Reassessing Ward Boundaries in the City of Chicago

Recommendations for Boundaries without Gerrymandering

Wesley Street, Patrick McPartlan, Mike Adduci

GEOG 574 - Fall 2020

Abstract

Since 1837, the City of Chicago has been divided into 6 "Wards" which define political boundaries represented by individual councilmembers or Aldermen. Starting in 1923, 50 such wards were defined as representative bodies with associated numbers (1-50). However, since its inception, councilmembers, under the direction of the Mayor, have had the freedom to re-draw boundaries in a controversial practice known as gerrymandering.

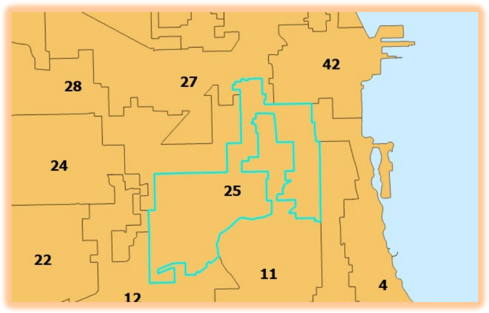
Problem Statement & Background

Several examples are broadly evident within the City of Chicago, making it a prime example of the ethical dilemma surround gerrymandering and whether the City's unusual boundaries actually are equitable and fair. The horseshoe-shaped 25th ward on Chicago's near-west side (Figure 1-1) is a prime example of obtuse boundaries which warrant a new data-driven model and process for drawing ethical and appropriate boundaries that do not serve a political agenda.

Introduction

To accomplish this, various available software and geospatial tools were employed to build balanced hypothetical boundaries which utilize a population-based formula to determine equitable boundaries with no data bias aside from population clusters and sustain fair representation of populations.

Figure 1-1



Data

A model was designed to conceptualize the initial data inputs and arrive at a probable solution to creating an output that does not *change* existing census or city data, but rather arrives at meaningful results that can be easily quantified by stakeholders or government leaders alike.

Data from the MMMG (Metric Geometry & Gerrymandering Group) was utilized to reference ward, precinct, and community boundaries in the database implementation stage. These feature classes provided a multi-tiered approach to our research which examines all aspects of neighborhoods and their existing boundaries. In particular, precinct data provides population characteristics which are invaluable in determining population-dense clusters for equitable hypothetical boundaries.

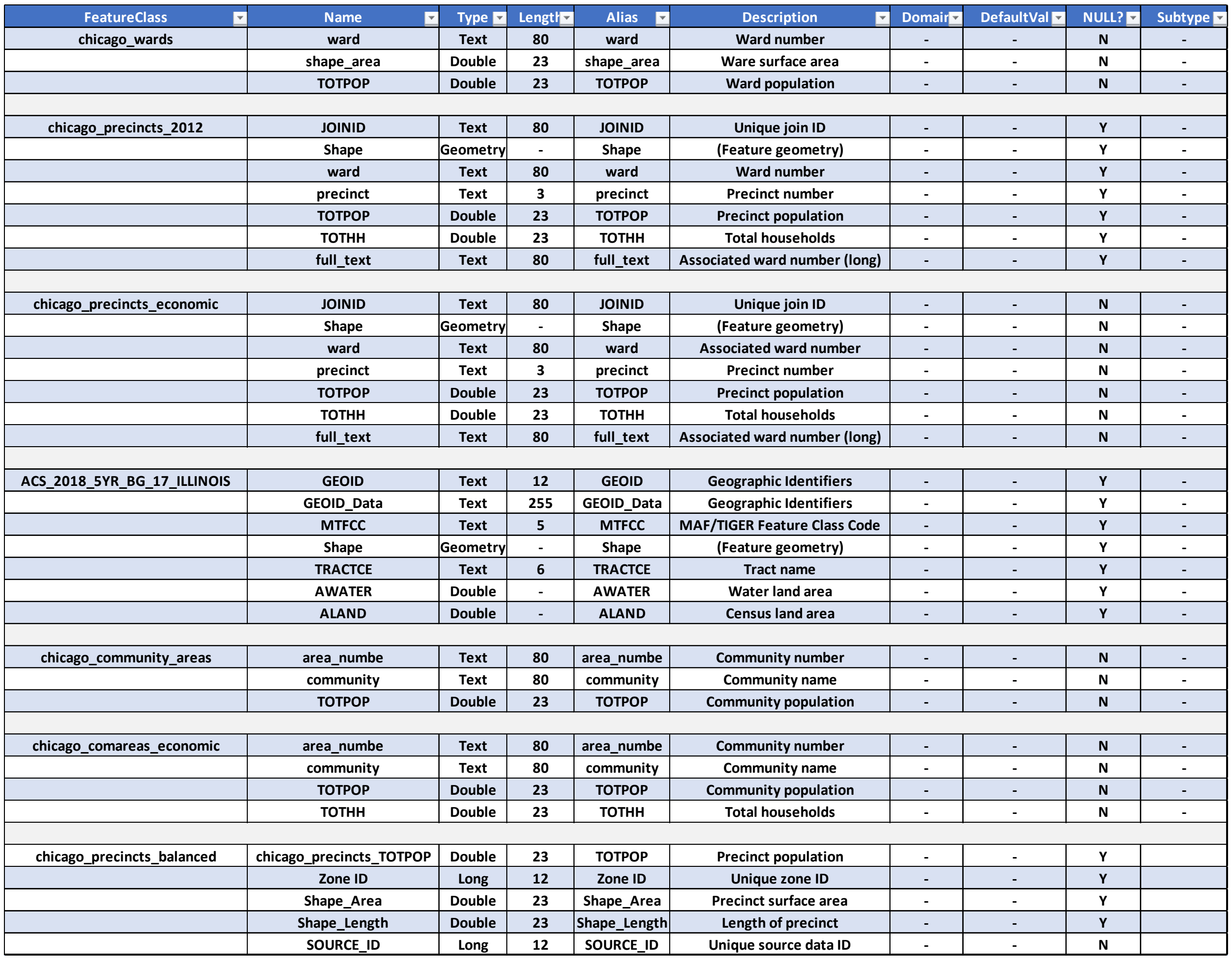
In addition, select field from the U.S. Census Bureau's 5-year American Community Survey (ACS) geodatabase were queried to extract and recompile as a master database of various census attributes.

Geometries from both resources were then merged together to form a new hypothetical set of fair boundaries which the City of Chicago can examine as a data-driven approach to ethical boundaries. These methods, queries, and results used to arrive at a result are detailed within the Data Implantation and Database Manipulation sections of this report.

Model Design

The initial conceptual design of this database is arranged by names, types, character lengths, alias name, description, and information on domains, NULL value handling, and subtypes, if any. The following table illustrates all of those values for this project, organized for each feature class input in the database.

Figure 1-2



FeatureClass	Name	Type	Lengtl	Alias	Description	Domair	DefaultVal	NULL?	Subtype
chicago_wards	ward	Text	80	ward	Ward number	-	-	N	-
	shape_area	Double	23	shape_area	Ware surface area	-	-	N	-
	TOTPOP	Double	23	TOTPOP	Ward population	-	-	N	-
chicago_precincts_2012	JOINID	Text	80	JOINID	Unique join ID	-	-	Y	-
	Shape	Geometry	-	Shape	(Feature geometry)	-	-	Y	-
	ward	Text	80	ward	Ward number	-	-	Y	-
	precinct	Text	3	precinct	Precinct number	-	-	Y	-
	TOTPOP	Double	23	TOTPOP	Precinct population	-	-	Y	-
	TOTHH	Double	23	TOTHH	Total households	-	-	Y	-
	full_text	Text	80	full_text	Associated ward number (long)	-	-	Y	-
chicago_precincts_economic	JOINID	Text	80	JOINID	Unique join ID	-	-	N	-
	Shape	Geometry	-	Shape	(Feature geometry)	-	-	N	-
	ward	Text	80	ward	Associated ward number	-	-	N	-
	precinct	Text	3	precinct	Precinct number	-	-	N	-
	TOTPOP	Double	23	TOTPOP	Precinct population	-	-	N	-
	TOTHH	Double	23	TOTHH	Total households	-	-	N	-
	full_text	Text	80	full_text	Associated ward number (long)	-	-	N	-
ACS_2018_5YR_BG_17_ILLINOIS	GEOID	Text	12	GEOID	Geographic Identifiers	-	-	Y	-
	GEOID_Data	Text	255	GEOID_Data	Geographic Identifiers	-	-	Y	-
	MTFCC	Text	5	MTFCC	MAF/TIGER Feature Class Code	-	-	Y	-
	Shape	Geometry	-	Shape	(Feature geometry)	-	-	Y	-
	TRACTCE	Text	6	TRACTCE	Tract name	-	-	Y	-
	AWATER	Double	-	AWATER	Water land area	-	-	Y	-
	ALAND	Double	-	ALAND	Census land area	-	-	Y	-
chicago_community_areas	area_numbe	Text	80	area_numbe	Community number	-	-	N	-
	community	Text	80	community	Community name	-	-	N	-
	TOTPOP	Double	23	TOTPOP	Community population	-	-	N	-
chicago_comareas_economic	area_numbe	Text	80	area_numbe	Community number	-	-	N	-
	community	Text	80	community	Community name	-	-	N	-
	TOTPOP	Double	23	TOTPOP	Community population	-	-	N	-
	TOTHH	Double	23	TOTHH	Total households	-	-	N	-
chicago_precincts_balanced	chicago_precincts_TOTPOP	Double	23	TOTPOP	Precinct population	-	-	Y	
	Zone ID	Long	12	Zone ID	Unique zone ID	-	-	Y	
	Shape_Area	Double	23	Shape_Area	Precinct surface area	-	-	Y	
	Shape_Length	Double	23	Shape_Length	Length of precinct	-	-	Y	
	SOURCE_ID	Long	12	SOURCE_ID	Unique source data ID	-	-	N	

Figure 1-3 on the following page illustrates the conceptual design of the database and the resulting relations formed with the merger of different geometries and attributes throughout the data process. The process can be read top to bottom beginning with wards, their associated precincts, ACS census data joins, and the resulting export to "chicago_precincts_balanced". Associated data which was referenced, but not joined directly is also included in the concept diagram.

The associated logical schema in Figure 1-4 illustrates the primary keys as underlined attributes and related foreign keys (fk) as underlined attributes. Unique values are notated with a "(u)".

Figure 1-3

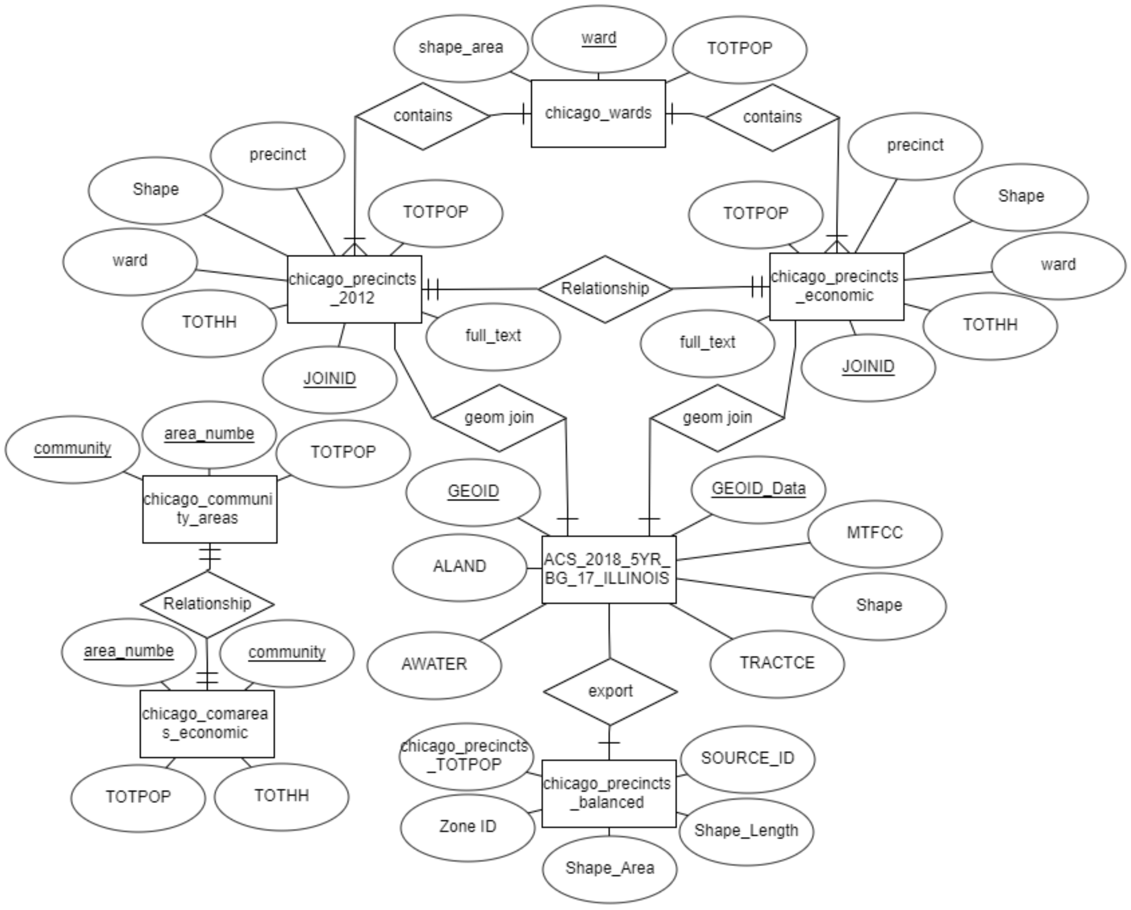
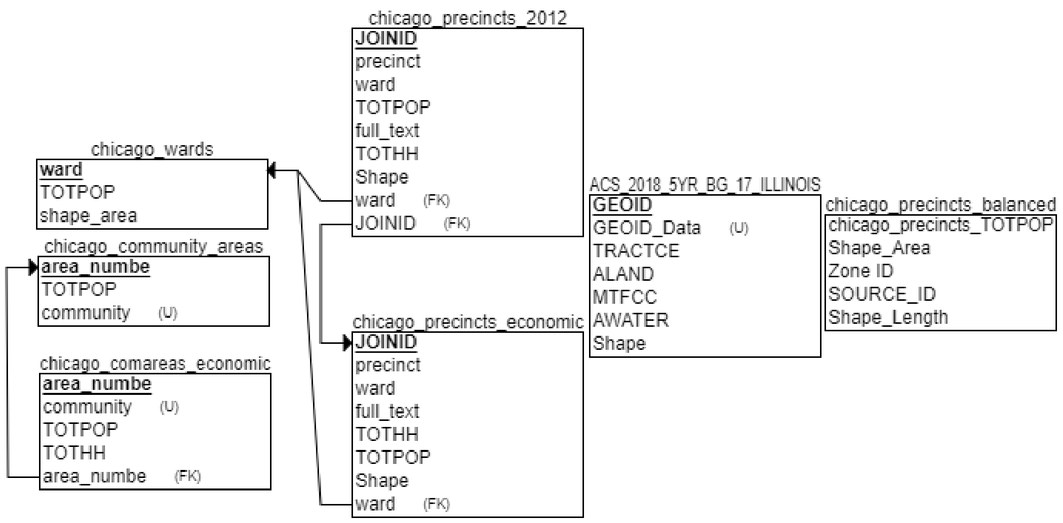


Figure 1-4

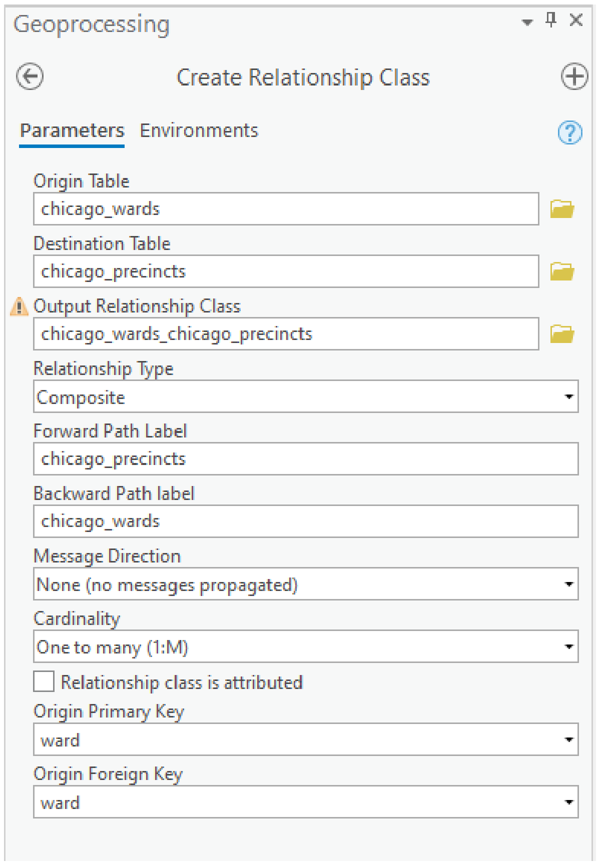


Database Implementation

The initial plan for this project was to use PG Admin to create a PostGIS database; however, the data that we were using for this project is in a file geodatabase already. A file geodatabase will allow a user to load data, it doesn't necessarily support all the data types and relationships in a PostGIS database. A file geodatabase and Arc in PostGIS doesn't support unique values outside of the OID field where SQL and Oracle do. With this in mind, it was decided to leverage a file geodatabase. This will also ease deployment to ArcGIS Online.

The first step was setting up the feature classes. Wards was built using the Create Feature Class geoprocessing tool (GP) and subsequently loaded with features; however, it was quickly realized that the tables and spatial data already existed and it was easier to manipulate these instead recreate them from scratch and then load the data. The existing feature classes of chicago_precincts and chicago_precincts_economic were manipulated to tables only using the Table to Table GP tool and then Delete Field GP tool to remove attributes that were not applicable to this project.

At this point there was a chicago_wards feature class and two supporting tables. The next step was to build relationship tables between the geometry and tables. This would enable declaration of primary keys and foreign keys. The first hurdle was the wards were captured in three different data types in the three tables, one text, one long, and one double and a relationship cannot be built between text and a number type. The solution to this was to add a field, calculate the value using python, remove the existing field, add a new field with the correct name, and use python to calculate it back. The same thing was done for precincts. Once this was complete, building a relationship in the file geodatabase was straight forward, using the ward as the primary key to the foreign key in the table and a one to many between wards and precincts (figure 2-1). The same process was used between JOINID between tables using a one-to-one relationship.


Geoprocessing

Create Relationship Class

Parameters Environments

Origin Table: chicago_wards

Destination Table: chicago_precincts

Output Relationship Class: chicago_wards_chicago_precincts

Relationship Type: Composite

Forward Path Label: chicago_precincts

Backward Path label: chicago_wards

Message Direction: None (no messages propagated)

Cardinality: One to many (1:M)

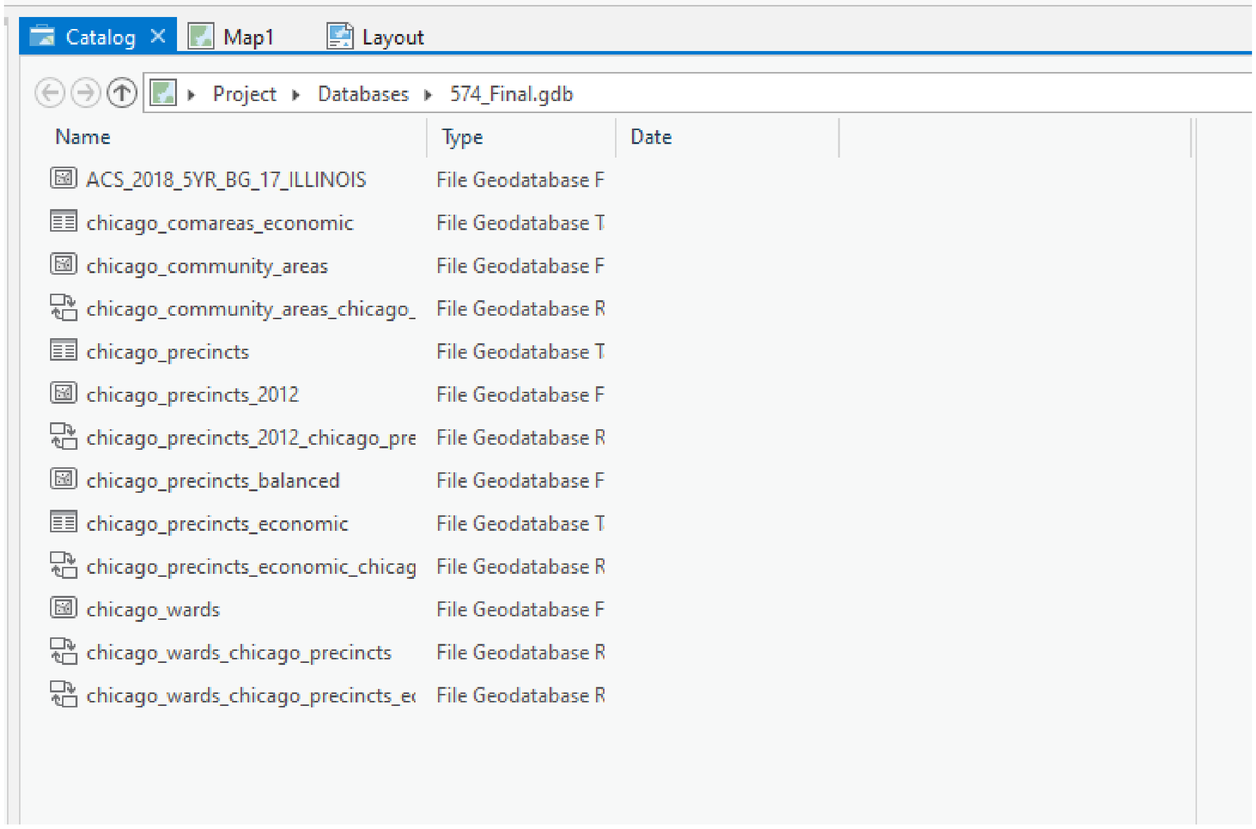
[ ] Relationship class is attributed

Origin Primary Key: ward

Origin Foreign Key: ward

Figure 2-1

Once wards were complete. A similar process was used between the geometry of precincts and the two precincts tables and the community areas and their economic data. No field cleaning was required since the area_numbe field used the same text data type. ACS data for the sate of Illinois was also imported. This feature class also had many fields exceeding the requirements of this project and were removed. The completed geodatabase had three feature classes, three tables, and four relationships (figure 2-2).



Name	Type	Date
ACS_2018_5YR_BG_17_ILLINOIS	File Geodatabase F	
chicago_comareas_economic	File Geodatabase T	
chicago_community_areas	File Geodatabase F	
chicago_community_areas_chicago_	File Geodatabase R	
chicago_precincts	File Geodatabase T	
chicago_precincts_2012	File Geodatabase F	
chicago_precincts_2012_chicago_pre	File Geodatabase R	
chicago_precincts_balanced	File Geodatabase F	
chicago_precincts_economic	File Geodatabase T	
chicago_precincts_economic_chicag	File Geodatabase R	
chicago_wards	File Geodatabase F	
chicago_wards_chicago_precincts	File Geodatabase R	
chicago_wards_chicago_precincts_ec	File Geodatabase R	

Figure 2-2

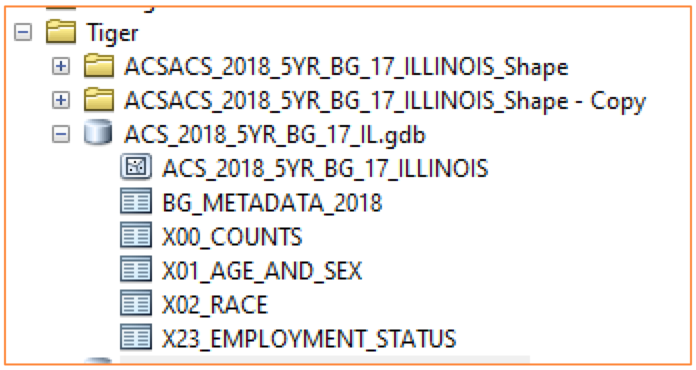
Database Manipulation:

A new database was created using postgresql and the postgis_25_sample template for the database, which acts as the installation of postgis engine extension (https://www.programmersought.com/article/35871027073/):

```
CREATE DATABASE final
    WITH
    OWNER = postgres
    TEMPLATE = postgis_25_sample
    ENCODING = 'UTF8'
    CONNECTION LIMIT = -1
```

The geodatabase labeled ACS_2018_5YR_BG_17_ILLINOIS.gdb was downloaded from online and put into a working directory. In arcmap, the number of tables related to the geodatabase was reduced to include the attribute tables that'll be used in future analysis to just X01_Age_AND_SEX, X02_Race and X23_Employment_Status. The result is shown below in Figure 3-1:

Figure 3-1



```
Tiger
  ACSACS_2018_5YR_BG_17_ILLINOIS_Shape
  ACSACS_2018_5YR_BG_17_ILLINOIS_Shape - Copy
  ACS_2018_5YR_BG_17_IL.gdb
    ACS_2018_5YR_BG_17_ILLINOIS
    BG_METADATA_2018
    X00_COUNTS
    X01_AGE_AND_SEX
    X02_RACE
    X23_EMPLOYMENT_STATUS
```

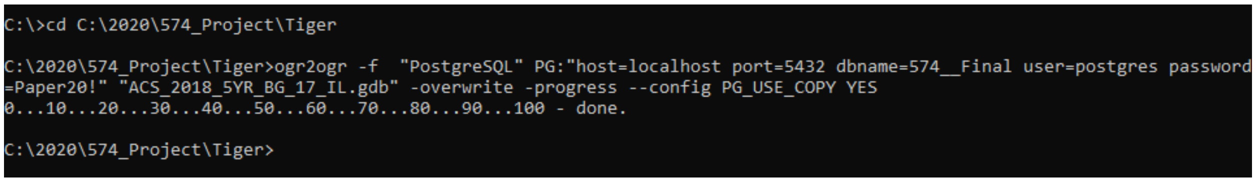
The number of columns affiliated with the X23_EMPLOYMENT_STATUS is over 1600+ columns, so the ArcPy function "DELETE FIELDS" was used to cut down the count to 22 fields (keeping relevant attributes and the GEOID field for future joins on the shapefile.

The extent of the expression used in ArcPY:

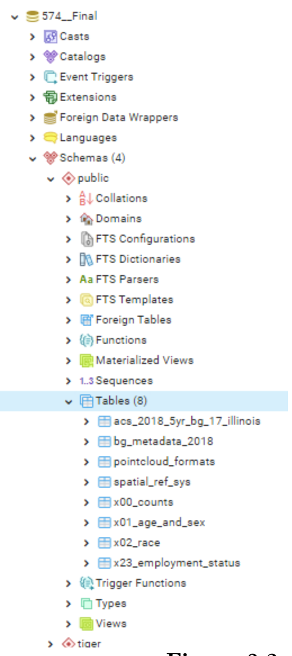
```
arcpy.DeleteField_management('X23_EMPLOYMENT_STATUS',
'B23001e1;B23001m1;B23001e2;B23001m2;B23001e3;B23001m3;B23001e4;
//..//B23027m35;B23027e36;B23027m36')
```

The majority of these fields produced NULL values. The research extensively looked into the capabilities of Postgresql to delete all fields with exclusive null values, but a valid methodology was not discovered, so the implementation of ArcPy functionality was useful. After the manipulation, the gdb was exported into the new postgresql geodatabase, using the following command in the OSGEO4W shell (after setting the relative pathways):

```
ogr2ogr -f "PostgreSQL" PG:"host=localhost port=5432 dbname=574__Final user=postgres
password=Paper20!" "ACS_2018_5YR_BG_17_IL.gdb" -overwrite -progress --config
PG_USE_COPY YES
```



```
C:\>cd C:\2020\574_Project\Tiger

C:\2020\574_Project\Tiger>ogr2ogr -f  "PostgreSQL" PG:"host=localhost port=5432 dbname=574__Final user=postgres password
=Paper20!" "ACS_2018_5YR_BG_17_IL.gdb" -overwrite -progress --config PG_USE_COPY YES
0...10...20...30...40...50...60...70...80...90...100 - done.

C:\2020\574_Project\Tiger>
```



```
574__Final
  Casts
  Catalogs
  Event Triggers
  Extensions
  Foreign Data Wrappers
  Languages
  Schemas (4)
    public
      Collations
      Domains
      FTS Configurations
      FTS Dictionaries
      FTS Parsers
      FTS Templates
      Foreign Tables
      Functions
      Materialized Views
      Sequences
      Tables (8)
        acs_2018_5yr_bg_17_illinois
        bg_metadata_2018
        pointcloud_formats
        spatial_ref_sys
        x00_counts
        x01_age_and_sex
        x02_race
        x23_employment_status
      Trigger Functions
      Types
      Views
    tiger
```

Figure 3-2

The graphic to the left (Figure 3-2) is a snap shot of the pgAdmin interface showing the results of the ogr2ogr command. When the file geodatabase was imported into postgis "final" geodatabase, a folder containing the shapefile file along with the reduced tables was created.

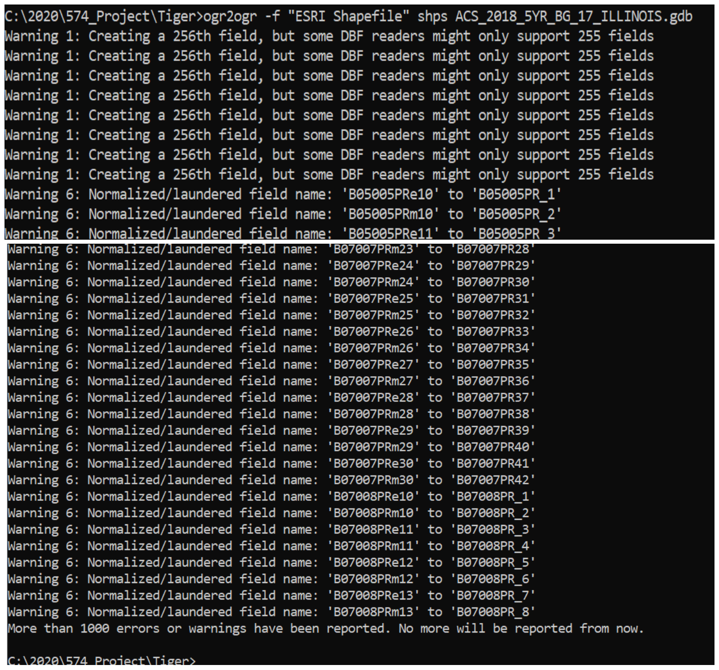
These files can now be manipulated to further analyze the data.

The shapefile was transferred over with the last column labeled as "shape" as the geometry field. This carries over spatial charecteristics from the shapefile, giving the table its place in space, with the projection pre-assigned with the file, GCS_North_American_1983 WKID: 4269 Authority: EPSG.

Also, the shapefile carried over the category GEOID from before. This field can be used in the next step when the attributes are being combined into the table affiliated with the shapefile.

As a side note, another option for inserting a gdb into PostGIS is to first convert it to a shapefile. The associated attributes tables will then turn into dbf files, using the following function;

```
ogr2ogr -f "ESRI Shapefile" shps ACS_2018_5YR_BG_17_ILLINOIS.gdb
```

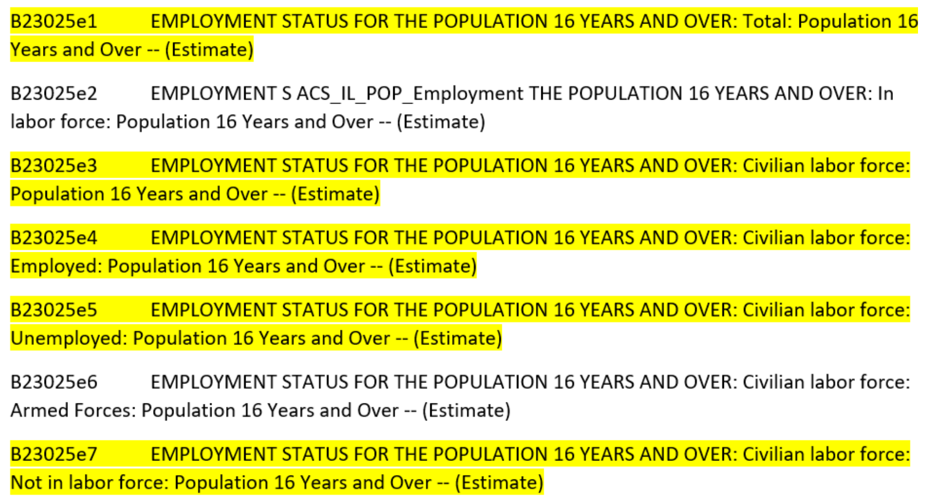


```
C:\2020\574_Project\Tiger>ogr2ogr -f "ESRI Shapefile" shps ACS_2018_5YR_BG_17_ILLINOIS.gdb
Warning 1: Creating a 256th field, but some DBF readers might only support 255 fields
Warning 1: Creating a 256th field, but some DBF readers might only support 255 fields
Warning 1: Creating a 256th field, but some DBF readers might only support 255 fields
Warning 1: Creating a 256th field, but some DBF readers might only support 255 fields
Warning 1: Creating a 256th field, but some DBF readers might only support 255 fields
Warning 1: Creating a 256th field, but some DBF readers might only support 255 fields
Warning 1: Creating a 256th field, but some DBF readers might only support 255 fields
Warning 1: Creating a 256th field, but some DBF readers might only support 255 fields
Warning 6: Normalized/laundered field name: 'B05005PRe10' to 'B05005PR_1'
Warning 6: Normalized/laundered field name: 'B05005PRm10' to 'B05005PR_2'
Warning 6: Normalized/laundered field name: 'B05005PRe11' to 'B05005PR_3'
Warning 6: Normalized/laundered field name: 'B07007PRm23' to 'B07007PR28'
Warning 6: Normalized/laundered field name: 'B07007PRe24' to 'B07007PR29'
Warning 6: Normalized/laundered field name: 'B07007PRm24' to 'B07007PR30'
Warning 6: Normalized/laundered field name: 'B07007PRe25' to 'B07007PR31'
Warning 6: Normalized/laundered field name: 'B07007PRm25' to 'B07007PR32'
Warning 6: Normalized/laundered field name: 'B07007PRe26' to 'B07007PR33'
Warning 6: Normalized/laundered field name: 'B07007PRm26' to 'B07007PR34'
Warning 6: Normalized/laundered field name: 'B07007PRe27' to 'B07007PR35'
Warning 6: Normalized/laundered field name: 'B07007PRm27' to 'B07007PR36'
Warning 6: Normalized/laundered field name: 'B07007PRe28' to 'B07007PR37'
Warning 6: Normalized/laundered field name: 'B07007PRm28' to 'B07007PR38'
Warning 6: Normalized/laundered field name: 'B07007PRe29' to 'B07007PR39'
Warning 6: Normalized/laundered field name: 'B07007PRm29' to 'B07007PR40'
Warning 6: Normalized/laundered field name: 'B07007PRe30' to 'B07007PR41'
Warning 6: Normalized/laundered field name: 'B07007PRm30' to 'B07007PR42'
Warning 6: Normalized/laundered field name: 'B07008PRe10' to 'B07008PR_1'
Warning 6: Normalized/laundered field name: 'B07008PRm10' to 'B07008PR_2'
Warning 6: Normalized/laundered field name: 'B07008PRe11' to 'B07008PR_3'
Warning 6: Normalized/laundered field name: 'B07008PRm11' to 'B07008PR_4'
Warning 6: Normalized/laundered field name: 'B07008PRe12' to 'B07008PR_5'
Warning 6: Normalized/laundered field name: 'B07008PRm12' to 'B07008PR_6'
Warning 6: Normalized/laundered field name: 'B07008PRe13' to 'B07008PR_7'
Warning 6: Normalized/laundered field name: 'B07008PRm13' to 'B07008PR_8'
More than 1000 errors or warnings have been reported. No more will be reported from now.

C:\2020\574_Project\Tiger>
```

Figure 3-3

However, as you can see in the graphic to the left (Figure 3-3), there were significant issues when converting this particular geodatabase to a shapefile due to the number of fields in the attribute table. Even after some manipulation of the tables, within ArcMap, there were still errors with the other tables and the total amount of fields in each of the tables. This is more practical of an application when the datasets, and the fields, are more condensed.

Back to the procedure, now it is time to join the datasets. This allows the associated tables to be combined with the shapefile containing the geometry field. This process is essentially acting as a join within ArcMap, but using SQL and PostGIS procedures.



B23025e1 EMPLOYMENT STATUS FOR THE POPULATION 16 YEARS AND OVER: Total: Population 16 Years and Over -- (Estimate)

B23025e2 EMPLOYMENT S ACS_IL_POP_Employment THE POPULATION 16 YEARS AND OVER: In labor force: Population 16 Years and Over -- (Estimate)

B23025e3 EMPLOYMENT STATUS FOR THE POPULATION 16 YEARS AND OVER: Civilian labor force: Population 16 Years and Over -- (Estimate)

B23025e4 EMPLOYMENT STATUS FOR THE POPULATION 16 YEARS AND OVER: Civilian labor force: Employed: Population 16 Years and Over -- (Estimate)

B23025e5 EMPLOYMENT STATUS FOR THE POPULATION 16 YEARS AND OVER: Civilian labor force: Unemployed: Population 16 Years and Over -- (Estimate)

B23025e6 EMPLOYMENT STATUS FOR THE POPULATION 16 YEARS AND OVER: Civilian labor force: Armed Forces: Population 16 Years and Over -- (Estimate)

B23025e7 EMPLOYMENT STATUS FOR THE POPULATION 16 YEARS AND OVER: Civilian labor force: Not in labor force: Population 16 Years and Over -- (Estimate)

From the economic dataset we want to add the categories that are highlighted in the following graphic to show how the employment/unemployment/not in labor force compare against one another.

The next script will take the shapefile geometries brought in during the gdb to postgis process and combine it with the attributes of the desired table, in this case the economic table. This takes advantage of the matching attribute the geoid.

CREATE TABLE ACS_IL_POP_Employment AS

SELECT e.objectid, e.geoid,

e.b23025e1 AS Total_Pop_Over16, e.b23025e4 AS Employed,

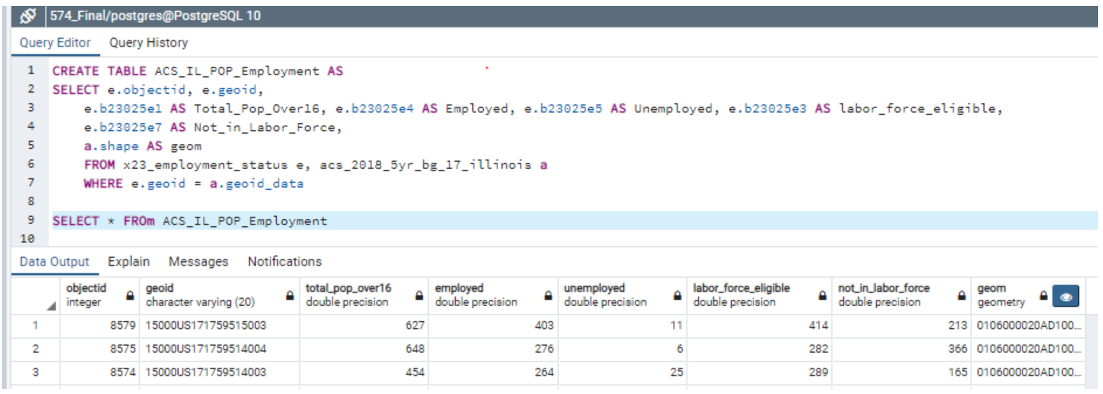
e.b23025e5 AS Unemployed, e.b23025e3 AS labor_force_eligible,

e.b23025e7 AS Not_in_Labor_Force, a.shape AS geom

FROM x23_employment_status e, acs_2018_5yr_bg_17_illinois a

WHERE e.geoid = a.geoid_data

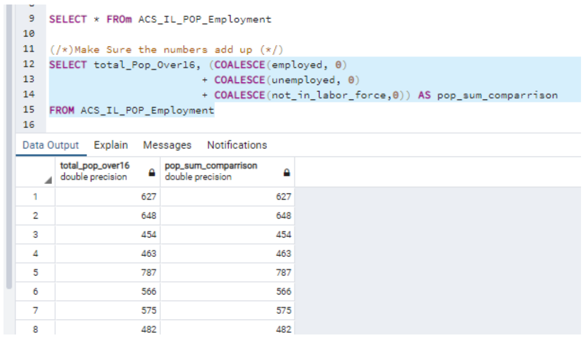
Figure 3-4



```sql
CREATE TABLE ACS_IL_POP_Employment AS
SELECT e.objectid, e.geoid,
    e.b23025e1 AS Total_Pop_Over16, e.b23025e4 AS Employed, e.b23025e5 AS Unemployed, e.b23025e3 AS labor_force_eligible,
    e.b23025e7 AS Not_in_Labor_Force,
    a.shape AS geom
    FROM x23_employment_status e, acs_2018_5yr_bg_17_illinois a
    WHERE e.geoid = a.geoid_data

SELECT * FROm ACS_IL_POP_Employment
```

	objectid integer	geoid character varying (20)	total_pop_over16 double precision	employed double precision	unemployed double precision	labor_force_eligible double precision	not_in_labor_force double precision	geom geometry
1	8579	15000US171759515003	627	403	11	414	213	0106000020AD100...
2	8575	15000US171759514004	648	276	6	282	366	0106000020AD100...
3	8574	15000US171759514003	454	264	25	289	165	0106000020AD100...

As you can see from the screen shot (Figure 3-4), the values that were important were transferred over to a new table within the database and given more practical names based off the shared attribute. Then an analysis was performed to compare the totals of the subcategories against the total proposed over 16 years old. The COALESCE function was used to find the sum of the subset values.



```sql
SELECT * FROm ACS_IL_POP_Employment

(/*)Make Sure the numbers add up (*/)
SELECT total_Pop_Over16, (COALESCE(employed, 0)
                        + COALESCE(unemployed, 0)
                        + COALESCE(not_in_labor_force,0)) AS pop_sum_comparrison
FROM ACS_IL_POP_Employment
```

	total_pop_over16 double precision	pop_sum_comparrison double precision
1	627	627
2	648	648
3	454	454
4	463	463
5	787	787
6	566	566
7	575	575
8	482	482

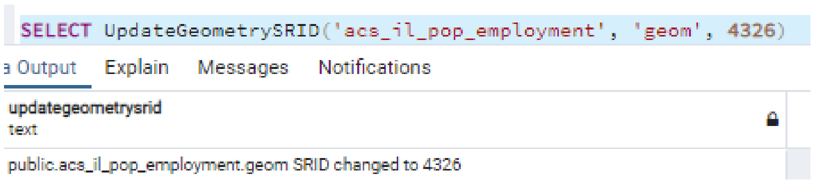
The next script is multifunctional. Firstly, it is a mock ‘SELECT BY LOCATION’ function between the ACS data and the Chicago data. It was this research’s best attempt to do a ‘SELECT BY LOCATION-HAVE THEIR CENTROID IN THE SOURCE LAYER FEATURE’. Through some ArcMap experimenting, the centroid method of selecting by location produced the most accurate results compared to INTERSECT, WITHIN, share a line with, etc. Secondly, the script creates a standalone layer and carries over the important attribute data, layer constraints and both layers geometries. Lastly it does an equation to find the percentage of unemployed per the eligible population.

```
CREATE TABLE ACS_IN_CHICAGO_Employment_ AS
select b.objectid, b.ward, b.shape AS ward_shape,
        a.geoid, a.geom, a.total_pop_over16, a.employed,
        a.unemployed, a.labor_force_eligible, a.not_in_labor_force,
        a.unemployed/ NULLIF(a.labor_force_eligible,0) AS Unemployed_Per_Eligible
FROM ACS_IL_POP_Employment a, chicago_wards b
WHERE a.geom && b.shape AND ST_Intersects(ST_Centroid(a.geom),b.shape)
```

But prior to the implementation of this SELECT BY LOCATION script, the SRID (coordinate reference systems) must add up (Figure 3-5). As mentioned before, the ACS data uses GCS_North_American_1983 WKID: 4269 projection. So the following script had to be used to adjust the ACS data to the WGS84 WKID: 4326 Coordinate System (either dataset could have been manipulated for this to work):

```
        SELECT UpdateGeometrySRID('acs_il_pop_employment', 'geom', 4326)
```

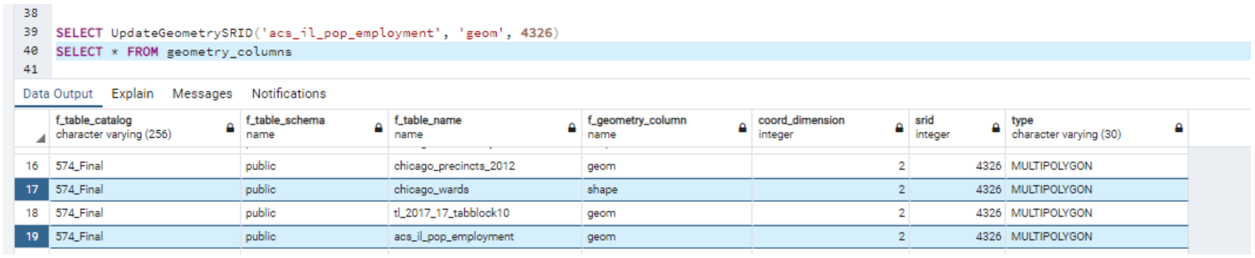
Figure 3-5



```
SELECT UpdateGeometrySRID('acs_il_pop_employment', 'geom', 4326)
```

Output | Explain | Messages | Notifications

```
updategeometrysrid
text
public.acs_il_pop_employment.geom SRID changed to 4326
```



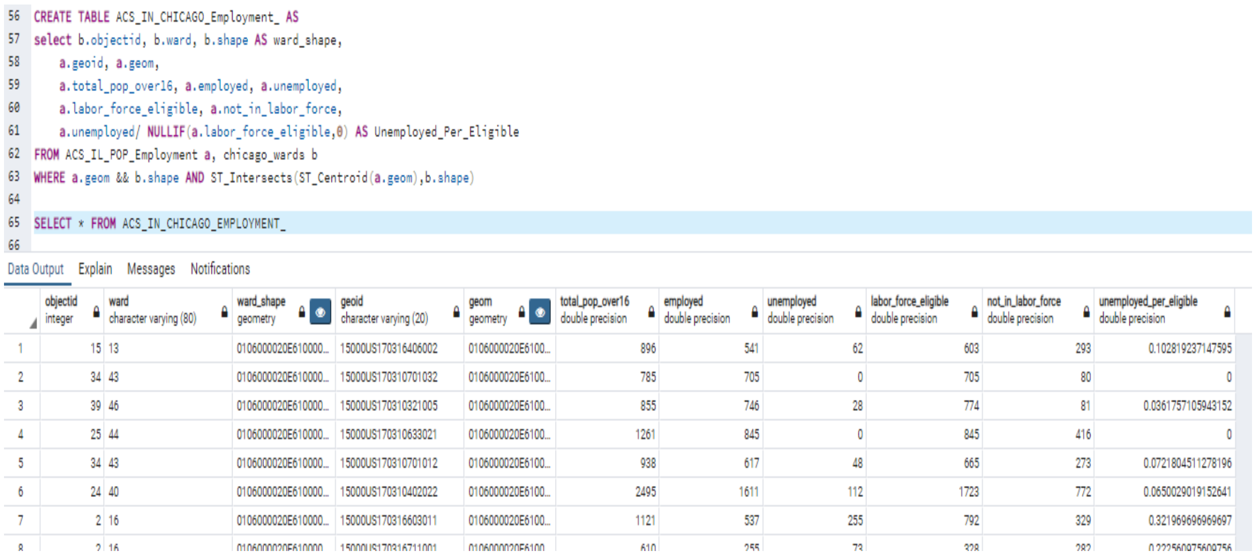
```
SELECT UpdateGeometrySRID('acs_il_pop_employment', 'geom', 4326)
SELECT * FROM geometry_columns
```

Data Output | Explain | Messages | Notifications

	f_table_catalog character varying (256)	f_table_schema name	f_table_name name	f_geometry_column name	coord_dimension integer	srid integer	type character varying (30)
16	574_Final	public	chicago_precincts_2012	geom	2	4326	MULTIPOLYGON
17	574_Final	public	chicago_wards	shape	2	4326	MULTIPOLYGON
18	574_Final	public	tl_2017_17_tabblock10	geom	2	4326	MULTIPOLYGON
19	574_Final	public	acs_il_pop_employment	geom	2	4326	MULTIPOLYGON

Once the coordinate systems matched up (see above graphic) then the PostGIS, prior to the coordinate system transition, was then effective. The resulting table is depicted in the following image:

Figure 3-6



```
CREATE TABLE ACS_IN_CHICAGO_Employment_ AS
select b.objectid, b.ward, b.shape AS ward_shape,
    a.geoid, a.geom,
    a.total_pop_over16, a.employed, a.unemployed,
    a.labor_force_eligible, a.not_in_labor_force,
    a.unemployed/ NULLIF(a.labor_force_eligible,0) AS Unemployed_Per_Eligible
FROM ACS_IL_POP_Employment a, chicago_wards b
WHERE a.geom && b.shape AND ST_Intersects(ST_Centroid(a.geom),b.shape)

SELECT * FROM ACS_IN_CHICAGO_EMPLOYMENT_
```

Data Output Explain Messages Notifications

	objectid integer	ward character varying (80)	ward_shape geometry	geoid character varying (20)	geom geometry	total_pop_over16 double precision	employed double precision	unemployed double precision	labor_force_eligible double precision	not_in_labor_force double precision	unemployed_per_eligible double precision
1	15	13	0106000020E610000...	15000US170316406002	0106000020E6100...	896	541	62	603	293	0.102819237147595
2	34	43	0106000020E610000...	15000US170310701032	0106000020E6100...	785	705	0	705	80	0
3	39	46	0106000020E610000...	15000US170310321005	0106000020E6100...	855	746	28	774	81	0.0361757105943152
4	25	44	0106000020E610000...	15000US170310633021	0106000020E6100...	1261	845	0	845	416	0
5	34	43	0106000020E610000...	15000US170310701012	0106000020E6100...	938	617	48	665	273	0.0721804511278196
6	24	40	0106000020E610000...	15000US170310402022	0106000020E6100...	2495	1611	112	1723	772	0.065002901915264
7	2	16	0106000020E610000...	15000US170316603011	0106000020E6100...	1121	537	255	792	329	0.321969696969697
8	2	16	0106000020E610000	15000US170316711001	0106000020E6100	610	255	73	328	282	0.222560975609756

Then a similar equation was used to extract the ACS Tiger values, within the boundary of Chicago, having surveyed populations where the total number of unemployed is larger than the employed population. The division function between the unemployment per population was removed because the point of the following script is to export only the values that have that higher level of unemployment to eventually highlight on the map.

CREATE TABLE ACS_IN_CHICAGO_Higher_Unemployed_ AS

select b.ward AS Ward_Number,

a.geoid AS ACS_Geoid, a.geom AS geom,

a.total_pop_over16, a.employed, a.unemployed,

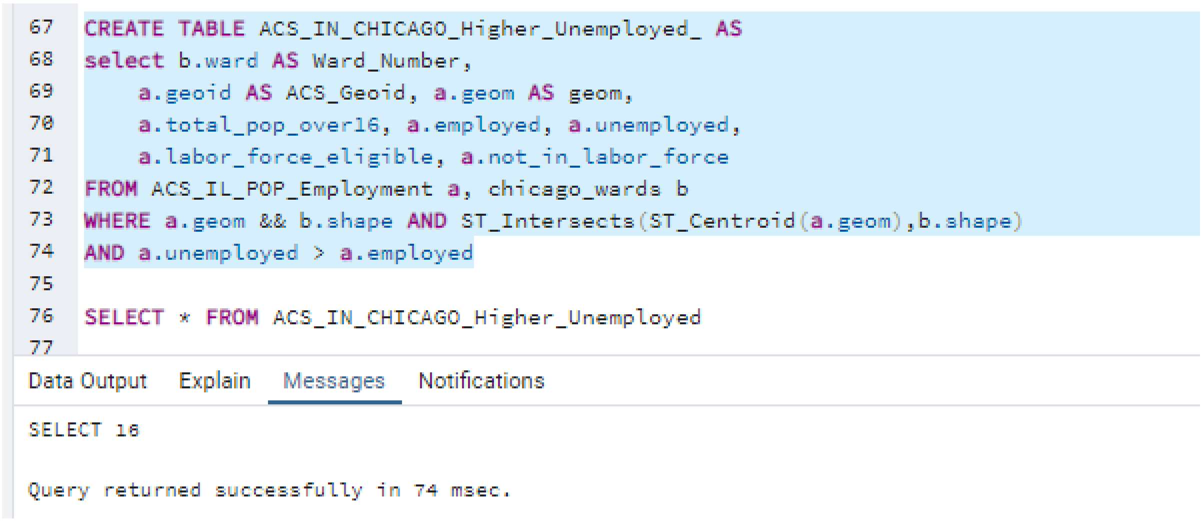
a.labor_force_eligible, a.not_in_labor_force

FROM ACS_IL_POP_Employment a, chicago_wards b

WHERE a.geom && b.shape AND ST_Intersects(ST_Centroid(a.geom),b.shape)

AND a.unemployed > a.employed

Figure 3-7



```
CREATE TABLE ACS_IN_CHICAGO_Higher_Unemployed_ AS
select b.ward AS Ward_Number,
    a.geoid AS ACS_Geoid, a.geom AS geom,
    a.total_pop_over16, a.employed, a.unemployed,
    a.labor_force_eligible, a.not_in_labor_force
FROM ACS_IL_POP_Employment a, chicago_wards b
WHERE a.geom && b.shape AND ST_Intersects(ST_Centroid(a.geom),b.shape)
AND a.unemployed > a.employed

SELECT * FROM ACS_IN_CHICAGO_Higher_Unemployed
```

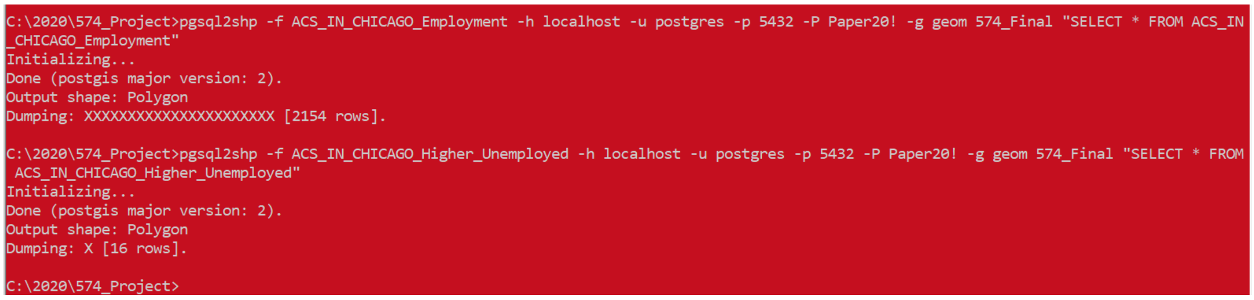
```
SELECT 16

Query returned successfully in 74 msec.
```

These two layers together, and similar analysis, leads to great comparison of demographic data, especially related to gerrymandering and voting districts. Any of this methodology can be repeated to include any of the attribute information found in the initial ACS dataset.

Now the layers have to be exported out of their PostGIS tables and into a shapefile format for further analysis and display within a GIS framework. This is done using the command prompt and the pgsql2shp function for each of the new tables. The relative paths were set and the following functions created the results in the corresponding snapshot of the command prompt:

pgsql2shp -f ACS_IN_CHICAGO_Employment_ -h localhost -u postgres -p 5432 -P Paper20! -g geom 574_Final "SELECT * FROM ACS_IN_CHICAGO_Employment_"

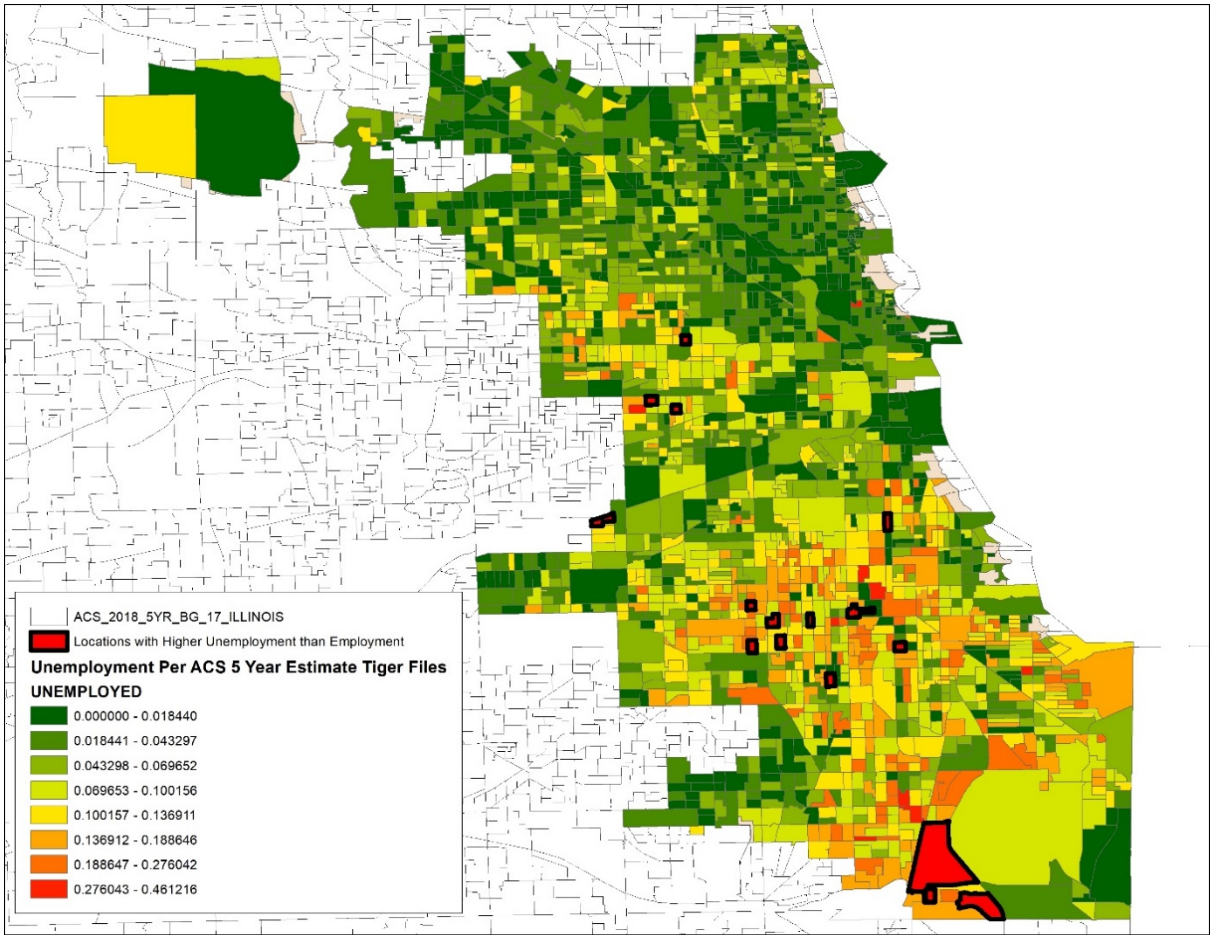
pgsql2shp -f ACS_IN_CHICAGO_Higher_Unemployed_ -h localhost -u postgres -p 5432 -P Paper20! -g geom 574_Final "SELECT * FROM ACS_IN_CHICAGO_Higher_Unemployed_"



```
C:\2020\574_Project>pgsql2shp -f ACS_IN_CHICAGO_Employment -h localhost -u postgres -p 5432 -P Paper20! -g geom 574_Final "SELECT * FROM ACS_IN
_CHICAGO_Employment"
Initializing...
Done (postgis major version: 2).
Output shape: Polygon
Dumping: XXXXXXXXXXXXXXXXXXXXXX [2154 rows].

C:\2020\574_Project>pgsql2shp -f ACS_IN_CHICAGO_Higher_Unemployed -h localhost -u postgres -p 5432 -P Paper20! -g geom 574_Final "SELECT * FROM
 ACS_IN_CHICAGO_Higher_Unemployed"
Initializing...
Done (postgis major version: 2).
Output shape: Polygon
Dumping: X [16 rows].

C:\2020\574_Project>
```

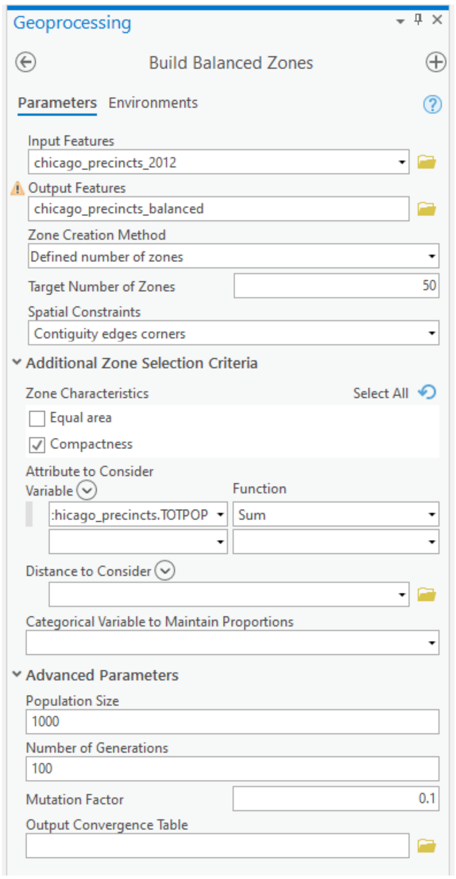
The shapefiles were placed into the GIS, ArcMap. ACS_IN_CHICAGO_Employment layer was classified using with an 8 class Jenks Natural Breaks methodology and a gradual color symbology. With red representing areas with high levels of unemployment, while the green represents the areas in the city with the highest level of employment per person.

Figure 3-8



Arc Pro has a GP tool called build balanced areas. It allows you to take a point or polygon dataset and try to divide it into equal areas based on attributes and/or area. For this project we used the precincts and tried to recreate the 50 wards of Chicago based on precinct population and optimized for compactness. Adjusting the tools population and generations in Advanced properties from the default 100 and 50 to 1000 and 100 created more equal populations bringing the largest zone from 95,000 down to 85,000. One downside with this tool is that ESRI does not disclose the math it used when creating these zones.

Figure 3-9 through 3-11



Geoprocessing

Build Balanced Zones

Parameters Environments

Input Features: chicago_precincts_2012

Output Features: chicago_precincts_balanced

Zone Creation Method: Defined number of zones

Target Number of Zones: 50

Spatial Constraints: Contiguity edges corners

Additional Zone Selection Criteria

Zone Characteristics (Select All)

- [ ] Equal area
- [x] Compactness

Attribute to Consider

Variable	Function
:hicago_precincts.TOTPOP	Sum

Distance to Consider:

Categorical Variable to Maintain Proportions:

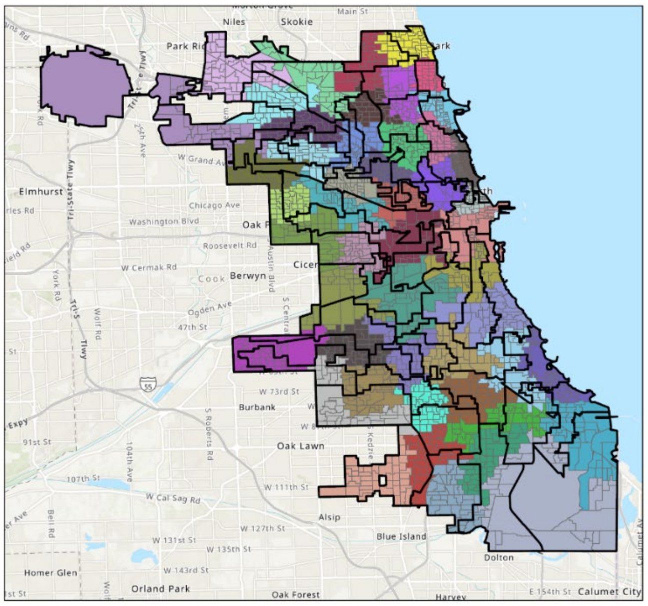
Advanced Parameters

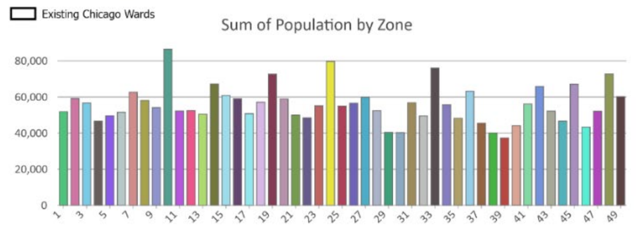
Population Size: 1000

Number of Generations: 100

Mutation Factor: 0.1

Output Convergence Table:



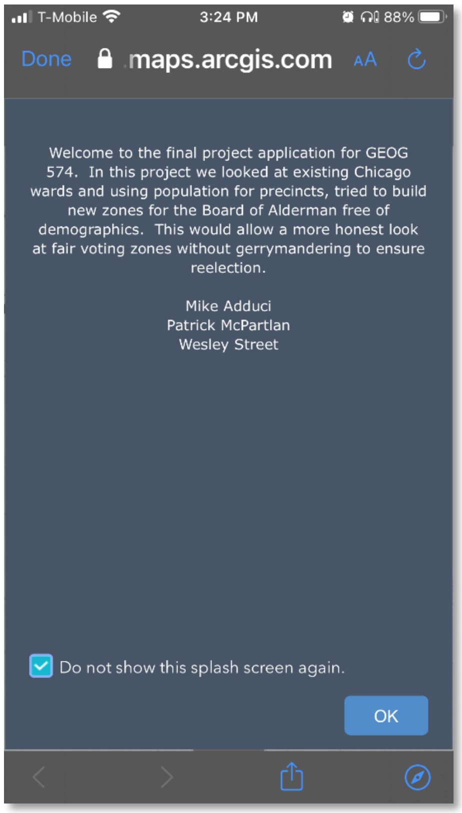


MOBILE/WEB APPLICATION:

Since the internet is the easiest way to find and share data, we used UW-M ArcGIS Online site to build a webapp and share it publicly. It displays the same information as the stand-alone Gerrymandering map but in an interactive format (Figure 4-1). The app can be accessed at the following link:

https://uw-mad.maps.arcgis.com/apps/webappviewer/index.html?id=e3ee28cf5058469995769df923fcd8bf

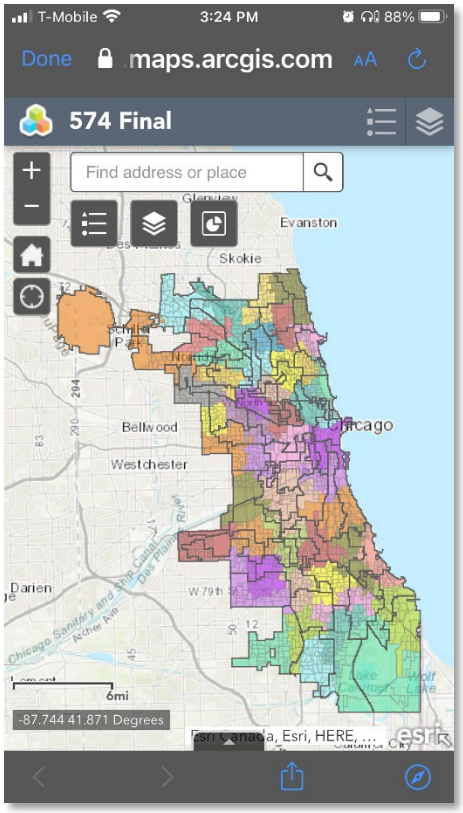
Figure 4-1

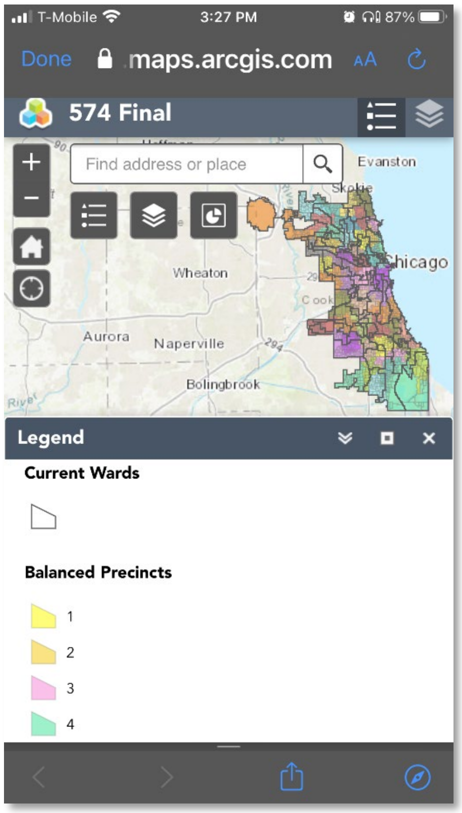


Welcome to the final project application for GEOG 574. In this project we looked at existing Chicago wards and using population for precincts, tried to build new zones for the Board of Alderman free of demographics. This would allow a more honest look at fair voting zones without gerrymandering to ensure reelection.

Mike Adduci
Patrick McPartlan
Wesley Street

✓ Do not show this splash screen again.

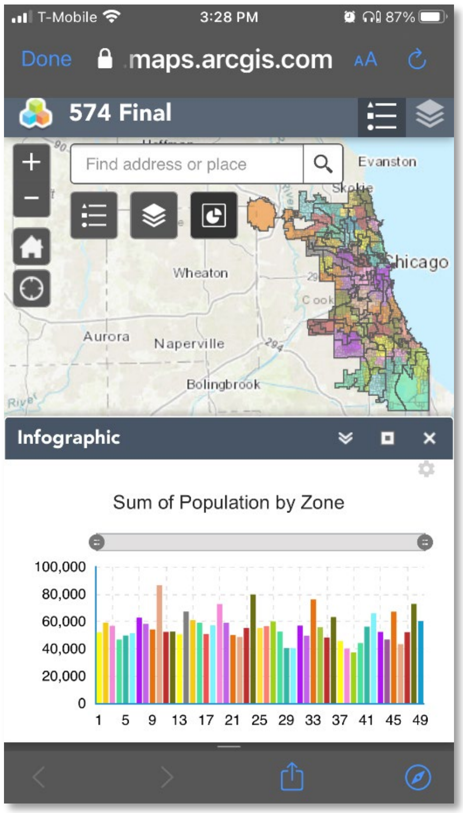




Current Wards

Balanced Precincts

🟨	1
🟨	2
🟪	3
🟩	4



Results & Conclusion

The resulting database enables the City of Chicago to implement more equitable and data-driven ward boundaries using an intelligent combination of population and demographic data from both the City of Chicago and the U.S. Census Bureau's open and free data files.

Moreover, the overall geometry of the new ward boundaries are more normalized as they do not employ gerrymandering techniques or any political decisions in their inception which may produce obtuse geometries and unusual patterns.

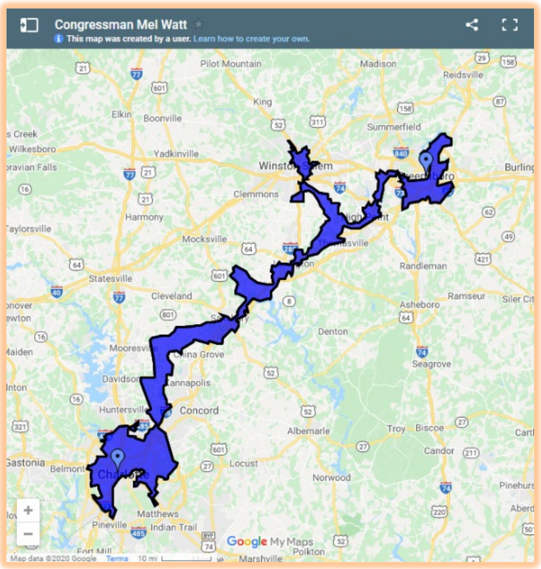
As a result, the data-driven approach has created both a repeatable and trustworthy method for drawing or re-drawing boundaries in a dynamic fashion which takes into account a changing demographic in terms of population and any other factor which the Census reveals.

Future Applications of Concept

Potential opportunities for expansion of these database methods is applicable at the state level, even outside of Illinois. Due to the fact that Illinois Census data was used, the same analysis could be re-done with congressional districts at the state level and have usability in all U.S. states and territories across the nation.

A prime example of this is in North Carolina's 12th Congressional District, referred to as the "snake-shaped district" in an article by The Atlantic (Figure 4-2).

Figure 4-2



Reference Citations

"American Community Survey." U.S. Census Bureau, U.S. Census Bureau, www.census.gov/acs/www/data/data-tables-and-tools/data-profiles/2018. Accessed 12 Jan. 2020.

"Just Look at FHFA Pick Mel Watts's Awful Gerrymandered District." The Atlantic [Boston, MA], 1 May 2013, www.theatlantic.com/politics/archive/2013/05/just-look-at-fhfa-pick-mel-wattss-awful-gerrymandered-district/275485.

"Study of Reform Proposals for Chicago City Council." MMMG (Metric Geometry & Gerrymandering Group), MGGG Redistricting Lab, mggg.org/chicago. Accessed 3 Dec. 2020.