Here, we will be learning some geocomputation and geovisualization

First, lets pull in the correct modules and read in the data

In [16]:

```
#Modules
%matplotlib inline
import matplotlib
import numpy as np
import matplotlib.pyplot as plt
import geopandas as gpd

#Data
routes_df = gpd.read_file('data/Truck_Route_Network.shp')

tracts_df = gpd.read_file('data/clinics.shp')
```

Fix The Tracts

In [4]:

```
tracts_df['dummy'] = 1.0
county = tracts_df.dissolve(by='dummy')
```

Now we will select all the roads who intersect with any of the tracts, and then limit those roads just it only has them within the tracts, not moving out of it

In [5]:

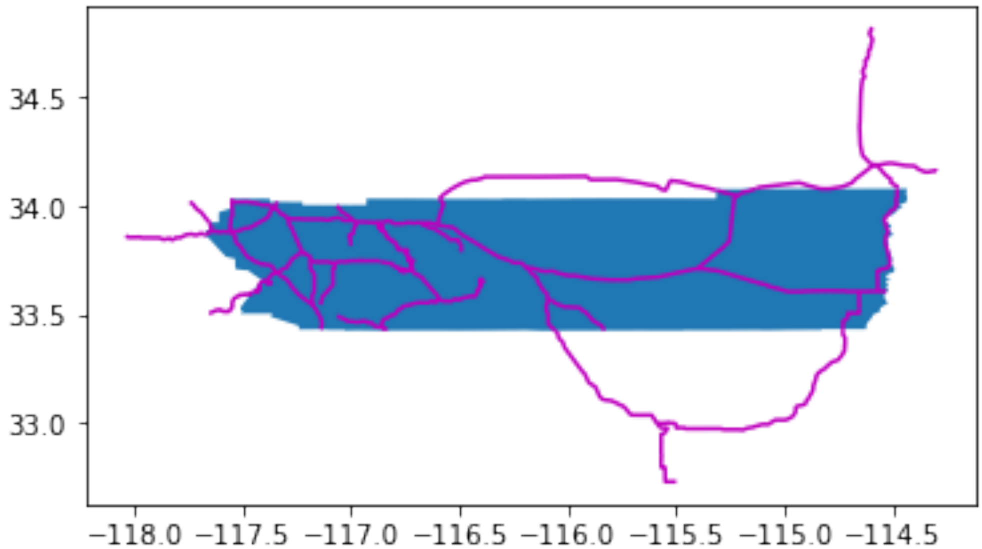
```
#Rc_Routes are all roads that at some point intersect with a tra
ct
r = routes_df['geometry']
rc_routes = r[r.apply(lambda x: x.intersects(county.iloc[0]['geo
metry']))]

#rc_hw are all roads that are within the tracts
geoms = []
for idx, route in enumerate(rc_routes):
    print(idx)
    geoms.append(route.intersection(county.iloc[0]['geometry']))
rc_hw = gpd.GeoSeries(geoms)
rc_hw = gpd.GeoDataFrame({'geometry': rc_hw})
```

Lets compare the two visually

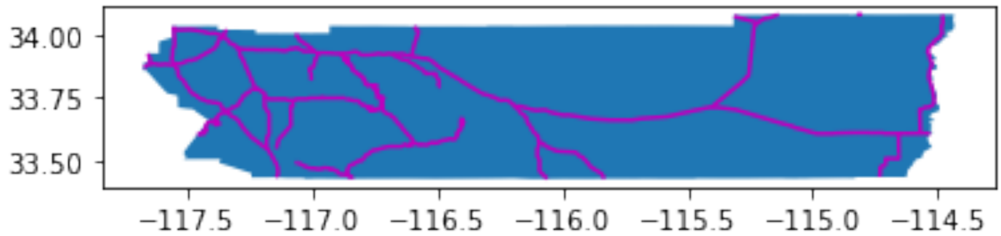
In [6]:

```
ax = plt.gca()
rc_routes.plot(ax=ax,edgecolor='m')
county.plot(ax=ax)
plt.show()
```



In [7]:

```
ax = plt.gca()
rc_hw.plot(ax=ax,edgecolor='m')
county.plot(ax=ax)
plt.show()
```



Create CRS for rc_hw and set the crs to the same as the tracts

In [8]:

```
rc_hw.crs = tracts_df.crs # create a crs for the rc_hw
rc_hw = rc_hw.to_crs(tracts_df.crs)
```

Now, create a new data frame that shows all the tracts that interect with a road We will then create another new data frame that will have true false values for whether or not there is a road intersecting it. We will then return to the old data frame, and make a collumn that will have 0 or 1 values for the intersection of a road.

In [9]:

```
tracts_with_roads = gpd.sjoin(tracts_df, rc_hw, how='inner', op=
'intersects')

# Let's create an indicator (dummy) variable for use later
import numpy as np
geoids = tracts_df['GEOID10'].values
tract_hw = np.array([geoid in tracts_with_roads['GEOID10'].value
s for geoid in geoids])



tracts_df['intersectshw'] = tract_hw*1.
```

Next we will look at one particular city, and look at the roads that at some point intersect that city.

In [10]:

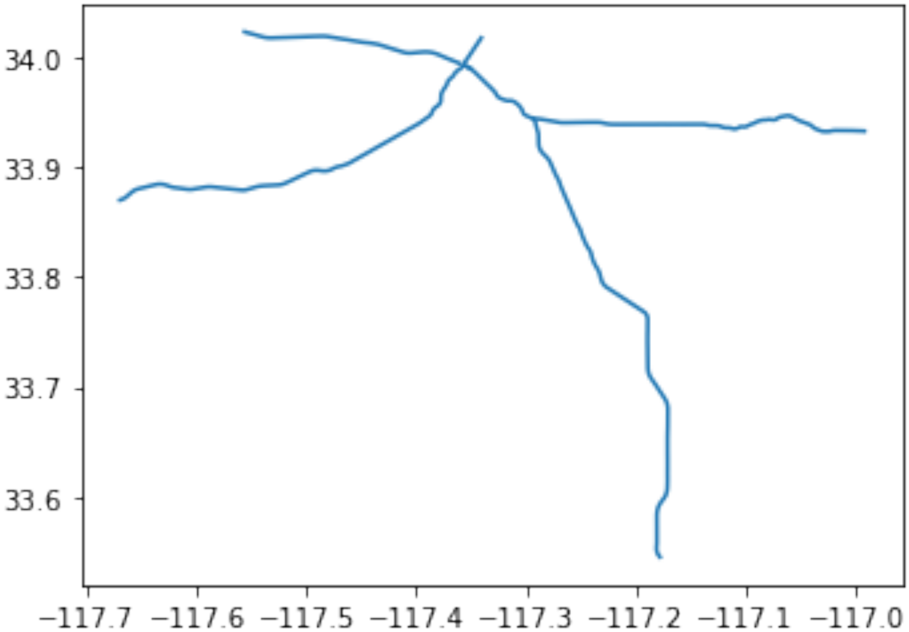
```
#City
city = gpd.read_file('data/riverside_city.shp')



#Roads that intersect that city
r = routes_df['geometry']
r.apply(lambda x: x.intersects(city.iloc[0]['geometry']))
city_routes = r[r.apply(lambda x: x.intersects(city.iloc[0]['geo
metry']))]
city_routes.plot()
```

Out[10]:

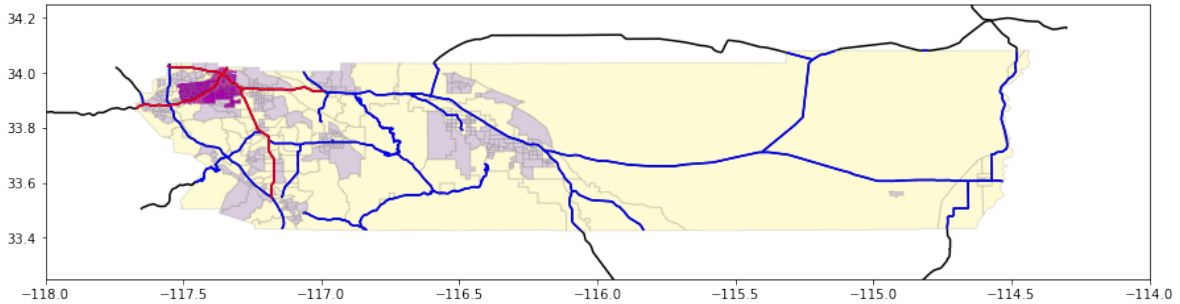
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5d2031
5b00>
```



Okay, lets put it all together. The following map shows a lot of information, so stick with me.

In [11]:

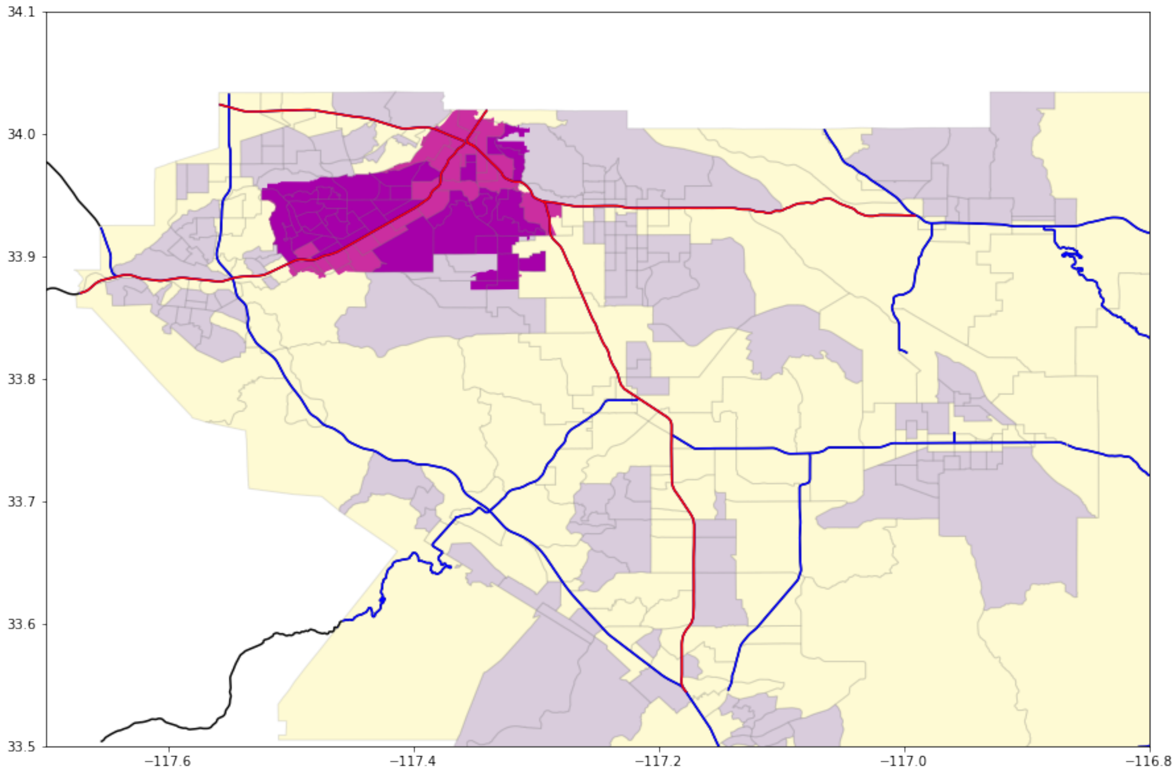
```
#Setting figure size
plt.rcParams['figure.figsize'] = (15, 15)
#Setting an axis
ax = plt.gca()
#Roads that intersect with a tract but are outside the county bo
undaries
rc_routes.plot(ax=ax, edgecolor='k')
#Roads that intersect with a tract and stay within the county th
e whole time
rc_hw.plot(ax=ax, edgecolor='b')
#The city limits
city.plot(ax=ax, color="m")
#The county
tracts_df.plot(ax=ax, column='intersectshw',edgecolor='grey', al
pha=0.2)
#Roads that go through the city
city_routes.plot(ax=ax, color="red")
#Setting the scale of the map
ax.set_xlim(-118.0, -114.0); ax.set_ylim(33.25, 34.25)
ax.set_aspect('equal')
plt.show()
```



Now, lets zoom in, and make the city a little bit clearler. Can you see the coding difference between this and the last image?

In [12]:

```
plt.rcParams['figure.figsize'] = (15, 15)
ax = plt.gca()
rc_routes.plot(ax=ax, edgecolor='k')
rc_hw.plot(ax=ax, edgecolor='b')
city.plot(ax=ax, color="m")
tracts_df.plot(ax=ax, column='intersectshw',edgecolor='grey', al
pha=0.2)
city_routes.plot(ax=ax, color="red")
ax.set_xlim(-117.7, -116.8); ax.set_ylim(33.5, 34.1)
ax.set_aspect('equal')
plt.show()
```



Now that we have done that, lets fous on geovisualization of data frames. First load in modules are read data

In [13]:

```
import matplotlib.pyplot as plt # make plot larger
from pysal.viz import mapclassify
import geopandas as gpd
import matplotlib.pyplot as plt
import geopandas as gpd
shp_link = "data/texas.shp"
tx = gpd.read_file(shp_link)
```

```
/srv/conda/envs/notebook/lib/python3.6/site-packages
/pysal/explore/segregation/network/network.py:16: Us
erWarning: You need pandana and urbanaccess to work
with segregation's network module
You can install them with  `pip install urbanaccess
pandana` or `conda install -c udst pandana urbanacce
ss`
  "You need pandana and urbanaccess to work with seg
regation's network module\n"
/srv/conda/envs/notebook/lib/python3.6/site-packages
/pysal/model/spvcm/abstracts.py:10: UserWarning: The
`dill` module is required to use the sqlite backend
fully.
  from .sqlite import head_to_sql, start_sql
```

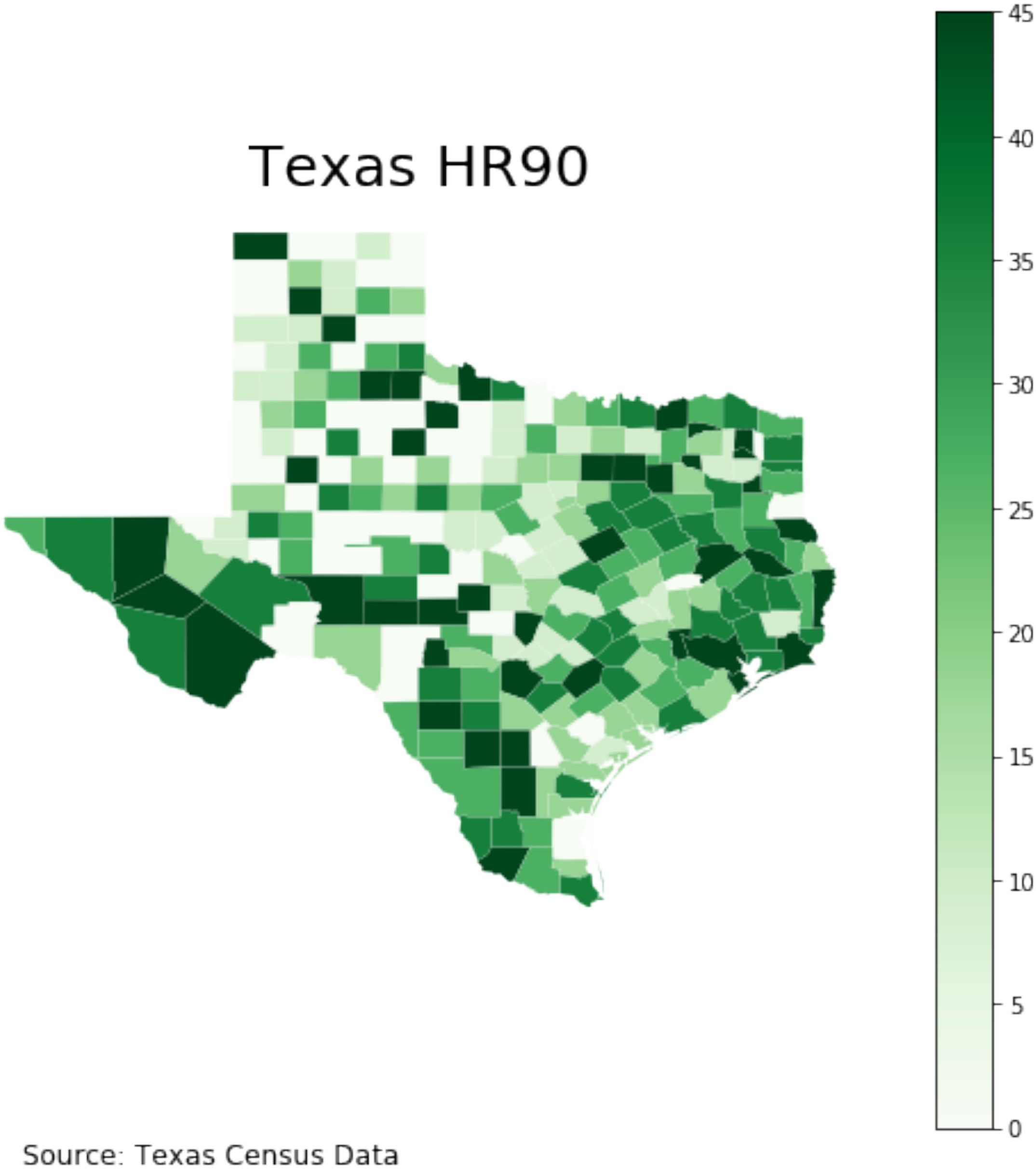
Now I will show you a final visualization, that is quite complicated, but I will walk you through it

In [14]:

```
variable= 'HR90'
vmin,vmax = 0,45
f, ax = plt.subplots(1, figsize=(9, 9))
tx.plot(column=variable, scheme='QUANTILES', \
        k=6, cmap='Greens', linewidth=0.1, ax=ax, \
        edgecolor='white', legend=False)

ax.set_axis_off()
ax.set_title("Texas HR90", fontdict={'fontsize': '25', 'fontweig
ht' : '3'})
ax.annotate('Source: Texas Census Data',fontsize=12,xy=(0.1, .08
), xycoords='figure fraction',
            horizontalalignment='left', verticalalignment='top')

sm = plt.cm.ScalarMappable(cmap='Greens', norm=plt.Normalize(vmi
n=vmin, vmax=vmax))
# empty array for the data range
sm._A = []
# add the colorbar to the figure
cbar = f.colorbar(sm)
```



The second part of the geovisualizations lab did not work for me, it simply did not run. I read through and undrerstand it relatively well, it just I cannot draw on it for my extension here.

I hope this lesson helps you a little bit with geovisualization and geocompuation! The information in this tutorial should help you find intersecting geometries, give you a solid basis with the .plot fuction to better visualize data, and give you a taste of what python can do.

In [ ]:

```

```

In [ ]:

```

```