SOCIETY OF ACTUARIES
Antitrust Compliance Guidelines

Active participation in the Society of Actuaries is an important aspect of membership. While the positive contributions of professional societies and associations are well-recognized and encouraged, association activities are vulnerable to close antitrust scrutiny. By their very nature, associations bring together industry competitors and other market participants.

The United States antitrust laws aim to protect consumers by preserving the free economy and prohibiting anti-competitive business practices; they promote competition. There are both state and federal antitrust laws, although state antitrust laws closely follow federal law. The Sherman Act, is the primary U.S. antitrust law pertaining to association activities. The Sherman Act prohibits every contract, combination or conspiracy that places an unreasonable restraint on trade. There are, however, some activities that are illegal under all circumstances, such as price fixing, market allocation and collusive bidding.

There is no safe harbor under the antitrust law for professional association activities. Therefore, association meeting participants should refrain from discussing any activity that could potentially be construed as having an anti-competitive effect. Discussions relating to product or service pricing, market allocations, membership restrictions, product standardization or other conditions on trade could arguably be perceived as a restraint on trade and may expose the SOA and its members to antitrust enforcement procedures.

While participating in all SOA in person meetings, webinars, teleconferences or side discussions, you should avoid discussing competitively sensitive information with competitors and follow these guidelines:

- Do not discuss prices for services or products or anything else that might affect prices
- Do not discuss what you or other entities plan to do in a particular geographic or product markets or with particular customers.
- Do not speak on behalf of the SOA or any of its committees unless specifically authorized to do so.
- Do leave a meeting where any anticompetitive pricing or market allocation discussion occurs.
- Do alert SOA staff and/or legal counsel to any concerning discussions
- Do consult with legal counsel before raising any matter or making a statement that may involve competitively sensitive information.

Adherence to these guidelines involves not only avoidance of antitrust violations, but avoidance of behavior which might be so construed. These guidelines only provide an overview of prohibited activities. SOA legal counsel reviews meeting agenda and materials as deemed appropriate and any discussion that departs from the formal agenda should be scrutinized carefully. Antitrust compliance is everyone’s responsibility; however, please seek legal counsel if you have any questions or concerns.
Presentation Disclaimer

Presentations are intended for educational purposes only and do not replace independent professional judgment. Statements of fact and opinions expressed are those of the participants individually and, unless expressly stated to the contrary, are not the opinion or position of the Society of Actuaries, its cosponsors or its committees. The Society of Actuaries does not endorse or approve, and assumes no responsibility for, the content, accuracy or completeness of the information presented. Attendees should note that the sessions are audio-recorded and may be published in various media, including print, audio and video formats without further notice.
Introduction
A new perspective with geospatial analytics

- Identify relationships, spot trends and make inferences that may not be available in tabular form
- Insights can be used as coefficients in predictive models
- A robust geospatial capability may require hiring GIS specialists and investing in specialized software
- Today we’ll discuss using open source geospatial methods as a proof of concept for integrating geospatial analytics into your business area
Choropleth Map
CMS Data Entrepreneurs File is used for training purposes only

CMS Data Entrepreneurs File (DE-SynPUF) is a 5% sample of 2008 Medicare beneficiaries with significant alterations applied to beneficiary and claims data. Per the [DE-SynPUF User Manual](#):

- All beneficiaries are synthetic beneficiaries, meant to represent actual beneficiaries, but based on at least 3 Medicare beneficiaries.
- Rare records are suppressed and dates are perturbed.
- All codes identifying providers and physicians are random numbers/characters with no association to any known id number.
- Variable names are the same as those in actual Medicare data unless they begin with “SP_”. SP variables were significantly coarsened to decrease re-identification risk.

Due to the alterations that have been applied to the 5% sample, analytics insights from this data are of limited value.
Interactive Choropleth Map

A map with geographic regions shaded by a metric

• Regions can be states, counties, zip codes, census tracts, or custom defined regions such as neighborhoods

• Choropleth maps help to spot patterns in geographic clusters, for instance:
  • Does northeast region have higher healthcare spend than central region?
  • Do urban areas have higher rates of healthcare spend than rural areas?

• Code to create the map is saved here on GitHub.

What we’ll create:

• Map of Emergency Room visits per thousand ("ER/K") by county
• Overlay of de-identified facilities
# Data Requirements

## Definition

The variable that we’re interested in, summarized by geographic region.

## Sample Project Input

<table>
<thead>
<tr>
<th>county</th>
<th>ssastate</th>
<th>fipsstate</th>
<th>ER_K</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATLANTIC</td>
<td>31</td>
<td>34</td>
<td>266.0</td>
</tr>
<tr>
<td>BERGEN</td>
<td>31</td>
<td>34</td>
<td>230.0</td>
</tr>
<tr>
<td>BURLINGTON</td>
<td>31</td>
<td>34</td>
<td>201.0</td>
</tr>
<tr>
<td>CAMDEN</td>
<td>31</td>
<td>34</td>
<td>188.0</td>
</tr>
<tr>
<td>CAPE MAY</td>
<td>31</td>
<td>34</td>
<td>229.0</td>
</tr>
</tbody>
</table>

## Data Sources

- NJ Medicare ER/K by county from [CMS Data Entrepreneurs](#) files
  - Beneficiary Summary File used to pull de-identified beneficiaries by SSA county
  - Carrier Claims File used to pull ER claims (revenue code 99281 – 99285)
- Crosswalk of SSA to FIPS county code

- Coordinates of NJ counties from [Census 2008 Shapefiles](#)
  - The Census provides Shapefiles of many legal and statistical regions
  - If you cannot obtain the region you’re interested in from the census Shapefiles then geographic coordinates in these formats can be used.
## Package Requirements

<table>
<thead>
<tr>
<th>Packages</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pandas</td>
<td>Easy to use data structures and data analysis tools</td>
</tr>
<tr>
<td>Geopandas</td>
<td>Makes it easier to work with geospatial data in python</td>
</tr>
<tr>
<td>Folium</td>
<td>Leverages the JavaScript library Leaflet through python</td>
</tr>
<tr>
<td>Branca</td>
<td>Hosts the non-map-specific features of Folium</td>
</tr>
</tbody>
</table>
Create Map | Load Inputs & Merge

Code

```python
import pandas as pd
import geopandas as gpd

## Load ER visits per K by FIPS county
ER_rate_county = pd.read_csv("data\NJ_ER_County_2008.csv")

# Load Census county shape file
counties = gpd.read_file("gis_data\tl_2008_34_county\tl_2008_34_county.shp")

## concatenate state and county code to join to metric table
counties['fipscountyint'] = counties['STATEFP'].astype(str) + counties['COUNTYFP'].astype(str)
counties['fipscountyint'] = counties['fipscountyint'].astype(int)

### merge the variables of interest into the Geodataframe
counties = counties.merge(ER_rate_county, left_on='fipscountyint', right_on='fipscounty', how='left')
```

### Merge Output

<table>
<thead>
<tr>
<th>NAME</th>
<th>fipscountyint</th>
<th>geometry</th>
<th>ER_K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atlantic</td>
<td>34001</td>
<td>POLYGON (41.74.76451 39.20662, -74.74466 39.296...)</td>
<td>266.0</td>
</tr>
<tr>
<td>Bergen</td>
<td>34003</td>
<td>POLYGON (41.74.06249 40.80496, -74.06513 40.803...)</td>
<td>230.0</td>
</tr>
<tr>
<td>Burlington</td>
<td>34005</td>
<td>POLYGON (41.74.73772 40.13746, -74.73748 40.137...)</td>
<td>201.0</td>
</tr>
<tr>
<td>Camden</td>
<td>34007</td>
<td>POLYGON (41.75.12225 39.87385, -75.12255 39.873...)</td>
<td>188.0</td>
</tr>
<tr>
<td>Cape May</td>
<td>34009</td>
<td>POLYGON (41.74.76555 39.29754, -74.76549 39.297...)</td>
<td>229.0</td>
</tr>
</tbody>
</table>

The geometry column in the geopandas dataframe contains the outline of each of the NJ counties in latitude and longitude coordinates.
import branca

## Load ER visits per K by FIPS county
colorscale = branca.colormap.linear.YlOrRd_09.scale(
counties[variable].min(), counties[variable].max())

import folium

## identifies the center point of all the counties
centroid = counties.geometry.centroid

## initiaes a map based on the centroid
m = folium.Map(location = [centroid.y.mean(), centroid.x.mean()], zoom_start=8)
CHOROPLETH MAP

Create Map | Color Counties by ER/K

Code

colorscale.add_to(m) ## add colorscale to make legend

choropleth_counties = folium.GeoJson(
    counties, ## GeoPandas dataframe
    name = 'Counties',

    ## style function controls fill of counties
    style_function = lambda x: {'weight':1, 'color': '#545453',
                              'fillColor':colorscale(x['properties']['ER_K']) ## apply colorscale based on var
                             , 'fillOpacity': 0.5},

    ## changes styling of geo regions upon hover
    highlight_function = lambda x: {'weight':3, 'color':'black', 'fillOpacity': 1},

    ## tooltip can include info from any col in the GeoPandas dataframe
    tooltip = folium.features.GeoJsonTooltip(
        fields = ['name','ER_K'],
        aliases = ['County:','ER Visits per K'])
)

m.add_child(choropleth_counties)
Isochrone Map
Interactive Isochrone Map

Travel time maps– connecting distances of equivalent time

• Travel times can be measured based on walking, biking, or driving
• Isochrone maps help measure the effect of being close to something, for instance:
  • What is the likelihood a member visits a specific facility given they are within 20 minutes of it?
  • Are members that are within 10 minutes of an ER more likely to have an ER visit that those that are more than 20 minutes away?
• Code to create the map is saved here on GitHub.

What we’ll create:
• Map of 5, 10, and 20 minute drivetimes of each NJ emergency room

nj_drivetime_union_simple.html
## Data Requirements

<table>
<thead>
<tr>
<th>Points of Interest</th>
<th>Definition</th>
<th>Sample Project Input</th>
<th>Data Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latitude and Longitude for the center of each isochrone</td>
<td>Facility Name</td>
<td>Latitude</td>
<td>Longitude</td>
</tr>
<tr>
<td></td>
<td>ER 1</td>
<td>39.3584</td>
<td>-74.4344</td>
</tr>
<tr>
<td></td>
<td>ER 2</td>
<td>39.4776</td>
<td>-74.5479</td>
</tr>
<tr>
<td></td>
<td>ER 3</td>
<td>39.3144</td>
<td>-74.5923</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>ER 73</td>
<td>40.7012</td>
<td>-75.1782</td>
</tr>
</tbody>
</table>
# Package Requirements

<table>
<thead>
<tr>
<th>Packages</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Folium</td>
<td>Leverages the JavaScript library Leaflet through python</td>
</tr>
<tr>
<td>Pandas</td>
<td>Easy to use data structures and data analysis tools</td>
</tr>
<tr>
<td>geopandas</td>
<td>Makes it easier to work with geospatial data in python</td>
</tr>
<tr>
<td>osmnx</td>
<td>Retrieve, model, analyze, and visualize OpenStreetMap street networks</td>
</tr>
<tr>
<td>networkx</td>
<td>Create, manipulate, and study the structure, dynamics, and function of complex networks</td>
</tr>
<tr>
<td>shapely</td>
<td>Manipulation and analysis of geometric objects</td>
</tr>
<tr>
<td>descartes</td>
<td>Precision plotting of polygons</td>
</tr>
<tr>
<td>matplotlib</td>
<td>Creating static, animated, and interactive visualizations in python</td>
</tr>
</tbody>
</table>
The code I use to make isochrones follows osmnx example notebook #13. Here is an overview of how the code works:

- Set parameters
  - Define type of travel (drive, bike, walk)
  - Define trip times in minutes
  - Define speed in km/h
- Change retrieval of street networks from place to point and define distance around point (in meters) to retrieve
- Use networkx to create a subgraph of the street network based on the use of a convex hull or buffering
- Use the buffering method, which is described in detail here.
- Add code to output each isochrone as a GeoJSON file with EPSG:26918

Each run of “Make Isochrone” outputs a GeoJSON in the format of “ER_i.json”

- Individual isochrones can be used to check if a member lives within x mins of a specific facility
- Each isochrone has 160,000K+ coordinate pairs which results in a file size of 5MB to 10MB

Individual isochrones can be plotted on one map, which is shown on the left, but polygons need to be “simplified” for a browser to handle the data

- Shapely simplify algorithm with parameter of 2 and unpreserved topology reduces coordinate pairs by 80%

Individual isochrones can be merged, so that drivetimes of equivalent distance become one shape as shown on the right.

- Merged isochrones can be used to check if a member lives within x mins of any facility