid;
```

Step 2: Queries Cont.

- Note the last column… we have elevation! Now… to visualize it.

```
select * from volusia.parcel_elev where parcel_elevation is not null;
```

Output | Explain | Messages | Notifications

deletions double precision	sasd double precision	nsasd double precision	stxbl double precision	nstxbl double precision	cotxbl double precision	citxbl double precision	cra text	cur text	dtcreated text	naics text	naics_desc text	eqval double precision	penval double precision	livunit text	hx_flag text	parcel_elevation integer
0	158218	158218	158218	158218	158218	158218	[null]	Y	14-DEC-82	[null]	[null]	0	0	[null]	N	10
0	152817	152817	152817	152817	152817	152817	[null]	Y	14-DEC-82	[null]	[null]	0	0	[null]	N	10
0	197877	197877	172877	147877	147877	147877	[null]	Y	01-JUL-83	[null]	[null]	0	0	[null]	Y	10
0	55224	52476	55224	52476	52476	52476	[null]	Y	15-JUN-60	[null]	[null]	0	0	[null]	N	20
0	237264	237264	237264	237264	237264	237264	[null]	Y	26-MAR-74	[null]	[null]	0	0	[null]	N	20
0	300758	300758	275758	250758	250758	250758	[null]	Y	26-DEC-72	[null]	[null]	0	0	[null]	Y	20
0	63758	46780	63758	46780	46780	46780	88	Y	24-APR-52	[null]	[null]	0	0	[null]	N	20
0	80546	80546	80546	80546	80546	80546	[null]	Y	15-JUN-63	[null]	[null]	0	0	[null]	N	10
0	163451	163451	138451	113451	113451	113451	[null]	Y	15-JUN-77	[null]	[null]	0	0	[null]	Y	25
0	58910	53079	58910	53079	53079	53079	[null]	Y	15-DEC-70	[null]	[null]	0	0	[null]	N	15
0	231648	231648	206648	181648	181648	181648	[null]	Y	30-OCT-80	[null]	[null]	0	0	[null]	Y	25
0	198285	198285	173285	148285	148285	148285	[null]	Y	15-FEB-74	[null]	[null]	0	0	[null]	Y	25
0	150214	150214	125214	100214	100214	100214	[null]	Y	16-FEB-07	[null]	[null]	0	0	[null]	Y	15
0	67132	67132	67132	67132	67132	67132	[null]	Y	28-DEC-75	[null]	[null]	0	0	[null]	N	5
0	35700	35700	10700	10700	10700	10700	96	Y	30-JAN-84	[null]	[null]	0	0	[null]	Y	10
0	50678	50678	50678	50678	50678	50678	95	Y	11-SEP-06	[null]	[null]	0	0	[null]	N	15
0	59600	59600	59600	59600	59600	59600	[null]	Y	20-APR-67	[null]	[null]	0	0	[null]	N	15
0	434277	434277	408277	383277	383277	383277	[null]	Y	10-JAN-06	[null]	[null]	0	0	[null]	Y	20

Step 3: Visualizing in QGIS

- You have to have a geom column in the parcel table to be able to represent it in QGIS
- The last two queries to execute:

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

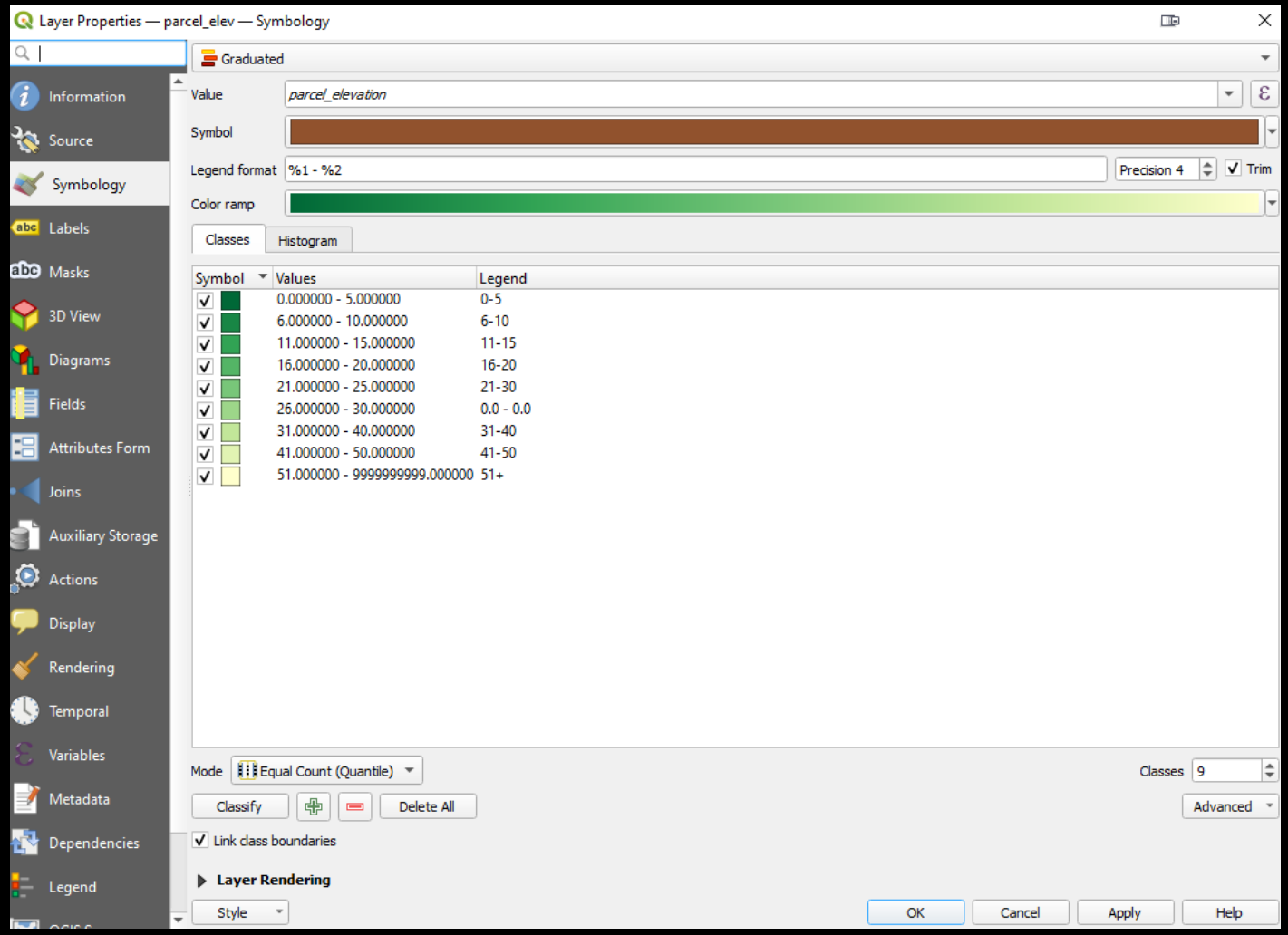
- *parcel_elev* is my copy of *parcel*, they are identical

Step 3: Visualizing in QGIS Cont.

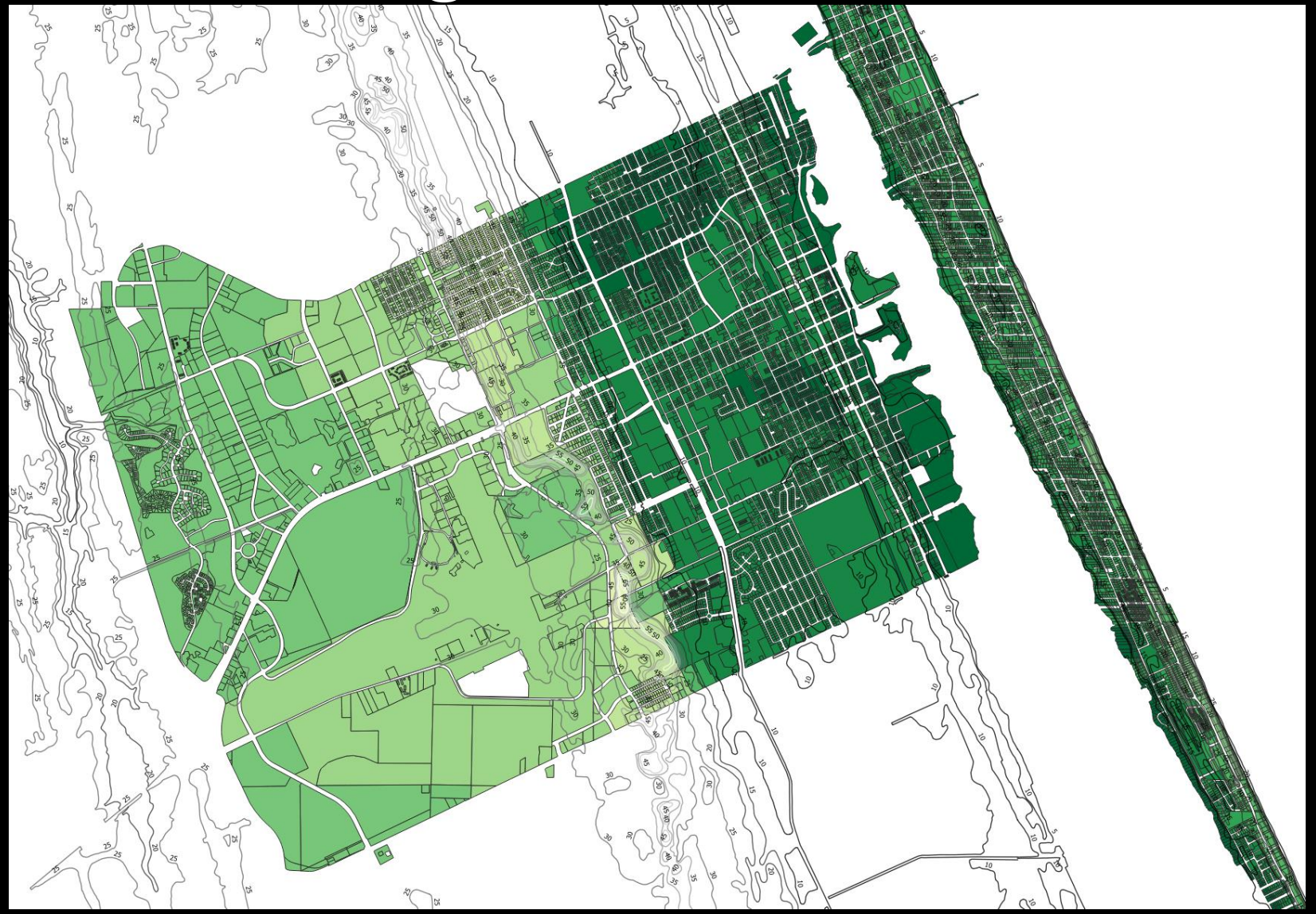
- The GEOM column is there, so now we can see stuff!
- Open QGIS and make sure you're connected to the server
- Add a PostGIS layer and select parcel_elev (or parcel)
- Right click the layer, go to properties. On Symbology do Graduated, for the symbol 'parcel_elevation’.
- Enter values for the elevation column like 0-5, 6-10, 10-20 etc., and give them each a color. You can specify a color graduation so it blends.
- Apply and enable the layer visibility

Step 3: Visualizing in QGIS Cont.

- My Symbology window (use the color ramp menu for the graduated colors)
- I made it so lower elevations are darker, higher elevations are lighter



Step 3: Visualizing in QGIS Cont.



Concluding Remarks

- Thanks for reading
- Find everything you need below

https://github.com/Psychobagger/CS540_Project