Viewing TLE data

This Jupyter (https://jupyter.org) notebook explores and analyzes the raw Two-line Element Set (TLE) (https://en.wikipedia.org/wiki/Two-line_element_set) satellite tracking data provided for the 12/2020 VAULT technical scenario, showing what data is available and how it can be accessed and visualized from Python. The notebook also acts as a runnable application that can be put on a server to allow users to explore the raw data interactively.

In [15]:
```
import time, datetime, calendar
import pandas as pd
import numpy as np
import colorcet as cc
import panel as pn
import datetime as dt
import holoviews as hv
from holoviews.operation.datashader import rasterize, dynspread
import skyfield
from skyfield.api import EarthSatellite, load
from skyfield.framelib import itrs
hv.extension('bokeh')
```

First, let's load some 2017 TLE data using pandas (https://pandas.pydata.org):

In [2]:
```
tle = pd.read_csv('data/satellite data/Cleaned TLE/tle2017.csv')
tle.head(2)
```

Out[2]:

	Unnamed: 0	norad_id	epoch_year	epoch_day	tle
0	0	41857	2016	366.944278	None\n1 41857U 16068A 16366.94427814 -.00000...
1	1	41909	2016	366.950892	None\n1 41909U 16083C 16366.95089246 -.00000...

We'll need to work with the `tle` field, where each entry is a three-line string, with the 2nd and 3rd lines containing the data about the satellite position, heading, etc.

In [3]:
```
tle.iloc[0]['tle']
```

Out[3]:
```
'None\n1 41857U 16068A   16366.94427814 -.00000043  00000-0  00000+0 0  1682\n2 41857  98.5102   3.1811 0012704
113.7213 246.5323 14.32700070  7161\n'
```

Let's convert that to a list of string pairs (2nd and 3rd lines):

In [4]:
```
%%time
new_lines = [el.replace('None\n', '').split('\n')[:2] for el in tle['tle']] # Splitting the file
```

```
CPU times: user 2.43 s, sys: 216 ms, total: 2.65 s
Wall time: 2.64 s
```

In [5]:
```
new_lines[:5]
```

Out[5]:
```
[['1 41857U 16068A   16366.94427814 -.00000043  00000-0  00000+0 0  1682',
  '2 41857  98.5102   3.1811 0012704 113.7213 246.5323 14.32700070  7161'],
 ['1 41909U 16083C   16366.95089246 -.00000722  22793-5  00000+0 0   222',
  '2 41909  97.5945  78.7102 0220259 352.7158   7.6030 15.68490028   582'],
 ['1 41844U 16066D   17001.00057939 -.00000070  00000-0  00000+0 0  1823',
  '2 41844  97.4050  10.7457 0018148 104.4001 338.9419 15.21567598  7917'],
 ['1 25544U 98067A   16366.94039976  .00001110  00000-0  24188-4 0  4578',
  '2 25544  51.6423 152.0418 0007056  39.1392 105.7402 15.53963830 35740'],
 ['1 41857U 16068A   17001.08394256 -.00000043  00000-0  00000+0 0  1698',
  '2 41857  98.5100   3.3185 0012675 113.2709 246.9829 14.32700431  7185']]
```

Given a TLE string pair, we can use the skyfield (https://rhodesmill.org/skyfield) library to calculate the longitude and latitude location of the satellite (projected down on the earth's surface) at the time of the TLE tracking record entry. Note that the calculation takes some time, for a large file.

In [6]:
```
def modulo_lon(val):
    return (val+180) % 360 - 180

def compute_lat_lon(line1, line2):
    """Get the Lat/Lon at the TLE epoch"""
    sat = EarthSatellite(line1, line2)
    lat, lon, _ = sat.at(sat.epoch).frame_latlon(itrs)
    return lat.degrees, lon.degrees

def lat_lon_from_lines(lines):
    lons, lats = [], []
    for line1, line2 in lines:
        lat, lon = compute_lat_lon(line1,line2)
        if None not in [lon, lat]:
            lons.append(modulo_lon(lon))
            lats.append(lat)
    return np.array(lats), np.array(lons)
```

In [7]:
```
%%time
lines=20000
lines=None # comment out for quick run
lats, lons = lat_lon_from_lines(new_lines[:lines])
```

```
CPU times: user 13min 40s, sys: 365 ms, total: 13min 41s
Wall time: 13min 41s
```

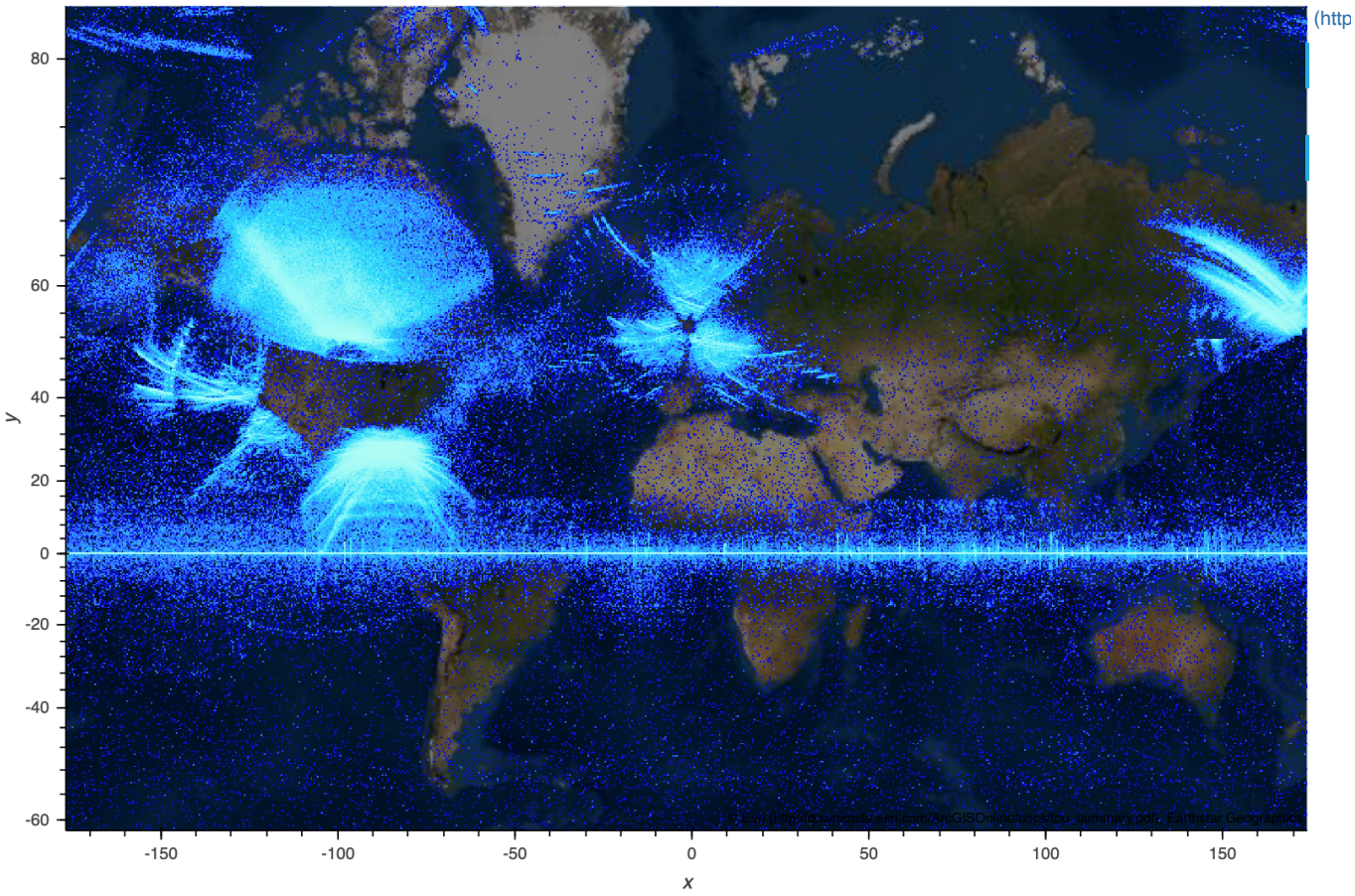
Once we have the lat,lon data, we can convert it to easting,northing for plotting with Datashader (https://datashader.org) and HoloViews (https://holoviews.org) on a web tile map:

In [8]:
```
%%time
eastings, northings = hv.util.transform.lon_lat_to_easting_northing(lons, lats)
```

```
CPU times: user 82.1 ms, sys: 2 µs, total: 82.1 ms
Wall time: 81.4 ms
```

In [16]:
```
tiles  = hv.element.tiles.ESRI().opts(alpha=0.5, bgcolor='black')
points = rasterize(hv.Points((eastings,northings)))
points = points.opts(width=900, height=600, cmap=cc.kbc[64:], cnorm='eq_hist')
tiles * dynspread(points)
```

Out[16]:



We can see lots of interesting structure in the locations of the satellites at the epoch time of each record. To help make sense of it, let's overlay data about the locations of dedicated and collateral tracking radar stations (https://en.wikipedia.org/wiki/Solid_State_Phased_Array_Radar_System) in the Space Surveillance Network (https://en.wikipedia.org/wiki/File:Space_Surveillance_Network.jpg):

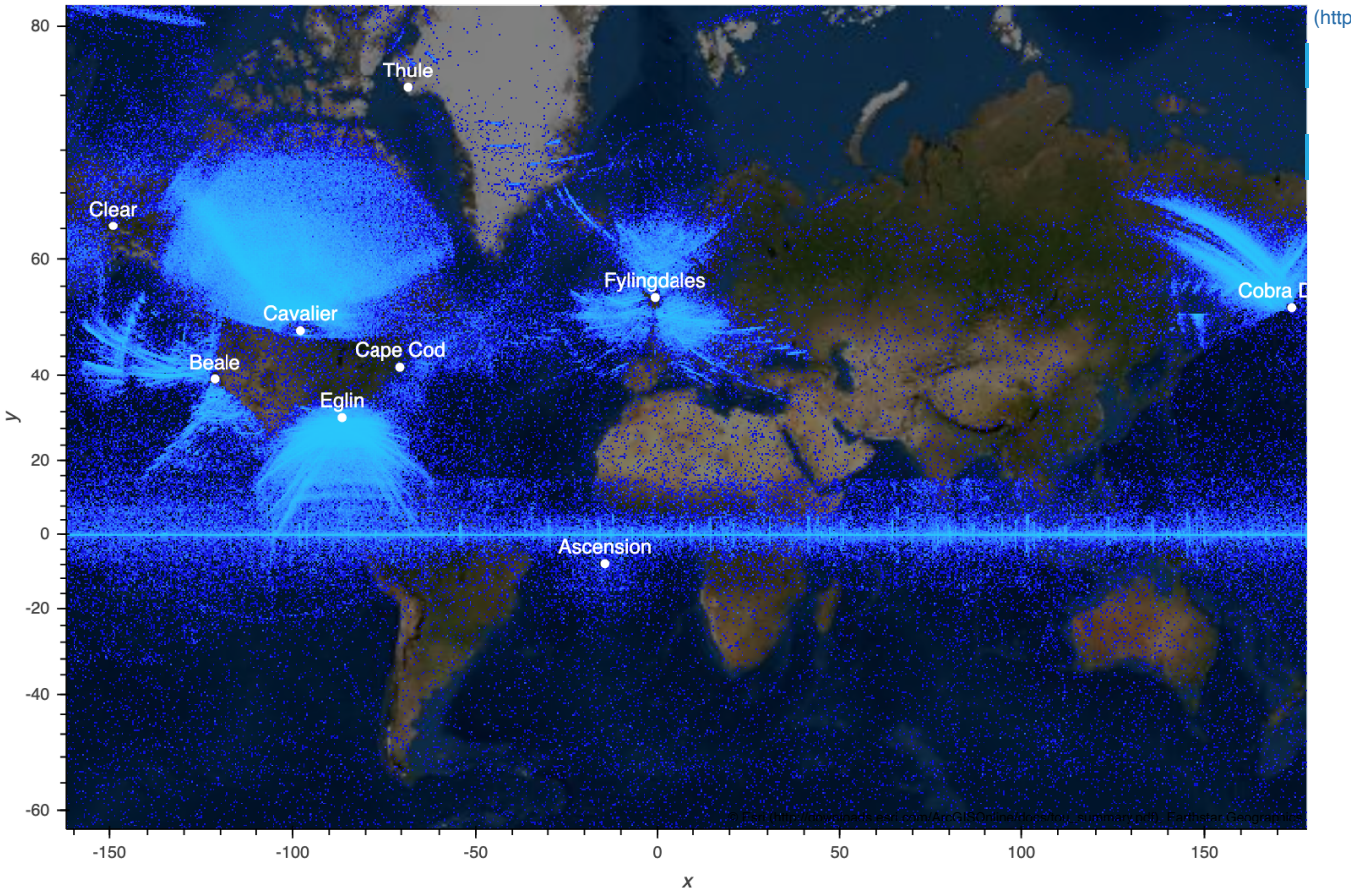
In [10]:
```
stations = {"Beale":       (39.136111, -121.436389),
            "Cape Cod":    (41.753333,  -70.538611),
            "Clear":       (64.290556, -149.187222),
            "Thule":       (76.567850,  -68.284214),
            "Fylingdales": (54.358889,   -0.669722),
            "Eglin":       (30.489444,  -86.542222),
            "Cavalier":    (48.731944,  -97.904444),
            "Ascension":   (-7.969444,  -14.393889),
            "Cobra Dane":  (52.712222,  174.113611)}

stations = {k:hv.util.transform.lon_lat_to_easting_northing(v[1],v[0]) for k,v in stations.items()}
```

In [11]:
```
radar = hv.Points(list(stations.values())).opts(color='white', size=5)
names = hv.Labels({('x', 'y'): list(stations.values()),
                   'text': list(stations.keys())}, ['x', 'y'], 'text')
names = names.opts(text_font_size='10pt', yoffset=0.5e6, text_color="white")
```

In [17]:
```
tiles  = hv.element.tiles.ESRI().opts(alpha=0.5, bgcolor='black')
points = rasterize(hv.Points((eastings,northings))).opts(width=900, height=600, cmap=cc.kbc[64:200], cnorm='eq_hist
tiles * dynspread(points) * radar * names
```

Out[17]:



Now we can see that most of the records (at the epoch) are from when the satellites were close to one of the radar-based monitoring stations indicated, apart from those along the equator (presumably in geostationary orbit) and a few other widely distributed observations.

Finally, we'll package up the above plot into a servable app using Panel (https://panel.holoviz.org):

In [13]:
```
pn.Column("# Raw TLE data",
          "TLE records plotted at the epoch time in geo coordinates, "
          "with Space Surveillance Network radar stations overlaid",
          tiles * points * radar * names).servable();
```

/