DBSCAN

In this notebook you will use GPU-accelerated DBSCAN to identify clusters of infected people.

Objectives

By the time you complete this notebook you will be able to:

- Use GPU-accelerated DBSCAN
- Use cuXfilter to visualize DBSCAN clusters

Imports

In [ ]:
```
import cudf
import cuml

import cuxfilter as cxf
```

Load Data

For this notebook, we again load a subset of our population data with only the columns we need. An `infected` column has been added to the data to indicate whether or not a person is known to be infected with our simulated virus.

In [ ]:
```
gdf = cudf.read_csv('./data/pop_2-04.csv', dtype=['float32', 'float32', '
print(gdf.dtypes)
gdf.shape
```

In [ ]:
```
gdf.head()
```

In [ ]:
```
gdf['infected'].value_counts()
```

DBSCAN Clustering

DBSCAN is another unsupervised clustering algorithm that is particularly effective when the number of clusters is not known up front and the clusters may have concave or other unusual shapes--a situation that often applies in geospatial analytics.

In this series of exercises you will use DBSCAN to identify clusters of infected people by location, which may help us identify groups becoming infected from common patient zeroes and assist in response planning.

Exercise: Make a DBSCAN Instance

Create a DBSCAN instance by using `cuml.DBSCAN`. Pass in the named argument `eps` (the maximum distance a point can be from the nearest point in a cluster to be considered possibly in that cluster) to be `5000`. Since the `northing` and `easting` values we created are measured in meters, this will allow us to identify clusters of infected people where individuals may be separated from the rest of the cluster by up to 5 kilometers.

In [ ]:

```

```

Solution

In [ ]:

```
%load solutions/dbscan_instance
```

Exercise: Identify Infected Clusters

Create a new dataframe from rows of the original dataframe where `infected` is `1` (true), and call it `infected_df` --be sure to reset the dataframe's index afterward. Use `dbscan.fit_predict` to perform clustering on the `northing` and `easting` columns of `infected_df`, and turn the resulting series into a new column in `infected_gdf` called "cluster". Finally, compute the number of clusters identified by DBSCAN.

In [ ]:

```

```

Solution

In [ ]:

```
%load solutions/identify_infected
```

Visualize the Clusters

Because we have the same column names as in the K-means example-- `easting`, `northing`, and `cluster` --we can use the same code to visualize the clusters.

Associate a Data Source with cuXfilter

In [ ]:

```
cxf_data = cxf.DataFrame.from_dataframe(infected_df)
```

Define Charts and Widgets

As in the K-means notebook, we have an existing integer column to use with multi-select: `cluster`.

In [ ]:

```
chart_width = 600
scatter_chart = cxf.charts.datashader.scatter(x='easting', y='northing',
                                              width=chart_width,
                                              height=int((gdf['northing']
                                                         (gdf['easting'].
                                                          chart_width))

cluster_widget = cxf.charts.panel_widgets.multi_select('cluster')
```

Create and Show the Dashboard

In [ ]:
```
dash = cxf_data.dashboard([scatter_chart, cluster_widget], theme=cxf.them
```

In [ ]:
```
scatter_chart.view()
```

In [ ]:
```
%%js
var host = window.location.host;
element.innerText = "'"+host+"'";
```

Set `my_url` in the next cell to the value just printed, making sure to include the quotes:

In [ ]:
```
my_url = # TODO: Set this value to the print out of the cell above, inclu
dash.show(my_url, port=8789)
```

... and you can run the next cell to generate a link to the dashboard:

In [ ]:
```
%%js
var host = window.location.host;
var url = 'http://'+host+'/lab/proxy/8789/';
element.innerHTML = '<a style="color:blue;" target="_blank" href='+url+'>
```

In [ ]:
```
dash.stop()
```

Please Restart the Kernel

In [ ]:
```
import IPython
app = IPython.Application.instance()
app.kernel.do_shutdown(True)
```

Next

In the next notebook, you will use GPU-accelerated logistic regression to estimate infection risk based on features of our population members.