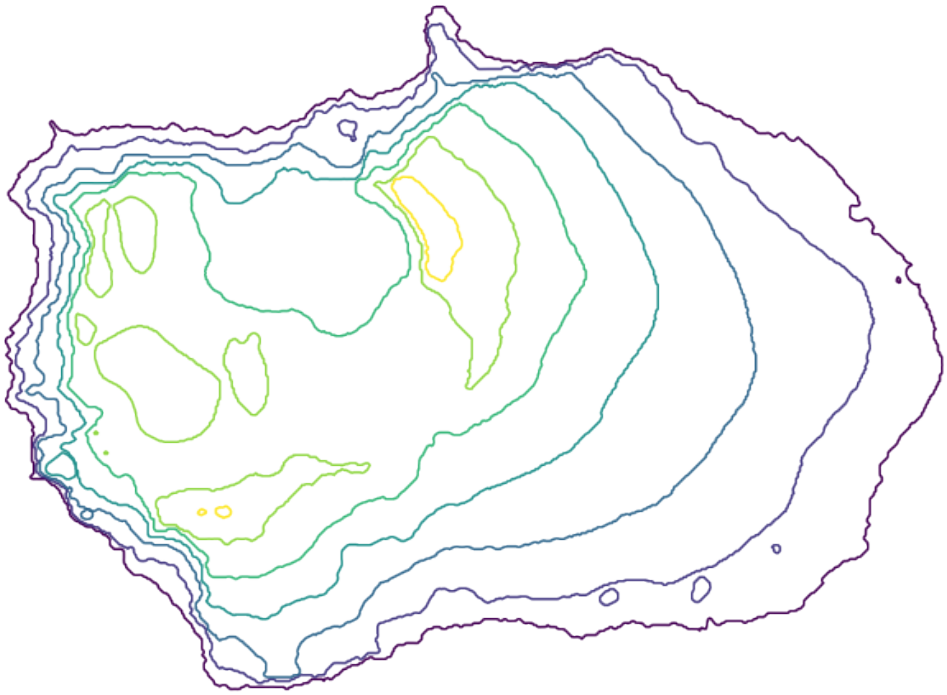
Vi legger til en dimensjon – analyse av kartinformasjon i Python

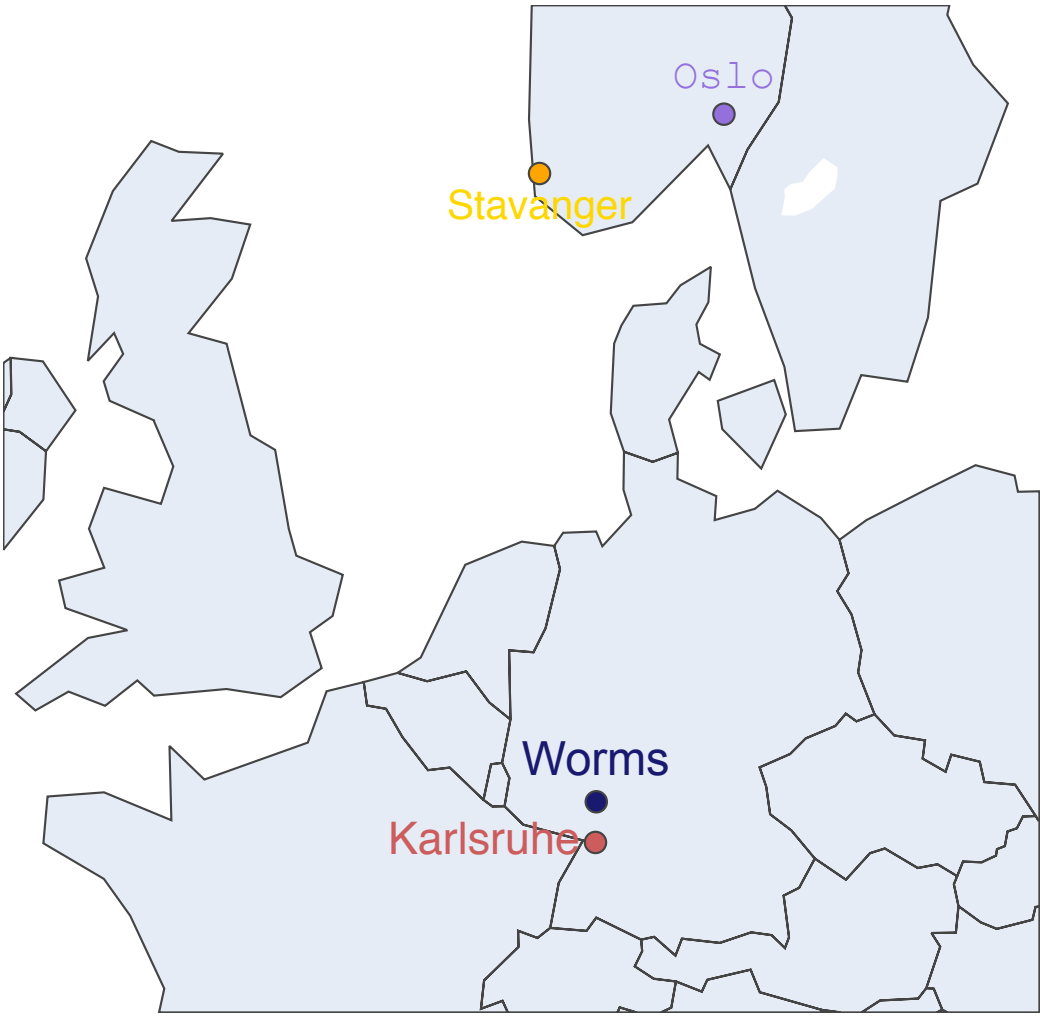
Foredrag på regionssamling Bouvet Rogaland, høsten 2019
Ture Friese, avdeling Skynety



Om meg

- hjemmebyen er Worms i Tyskland
- utdannelse: Sivilingeniør industriell økonomi og endringsledelse
- 10 år som systemutvikler og forretningsutvikler i finans i Oslo
- jobber som data scientist i avdeling Skynet siden juni 2019

- kontakt: ture.friese@bouvet.no (mailto:ture.friese@bouvet.no), mobil 99230426



Vi begynner med en titt på et typisk BI verktøy

Google BigQuery GeoViz tool:
[https://bigquerygeoviz.appspot.com (https://bigquerygeoviz.appspot.com)](https://bigquerygeoviz.appspot.com)

Query that we want to visualize:

In [5]:
```
%%bigquery stations

SELECT st_geogPoint(longitude, latitude) as WKT, num_bikes_available
FROM `bigquery-public-data.new_york.citibike_stations`
WHERE num_bikes_available > 30
```

Vi legger til

- stil **fillColor**: #0000FF
- stil **fillOpacity**: .5
- stil **circleRadius**: data-driven, field num_bikes_available, 30-60, 5-200

Hentet fra artikkel...
Getting started with BigQuery GIS
https://cloud.google.com/bigquery/docs/gis-getting-started
(https://cloud.google.com/bigquery/docs/gis-getting-started)

Observasjoner:

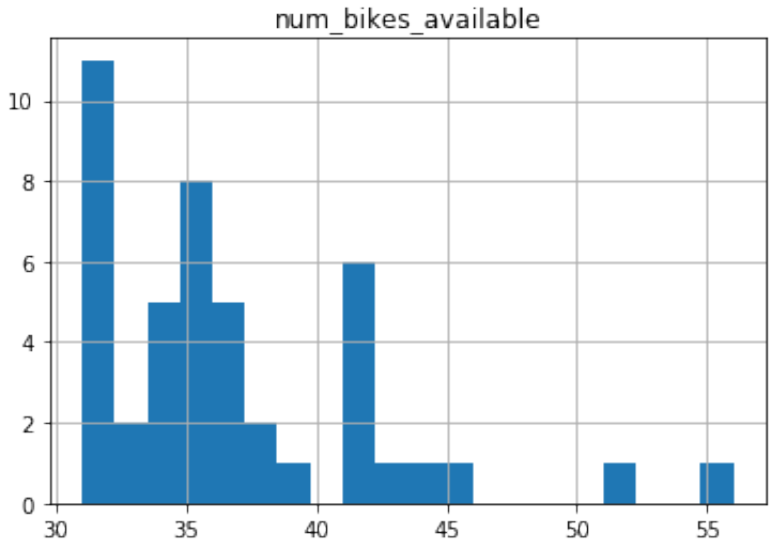
- begrenset i funksjonalitet
- point&click (and prey...)

Hva hvis du vil gjøre mer?

- tegne fyllingsgrad som stolpe?
- legge til flere lag med data, befolkningstetthet, kjøreruter, trafikk
- bruke geografi som dimensjon i maskinlæring?

Da kan vi gå over til python...

In [39]:
```
stations.hist(bins=20);
```

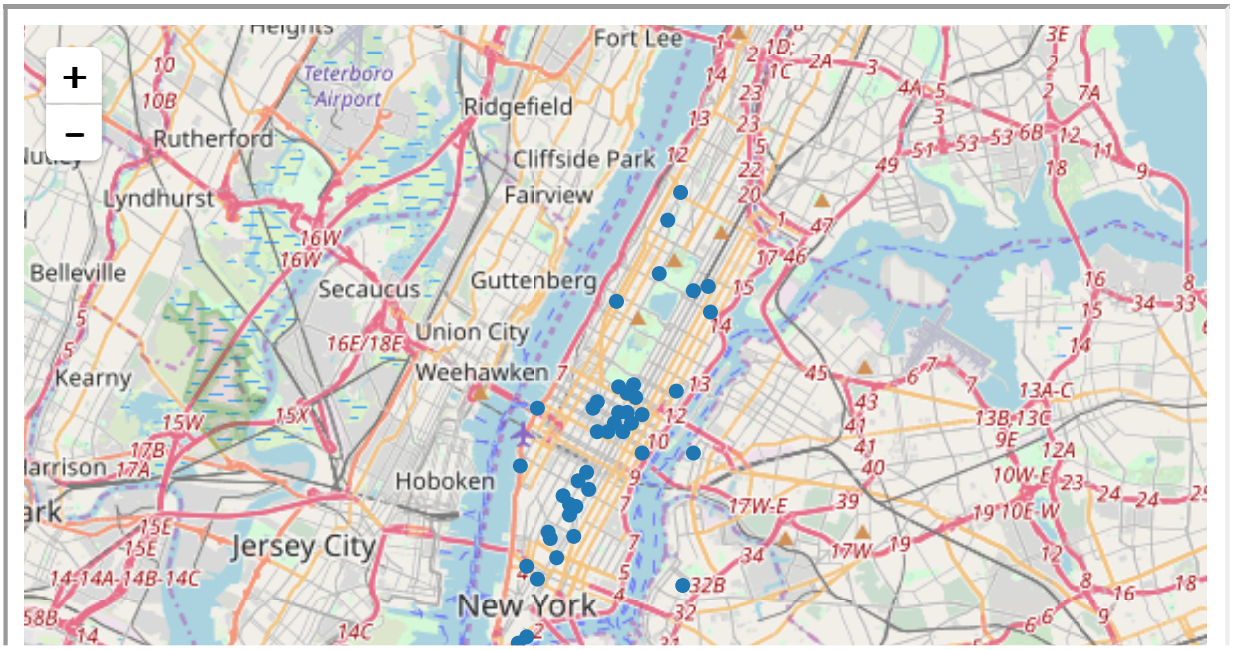


La oss plotte data på kart.

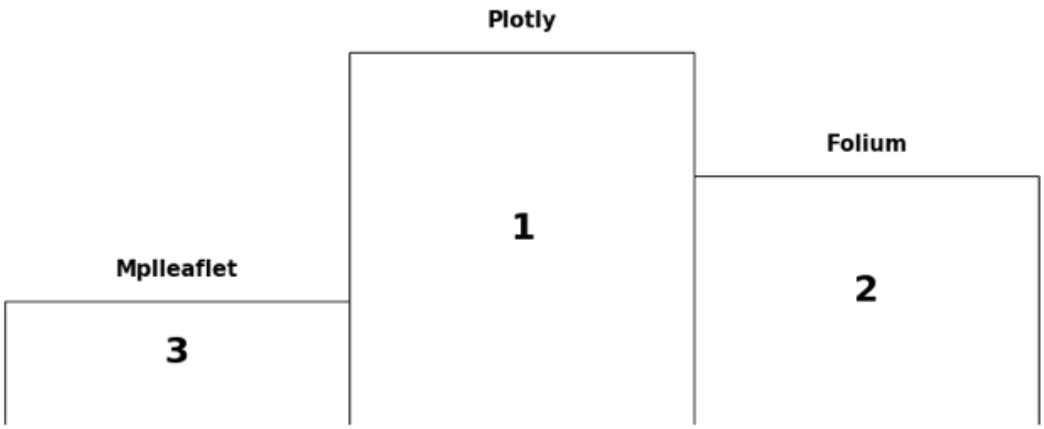
In [6]:
```
stations_gdf = gpd.GeoDataFrame(data=stations, geometry=stations.WKT.a
stations_gdf.plot(figsize=[13,7])

mplleaflet.display()
```

Out[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f90ac675f10>
```

Out[6]:



3 Python-bilbioteker for plotting av kart



- **Mplleaflet** er raskt og enkelt
  mer info på hjemmesiden:
  https://github.com/jwass/mplleaflet (https://github.com/jwass/mplleaflet)

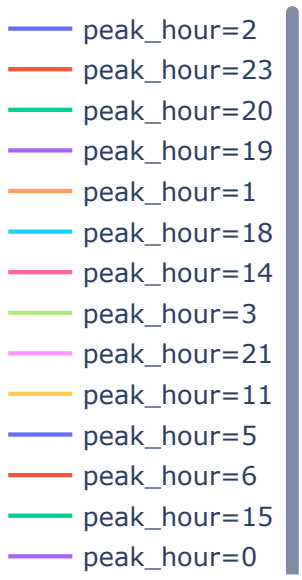
- **Folium** er bra for heatmaps, colorpleths og interaktive kart
  github-side:
  https://github.com/python-visualization/folium (https://github.com/python-visualization/folium)
  I Alexis Cook "*Interactive Maps*" tutorial på Kaggle.com finner vi noen eksempler for heatmap og colorpleth:
  https://www.kaggle.com/alexisbcook/interactive-maps (https://www.kaggle.com/alexisbcook/interactive-maps)
  Her finner man og kildekoden og forklaringen hvordan man gjør.

- **Plotly**, omfangrik, relativt enkelt å programmere, mest oppdatert
  Hjemmeside:
  https://plot.ly/python/plotly-express/ (https://plot.ly/python/plotly-express/)

Ikke bruk **Basemap** 🙁

Et eksemple med Plotly:

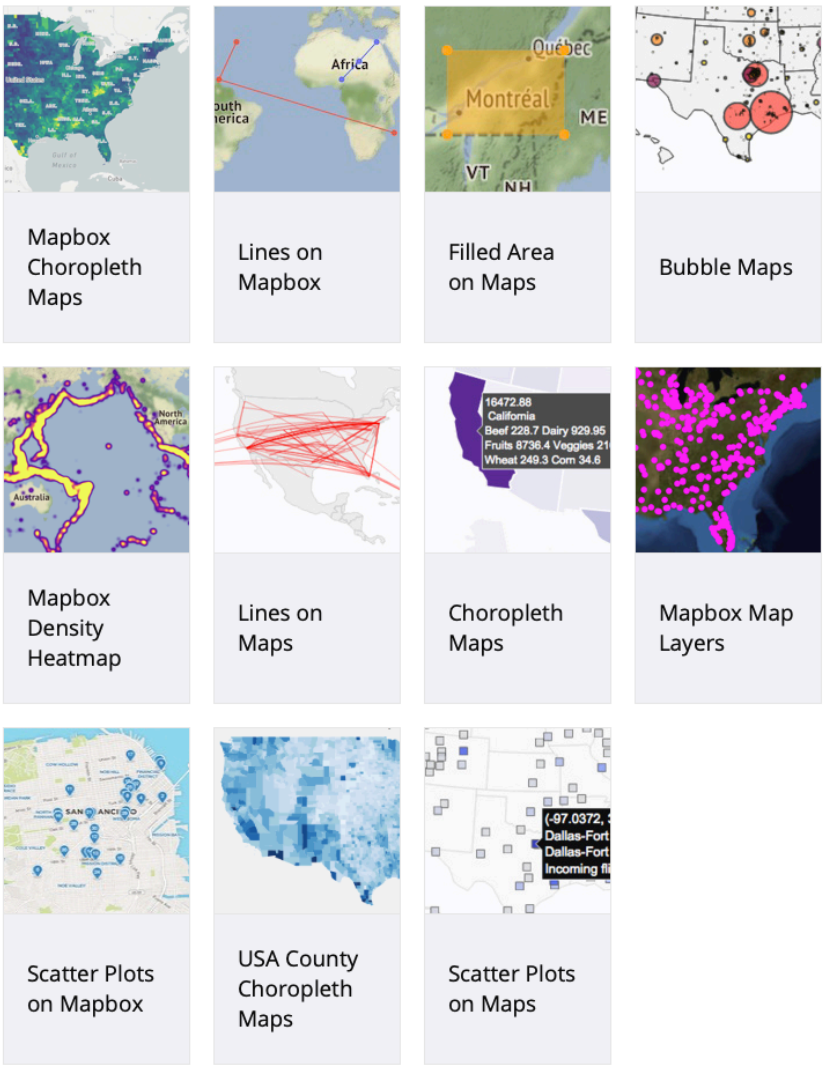
In [7]:
```
import plotly.express as px
px.set_mapbox_access_token('pk.eyJ1IjoidHVyZS1ib3V2ZXQiLCJhIjoiY2p6OGp

carshare = px.data.carshare()
fig = px.line_mapbox(carshare, lat="centroid_lat", lon="centroid_lon",
fig.show()
```



(hentet fra: https://plot.ly/python/plotly-express/#scatter-and-line-plots (https://plot.ly/python/plotly-express/#scatter-and-line-plots))

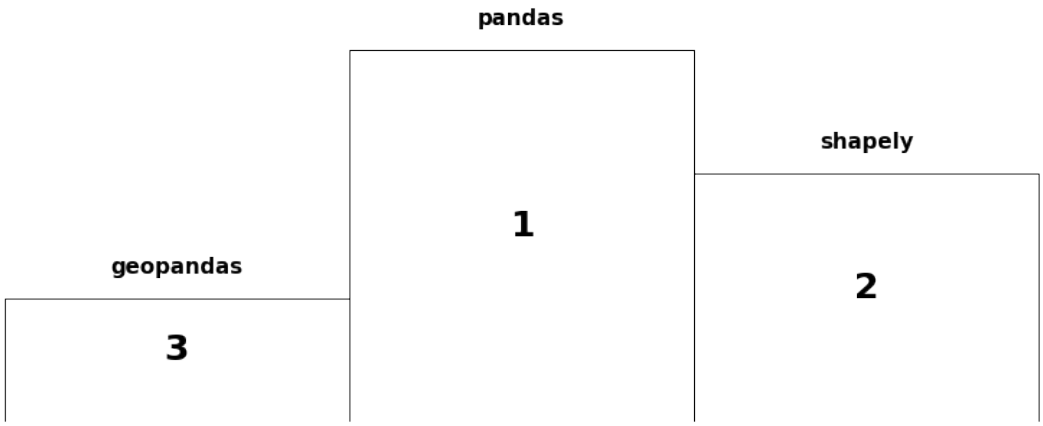
Mer kart eksempler på plotly hjemmeside:

Out[154]:



Ref. https://plot.ly/python/maps/#maps (https://plot.ly/python/maps/#maps)

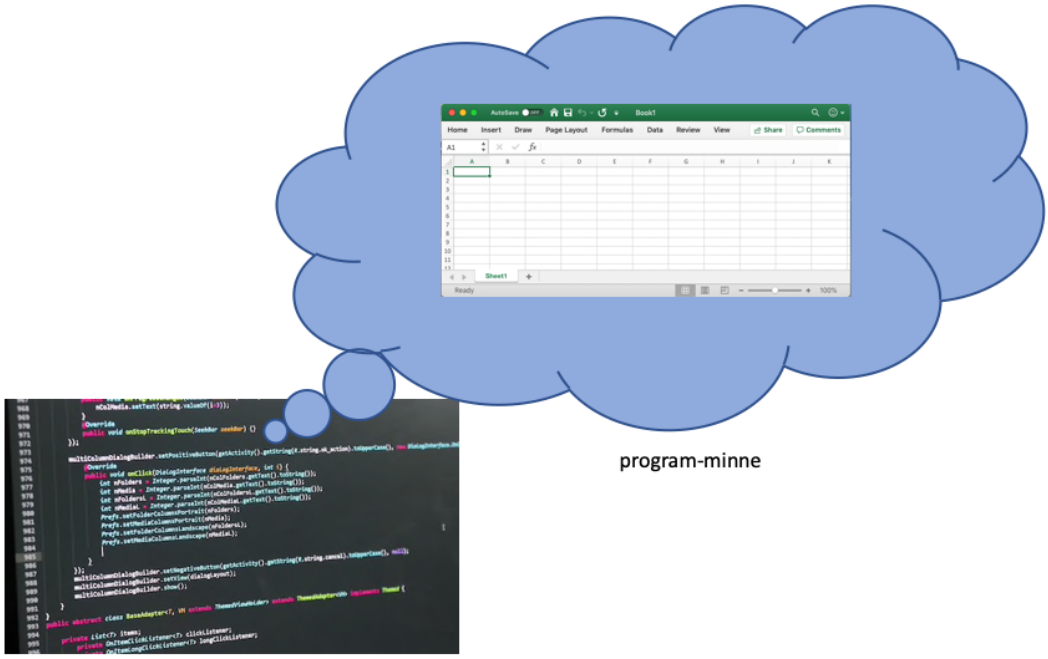
De viktigste biblioteker for kartdata i Python



Pandas – hjørnesteins bibliotek for Data Science

- DataFrame er den sentrale datastrukturer
- Tenk Excel i program-minne !
- ref. [https://pandas.pydata.org (https://pandas.pydata.org)](https://pandas.pydata.org)

Out[80]:



Pandas – sveisterkniv for tabell-data i Python

Out[119]:

In [110]:
```
print('Methods of pandas:\n\n', [entry for entry in dir(pd) if entry[(
print('\n\nMethods of DataFrame:\n\n', [entry for entry in dir(pd.Data
```

```
Methods of pandas:

 ['api', 'array', 'arrays', 'bdate_range', 'compat', 'concat', 'core
', 'crosstab', 'cut', 'date_range', 'datetime', 'describe_option', '
errors', 'eval', 'factorize', 'get_dummies', 'get_option', 'infer_fr
eq', 'interval_range', 'io', 'isna', 'isnull', 'lreshape', 'melt', '
merge', 'merge_asof', 'merge_ordered', 'notna', 'notnull', 'np', 'of
fsets', 'option_context', 'options', 'pandas', 'period_range', 'pivo
t', 'pivot_table', 'plotting', 'qcut', 'read_clipboard', 'read_csv',
'read_excel', 'read_feather', 'read_fwf', 'read_gbq', 'read_hdf', 'r
ead_html', 'read_json', 'read_msgpack', 'read_parquet', 'read_pickle
', 'read_sas', 'read_spss', 'read_sql', 'read_sql_query', 'read_sql_
table', 'read_stata', 'read_table', 'reset_option', 'set_eng_float_f
ormat', 'set_option', 'show_versions', 'test', 'testing', 'timedelta
_range', 'to_datetime', 'to_msgpack', 'to_numeric', 'to_pickle', 'to
_timedelta', 'tseries', 'unique', 'util', 'value_counts', 'wide_to_l
ong']


Methods of DataFrame:

 ['abs', 'add', 'add_prefix', 'add_suffix', 'agg', 'aggregate', 'ali
gn', 'all', 'any', 'append', 'apply', 'applymap', 'as_blocks', 'as_m
atrix', 'asfreq', 'asof', 'assign', 'astype', 'at', 'at_time', 'axes
', 'between_time', 'bfill', 'blocks', 'bool', 'boxplot', 'clip', 'cl
ip_lower', 'clip_upper', 'columns', 'combine', 'combine_first', 'com
pound', 'copy', 'corr', 'corrwith', 'count', 'cov', 'cummax', 'cummi
n', 'cumprod', 'cumsum', 'describe', 'diff', 'div', 'divide', 'dot',
```

```
'drop', 'drop_duplicates', 'droplevel', 'dropna', 'dtypes', 'duplica
ted', 'empty', 'eq', 'equals', 'eval', 'ewm', 'expanding', 'explode'
, 'ffill', 'fillna', 'filter', 'first', 'first_valid_index', 'floord
iv', 'from_dict', 'from_items', 'from_records', 'ftypes', 'ge', 'get
', 'get_dtype_counts', 'get_ftype_counts', 'get_value', 'get_values'
, 'groupby', 'gt', 'head', 'hist', 'iat', 'idxmax', 'idxmin', 'iloc'
, 'index', 'infer_objects', 'info', 'insert', 'interpolate', 'is_cop
y', 'isin', 'isna', 'isnull', 'items', 'iteritems', 'iterrows', 'ite
rtuples', 'ix', 'join', 'keys', 'kurt', 'kurtosis', 'last', 'last_va
lid_index', 'le', 'loc', 'lookup', 'lt', 'mad', 'mask', 'max', 'mean
', 'median', 'melt', 'memory_usage', 'merge', 'min', 'mod', 'mode',
'mul', 'multiply', 'ndim', 'ne', 'nlargest', 'notna', 'notnull', 'ns
mallest', 'nunique', 'pct_change', 'pipe', 'pivot', 'pivot_table', '
plot', 'pop', 'pow', 'prod', 'product', 'quantile', 'query', 'radd',
'rank', 'rdiv', 'reindex', 'reindex_like', 'rename', 'rename_axis',
'reorder_levels', 'replace', 'resample', 'reset_index', 'rfloordiv',
'rmod', 'rmul', 'rolling', 'round', 'rpow', 'rsub', 'rtruediv', 'sam
ple', 'select_dtypes', 'sem', 'set_axis', 'set_geometry', 'set_index
', 'set_value', 'shape', 'shift', 'size', 'skew', 'slice_shift', 'so
rt_index', 'sort_values', 'sparse', 'squeeze', 'stack', 'std', 'styl
e', 'sub', 'subtract', 'sum', 'swapaxes', 'swaplevel', 'tail', 'take
', 'to_clipboard', 'to_csv', 'to_dense', 'to_dict', 'to_excel', 'to_
feather', 'to_gbq', 'to_hdf', 'to_html', 'to_json', 'to_latex', 'to_
msgpack', 'to_numpy', 'to_parquet', 'to_period', 'to_pickle', 'to_re
cords', 'to_sparse', 'to_sql', 'to_stata', 'to_string', 'to_timestam
p', 'to_xarray', 'transform', 'transpose', 'truediv', 'truncate', 't
shift', 'tz_convert', 'tz_localize', 'unstack', 'update', 'values',
'var', 'where', 'xs']
```

Noen kode eksempler:

In [9]:
```
type(stations)

stations.head()

stations.shape
stations.columns.values.tolist()
```

Out[9]: pandas.core.frame.DataFrame

Out[9]:

	WKT	num_bikes_available	geometry
0	POINT(-73.95600096 40.71774592)	31	POINT (-73.95600 40.71775)
1	POINT(-73.94755757 40.7903051)	35	POINT (-73.94756 40.79031)
2	POINT(-73.97109243 40.76350532)	33	POINT (-73.97109 40.76351)
3	POINT(-73.99063168 40.6867443)	35	POINT (-73.99063 40.68674)
4	POINT(-73.956461 40.813358)	39	POINT (-73.95646 40.81336)

Out[9]: (45, 3)

Out[9]: ['WKT', 'num_bikes_available', 'geometry']

Shapely

- Jobbe med geometri-objekter som Point, LineString og Poligon
- Les mer her: https://shapely.readthedocs.io/en/stable/manual.html (https://shapely.readthedocs.io/en/stable/manual.html)

In [10]:
```
from shapely.geometry import Polygon, Point

station4 = Point(-73.956461, 40.813358)
station5 = Point(-73.998102, 40.729170)
lower_manhatten = Polygon([(-73.990533, 40.706828), (-74.024183, 40.69

station4.within(lower_manhatten)

station5.within(lower_manhatten)
```

Out[10]: False

Out[10]: True

Stemmer det? La oss se på kart.

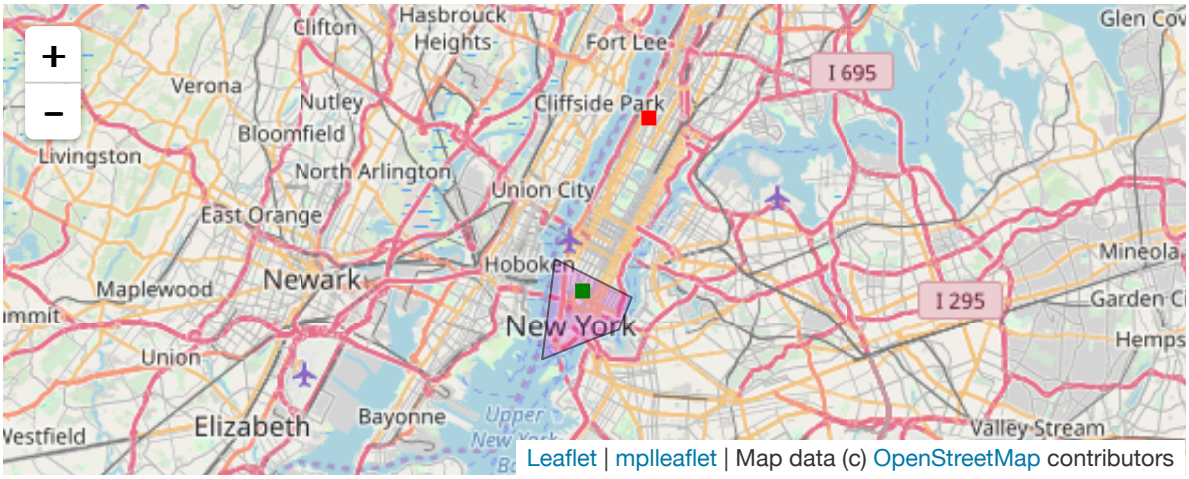
In [11]:
```
plt.figure().gca().add_patch(PolygonPatch(lower_manhatten, fc='#cc00cc
plt.plot(station4.coords[0][0], station4.coords[0][1], 'rs')
plt.plot(station5.coords[0][0], station5.coords[0][1], 'gs')

mplleaflet.display()
```

Out[11]: <matplotlib.patches.PathPatch at 0x7f90ad6ee550>

Out[11]: [<matplotlib.lines.Line2D at 0x7f90ad6fb700>]

Out[11]: [<matplotlib.lines.Line2D at 0x7f90ad6fb5b0>]

Out[11]:



Geopandas

- utvidelse av Pandas DataFrame med en geografisk dimensjon, dvs. kolonne
- tillater shaply operasjoner på hele DataFrame
- mer info:
  http://geopandas.org (http://geopandas.org)

In [44]:
```
%%bigquery boro

SELECT *
FROM `bigquery-public-data.new_york_subway.geo_nyc_borough_boundaries`
```

Noen kodeeksepler:

In [45]:
```
import geopandas as gpd

boroughs = gpd.GeoDataFrame(data=boro, geometry=boro.borough_geom.appl

type(boroughs)

boroughs
```

Out[45]:
```
geopandas.geodataframe.GeoDataFrame
```

Out[45]:

	borough_code	borough_name	borough_area	borough_len	borough_
0	2	Bronx	1.186612e+09	462958.188213	MULTIPOLYGON(((-73.88885148 40.7987063
1	5	Staten Island	1.623756e+09	325960.634597	MULTIPOLYGON(((-74.05314036 40.5777027
2	4	Queens	3.045885e+09	904390.137335	MULTIPOLYGON(((-73.80997059 40.6000675
3	1	Manhattan	6.366027e+08	361212.479734	MULTIPOLYGON(((-73.92133752 40.8008521
4	3	Brooklyn	1.937593e+09	738745.835869	MULTIPOLYGON(((-73.91990064 40.5996005

Inner join, left outer join, cross join, natural join... har du hørt om sjoin?

Spacial join med geopandas:

In [46]:
```
result = gpd.sjoin(stations_gdf, boroughs, how='inner', op='intersects

result
```

Out[46]:

	WKT	num_bikes_available	geometry	index_right	borough_code	borough
0	POINT(-73.95600096 40.71774592)	31	POINT (-73.95600 40.71775)	4	3	B
3	POINT(-73.99063168 40.6867443)	35	POINT (-73.99063 40.68674)	4	3	B
7	POINT(-73.98631746 40.69236178)	37	POINT (-73.98632 40.69236)	4	3	B
40	POINT(-73.981013 40.689888)	31	POINT (-73.98101 40.68989)	4	3	B
1	POINT(-73.94755757 40.7903051)	35	POINT (-73.94756 40.79031)	3	1	Ma
2	POINT(-73.97109243	33	POINT (-73.97109	3	1	Ma

In [136]:
```
boro
```

Out[136]:

	borough_code	borough_name	borough_area	borough_len	borough_
0	2	Bronx	1.186612e+09	462958.188213	MULTIPOLYGON(((-73.88885148 40.798706
1	5	Staten Island	1.623756e+09	325960.634597	MULTIPOLYGON(((-74.05314036 40.577702
2	4	Queens	3.045885e+09	904390.137335	MULTIPOLYGON(((-73.80997059 40.600067
3	1	Manhattan	6.366027e+08	361212.479734	MULTIPOLYGON(((-73.92133752 40.800852
4	3	Brooklyn	1.937593e+09	738745.835869	MULTIPOLYGON(((-73.91990064 40.599600

GeoPython konferanse

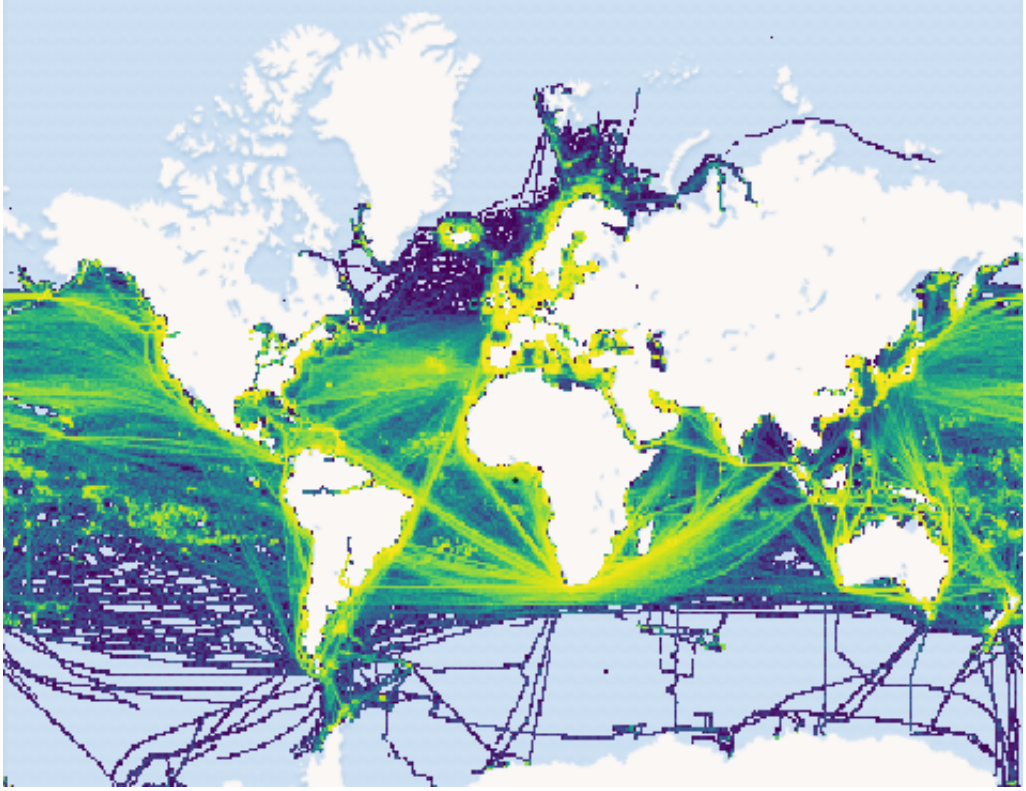
- årlig python konferanse i Basel/sveits
- hovedtema: geo-data, visulaisering machine learning
- ref. http://www.geopython.net (http://www.geopython.net)
- agenda: https://submit.geopython.net/geopython2019/talk/ (https://submit.geopython.net/geopython2019/talk/)

Out[155]:


2 fordrag fra GeoPython 2019:

- *PyViz for Mapping Global Shipping*
  summary: http://2019.geopython.net/geopython2019/talk/8WK7GT/ (http://2019.geopython.net/geopython2019/talk/8WK7GT/)
  code: https://github.com/UKHO/geopython2019/ (https://github.com/UKHO/geopython2019/)
  desverre ingen video fra foredraget

Out[177]:



- *Wikidata - a new source for geospatial data*
  https://k-nut.eu/static/Geopython-Wikidata.pdf (https://k-nut.eu/static/Geopython-Wikidata.pdf)
  -> kan brukes for geo-coding

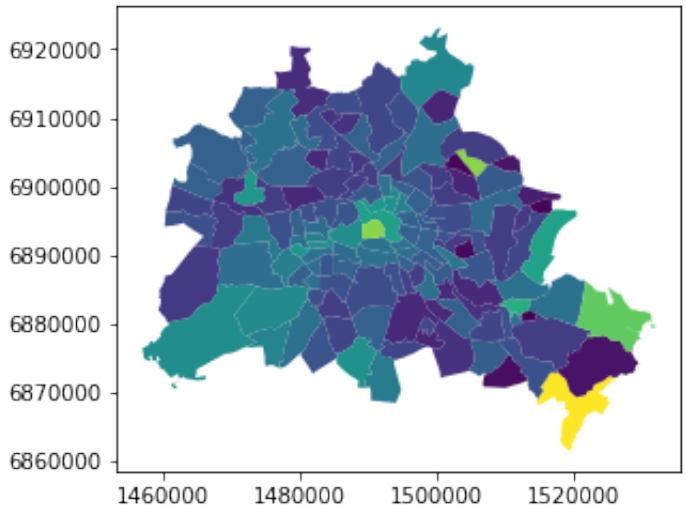
Out[12]:

```
SELECT ?swiss_city ?swiss_cityLabel
WHERE
{
  ?swiss_city wdt:P31 wd:Q54935504 .
  SERVICE wikibase:label { bd:serviceParam wikibase:langua
}
```

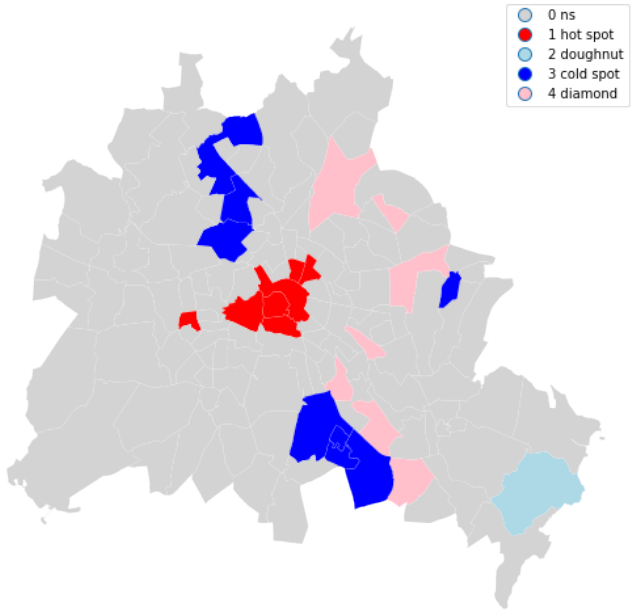
En fordrag å trekke fram fra GeoPython 2018:

- *Spatial Data Science with PyData*
  https://github.com/ljwolf/geopython (https://github.com/ljwolf/geopython)
  spesielt interessat hotspot/coldspot analyse som beskrevet i dette notebook:
  https://github.com/geopandas/scipy2018-geospatial-data/blob/master/06-exploratory-spatial-data-analysis.ipynb (https://github.com/geopandas/scipy2018-geospatial-data/blob/master/06-exploratory-spatial-data-analysis.ipynb)

Out[184]:



Out[200]:



Geobinning

- Rund av koordinatene slik at alle punkter faller på et raster
- Slå long/lat samme til en kolonne

Kondeeksemplet:

In [169]:

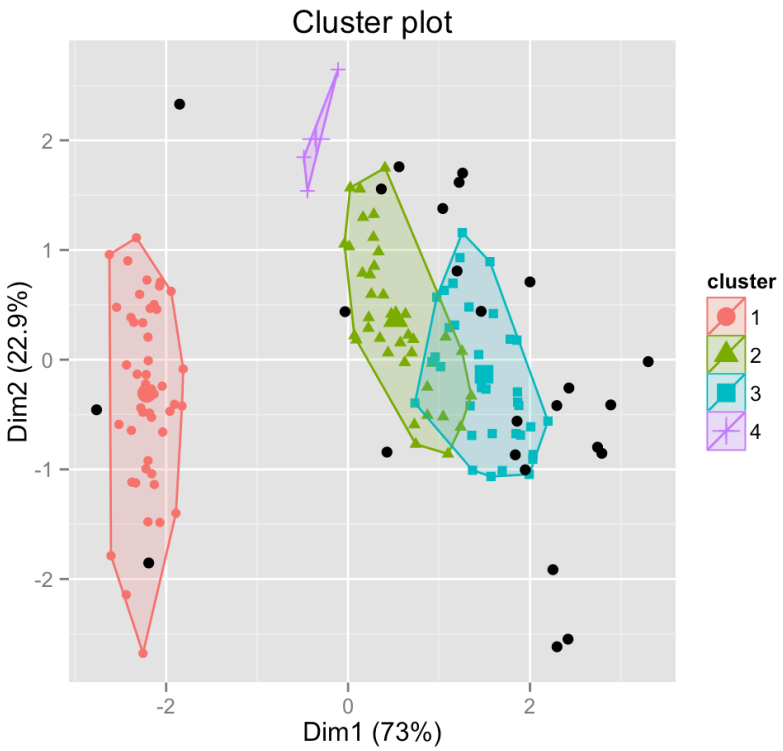
```
import pandas as pd

df = pd.DataFrame(columns=['lon','lat'])

df['geobin'] = (
                  round(df['lon'] / 9, 2) * 9 * 100 * 10000
                + round(df['lat'] / 4.5, 2) * 4.5 * 100

                ).astype(int)
```

Geo-clustering

- Grupperer punkter på kart i "cluster" med punkter som er nærliggende
- For clustering bruker man gjerne k-means og dbscan algorithmene fra skearn biblioteket
  Les mer her:
  https://scikit-learn.org/stable/modules/clustering.html (https://scikit-learn.org/stable/modules/clustering.html)

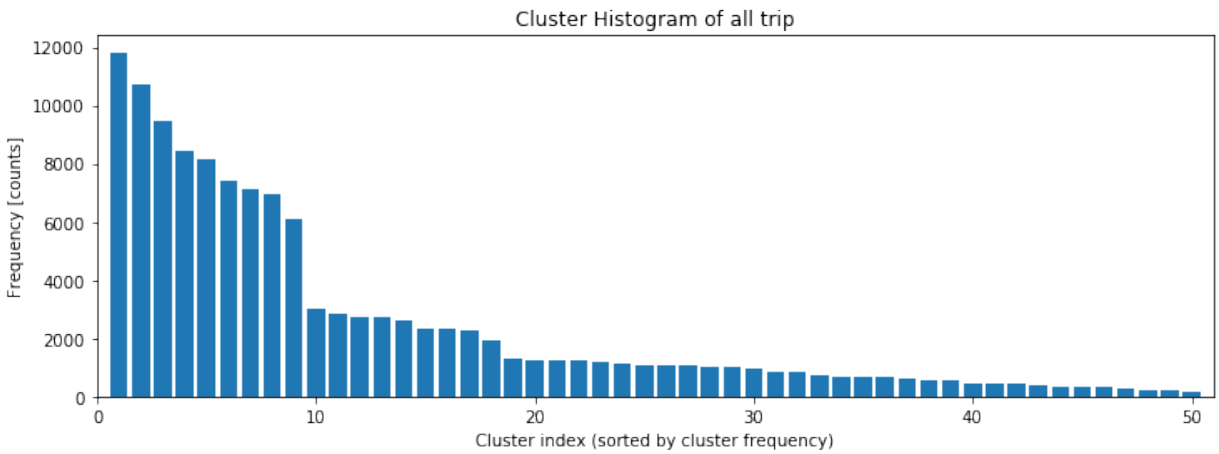
- God artikkel i forhold til **geo**-clustering:
  Clustering on New York City Bike Dataset
  https://chih-ling-hsu.github.io/2018/01/02/clustering-python (https://chih-ling-hsu.github.io/2018/01/02/clustering-python)

Out[124]:



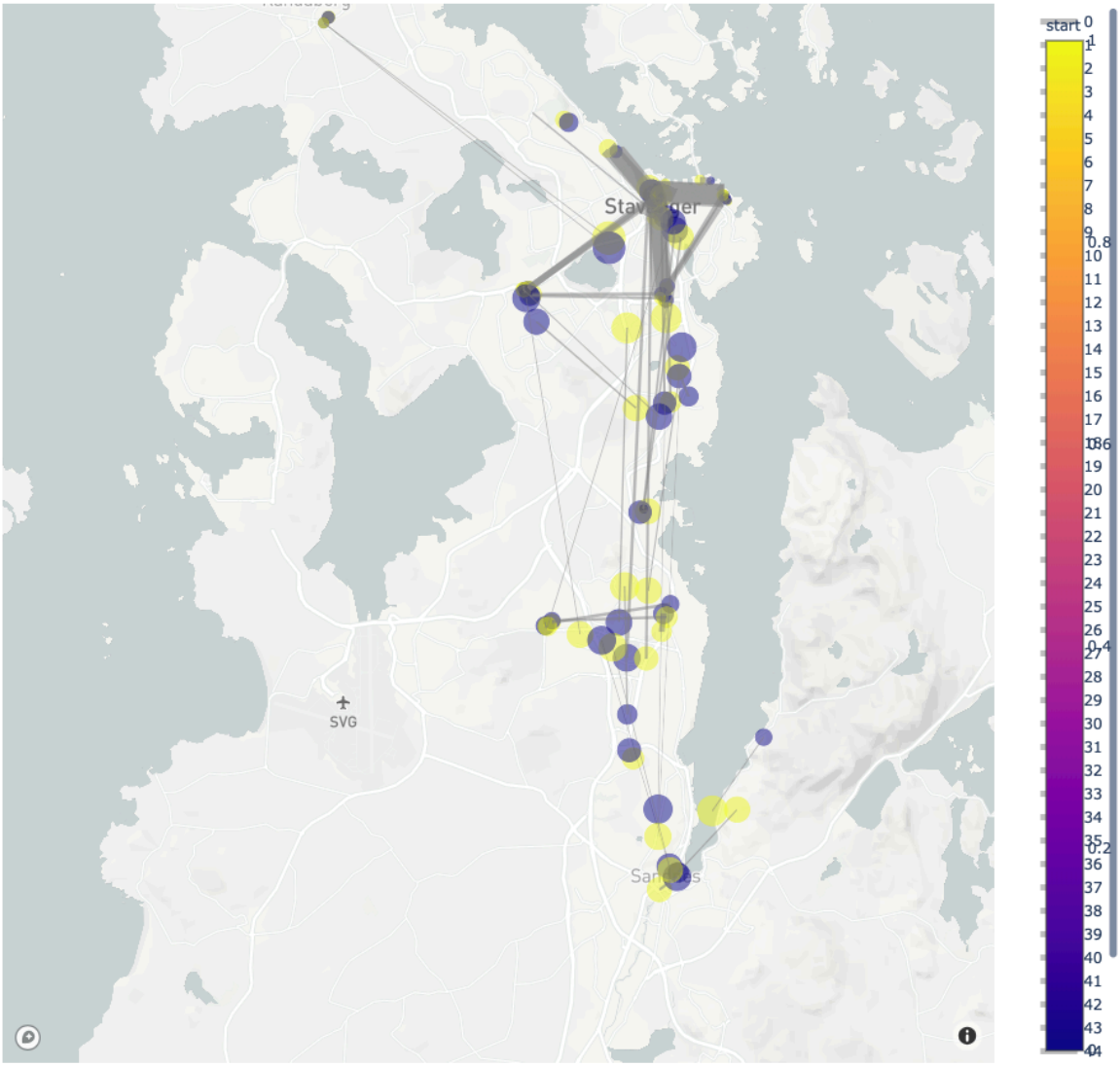
Her er det et eksempel fra bysykkel prosjektet for Kolumbus som vi jobber med:

Out[116]:



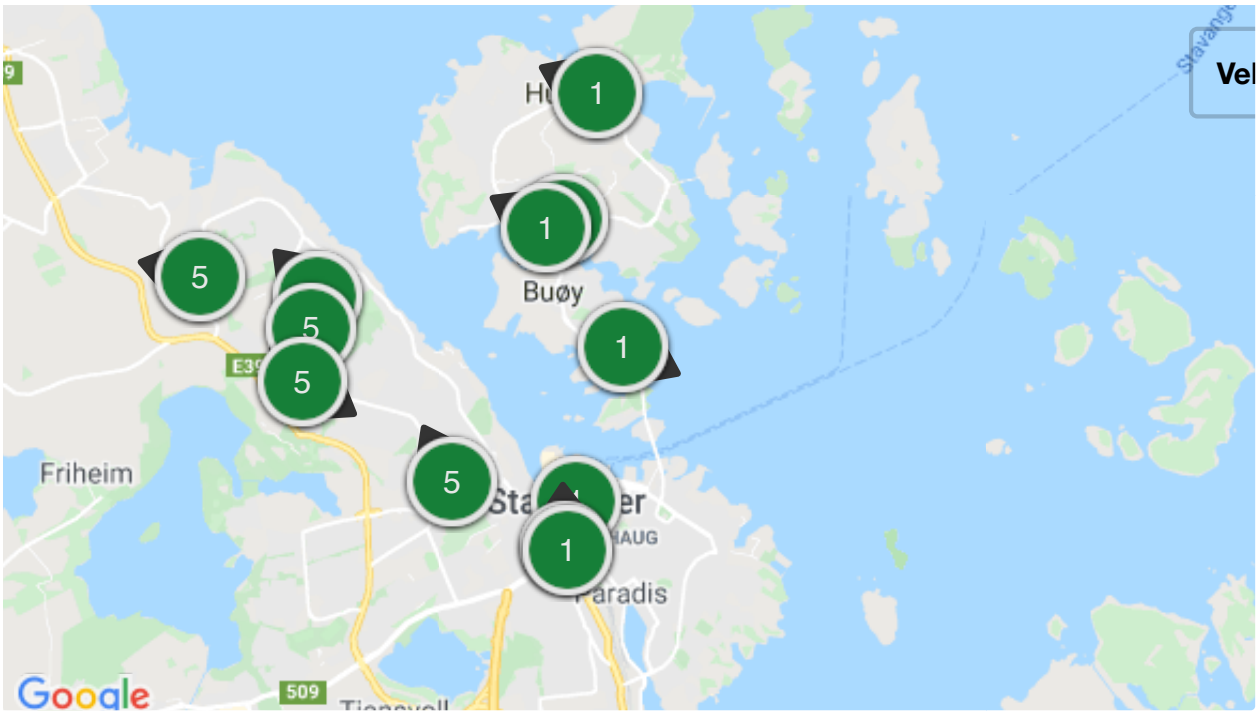
Visualisering av cluster:

Out[202]:



Tidsmaskin prosjekt i Kolumbus

Out[13]:



Takk for oppmerksomheten!

In [ ]: