Visualisation

You are required to produce a **visualisation** of food hygiene ratings for different establishments across London.

Your visualisation should contain a map-based display of the ratings, placed according to their geolocation data. A user should be able to intuitively see which businesses are 'safe' to eat at, and those which have not scored so well.

You will supplement the map with additional charts and graphics that you deem appropriate to tell a coherent story from the data available.

Question 0

In this assignment, you will use Bokeh, in particular `WMTSTileSource` class to add points to a map. You will use a convenience package `bokeh.tile_providers` (http://bokeh.pydata.org/en/latest/docs/reference/tile_providers.html) which creates a `WMTSTileSource` (like used in the guided exercise) instance with the `url` and `attribution` already set. So, instead of manually creating a tile, you can just use one of the variables already created. See the Bokeh source code (https://github.com/bokeh/bokeh/blob/master/bokeh/tile_providers.py) to look at how they have done it.

The available tiles supported by Bokeh use *Web Mercator* (https://en.wikipedia.org/wiki/Web_Mercator) format to represent location, so a function `wgs84_to_web_mercator` to convert the two is provided.

Run the code in the cell below to set up the Notebook.

In [1]:

```
# You don't need to write anything here
# Set up MongoDB
from pymongo import MongoClient

client = MongoClient('mongodb://cpduser:M13pV5woDW@mongodb/health_data', 27017)
db = client.health_data

from nose.tools import *

# # Numpy, Pandas and Bokeh imports
import numpy as np
import pandas as pd
from bokeh.palettes import Spectral6
from bokeh.io import output_notebook, show
from bokeh.models.sources import ColumnDataSource
from bokeh.models import *
from bokeh.io import curdoc
from bokeh.tile_providers import *
from bokeh.models.tiles import WMTSTileSource
import ipywidgets
from ipywidgets import interact, interactive
from ipywidgets import HBox, Label, IntSlider

from bokeh.plotting import figure
from bokeh.models import TapTool, CustomJS




def wgs84_to_web_mercator(df, lon="lon", lat="lat"):
    """
    Converts decimal longitude/latitude to Web Mercator format
    Source https://github.com/bokeh/bokeh-notebooks/blob/master/tutorial/11%20-%20ge
    """
    k = 6378137
    df["x"] = df[lon] * (k * np.pi/180.0)
    df["y"] = np.log(np.tan((90 + df[lat]) * np.pi/360.0)) * k
    return df

# from ipywidgets import *
# from bokeh.layouts import *
from IPython.display import display
from bokeh.io import output_file, output_notebook, show, push_notebook

output_notebook()
```

(http://bokeh.pydata.org) BokehJS successfully loaded.

In [2]:

```
# You don't need to write anything here
# Check it's set up correctly
try:
    imports = [MongoClient, db, np, pd, output_notebook, show, ColumnDataSource,
                output_notebook, show, ColumnDataSource, STAMEN_TERRAIN, figure
               ]
    assert True
    print('Successfully imported required libraries')
except NameError as e:
    print(e)
    assert False
```

```
Successfully imported required libraries
```

Question 1: Create Map

In this question, you will create functions which will **return** the different objects required for the visualisation of a map on: A `DataFrame` (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html), a `ColumnDataSource` and a `Figure`.

Question 1(a) [4 marks]

Create a function **`get_data()`** to extract data from the MongoDB database for all institutions which are in the **London** region with the following constraints:

- The results should include: `Lat`, `Lng`, `BusinessType`, `AddressLine1`, `BusinessName`, `RatingValue` but **NOT** the `_id` field
- The results should **only include businesses which have a RatingValue** (N.B. A value of 0 is a RatingValue)
- The results returned should **only include businesses which have a Geocode**
- The returned values should be **limited to 200** institutions
- **Add fields `x` and `y` in *Web Mercator* format** to specify co-ordinates on the map
- **Return** the result as as `DataFrame` (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html).

Hint: Week 3, Guided Exercise 4, Cursors
Hint: Week 4, Guided Exercise 2, Importing Data

In [3]:

```
def get_data():
    dff = db.uk.find({'Region' : 'london', 'Geocode' : {'$ne' : None }, 'RatingValue
                     {'Lat' : 1, 'Lng' : 1, 'BusinessType' : 1,  'AddressLine1' : 1,

    listy = list(dff)
    df1 = pd.DataFrame(listy)
    df = wgs84_to_web_mercator(df1, 'Lng', 'Lat')

    return df

    #return businesses (RV와 Geocode가 존재하고 200개 기관으로 제한 되어있어야함 비지니스만 +데이E
    raise NotImplementedError()
get_data().head()
```

Out[3]:

	AddressLine1	BusinessName	BusinessType	Lat	Lng	Rating
0	Bromley	118 Widmore Road	Hospitals/Childcare/Caring Premises	51.406561	0.028132	3
1	The 19th Hole Cafe	19th Hole Cafe	Restaurant/Cafe/Canteen	51.431513	0.076540	1
2	80 Grovelands Road	A & A Stores	Retailers - other	51.411260	0.104858	2
3	61 Anerley Road	A Butterfly	Retailers - other	51.416480	-0.072576	3
4	143 High Street	Aambal Local Store	Retailers - other	51.359719	0.072717	3

In [4]:

```
# You don't need to write anything here
test_data =  get_data()
assert_equal(type(test_data),pd.DataFrame)

for index, row in test_data.iterrows():
    try:
        i = int(row['RatingValue'])
    except:
        raise AssertionError('There is a row which is not an integer.  '
                             'Make sure you exclude all those without a RatingValue
        assert False

#print('All tests passed successfully')
assert_equal(len(test_data.index), 200)
print('All tests passed successfully')
```

```
All tests passed successfully
```

Question 1(b) [2 marks]

Create a function **`get_source`** which takes a **`DataFrame`** as a parameter and manipulates it to prepare for addition to the plot. The function should:

- **Contain a column `Colour`**, which contains a hex string of the colour with which to display the establishment on the map, e.g., #d53e4f. This should be used to distinguish different RatingValue values of the businesses.
- **RatingValues will be displayed by different colours** using an appropriate palette such as **`Spectral6`** from the standard Bokeh palettes (http://bokeh.pydata.org/en/latest/docs/reference/palettes.html)
- The function should **accept an integer** as a number to filter the businesses by `RatingValue`.
- If the rating value is equal to **`-1`**, then **all businesses should be included**. Otherwise, the data should be filtered to only include businesses with a `RatingValue` of the value passed.
- The function should **return** the result as a **`DataFrame`**

Hint: Week 5, Guided Exercise 2, Data Sources

In [7]:

```
def get_source(df, data_filter=-1):
    # YOUR CODE HERE
    Colour2 = Spectral6
    df['Colour'] = '#ffffff'
    for e in range(len(df)):
        RV = df['RatingValue'][e]
        Colour = ''
        if RV == 0:
            df['Colour'] == Colour2[0]
        elif RV == 1.0:
            df['Colour'] == Colour2[1]
        elif RV == 2.0:
            df['Colour'] == Colour2[2]
        elif RV == 3.0:
            df['Colour'] == Colour2[3]
        elif RV == 4.0:
            df['Colour'] == Colour2[4]
        elif RV == 5.0:
            df['Colour'] == Colour2[5]
        #elif RV == -1.0:
            #df['Colour'] == Colour2[0:5]

    df.loc[df['RatingValue'] == 0, 'Colour'] = Colour2[0]
    df.loc[df['RatingValue'] == 1, 'Colour'] = Colour2[1]
    df.loc[df['RatingValue'] == 2, 'Colour'] = Colour2[2]
    df.loc[df['RatingValue'] == 3, 'Colour'] = Colour2[3]
    df.loc[df['RatingValue'] == 4, 'Colour'] = Colour2[4]
    df.loc[df['RatingValue'] == 5, 'Colour'] = Colour2[5]

    if data_filter == -1:
        return df
    else:
        filter1 = df['RatingValue'] == data_filter
        filter2 = df[filter1]
        return filter2

    raise NotImplementedError()
get_source(get_data(), data_filter=-1).head()
```

Out[7]:

	AddressLine1	BusinessName	BusinessType	Lat	Lng	Rating
0	Bromley	118 Widmore Road	Hospitals/Childcare/Caring Premises	51.406561	0.028132	3
1	The 19th Hole Cafe	19th Hole Cafe	Restaurant/Cafe/Canteen	51.431513	0.076540	1
2	80 Grovelands Road	A & A Stores	Retailers - other	51.411260	0.104858	2
3	61 Anerley Road	A Butterfly	Retailers - other	51.416480	-0.072576	3
4	143 High Street	Aambal Local Store	Retailers - other	51.359719	0.072717	3

In [8]:

```
# You don't need to write anything here
test_source = get_source(get_data())
assert_equal(type(test_source),pd.DataFrame)
# Check the colours are different
rating_value_1 = test_source.loc[test_source['RatingValue'] == 1]
rating_value_2 = test_source.loc[test_source['RatingValue'] == 2]
rating_value_3 = test_source.loc[test_source['RatingValue'] == 3]
rating_value_4 = test_source.loc[test_source['RatingValue'] == 4]
rating_value_5 = test_source.loc[test_source['RatingValue'] == 5]

colour_list = [rating_value_1['Colour'].values[0],
               rating_value_2['Colour'].values[0],
               rating_value_3['Colour'].values[0],
               rating_value_4['Colour'].values[0],
               rating_value_5['Colour'].values[0]]
# If they are all different, then the length of the set should be 5
assert_equal(len(set(colour_list)), 5)

fields = test_source.columns.values
assert 'x' in fields
assert 'y' in fields

# Test that the x and y are the correct web mercator format
test_lng = rating_value_1['Lng'].values[0]
test_lat = rating_value_1['Lat'].values[0]

test_x = rating_value_1['x'].values[0]
test_y = rating_value_1['y'].values[0]

k = 6378137
# print(test_lat * (k * np.pi/180.0))

assert_equal(test_x, test_lng * (k * np.pi/180.0))
assert_equal(test_y, np.log(np.tan((90 + test_lat) * np.pi/360.0)) * k)
print('All tests passed successfully')
```

```
All tests passed successfully
```

Question 1(c) [2 marks]

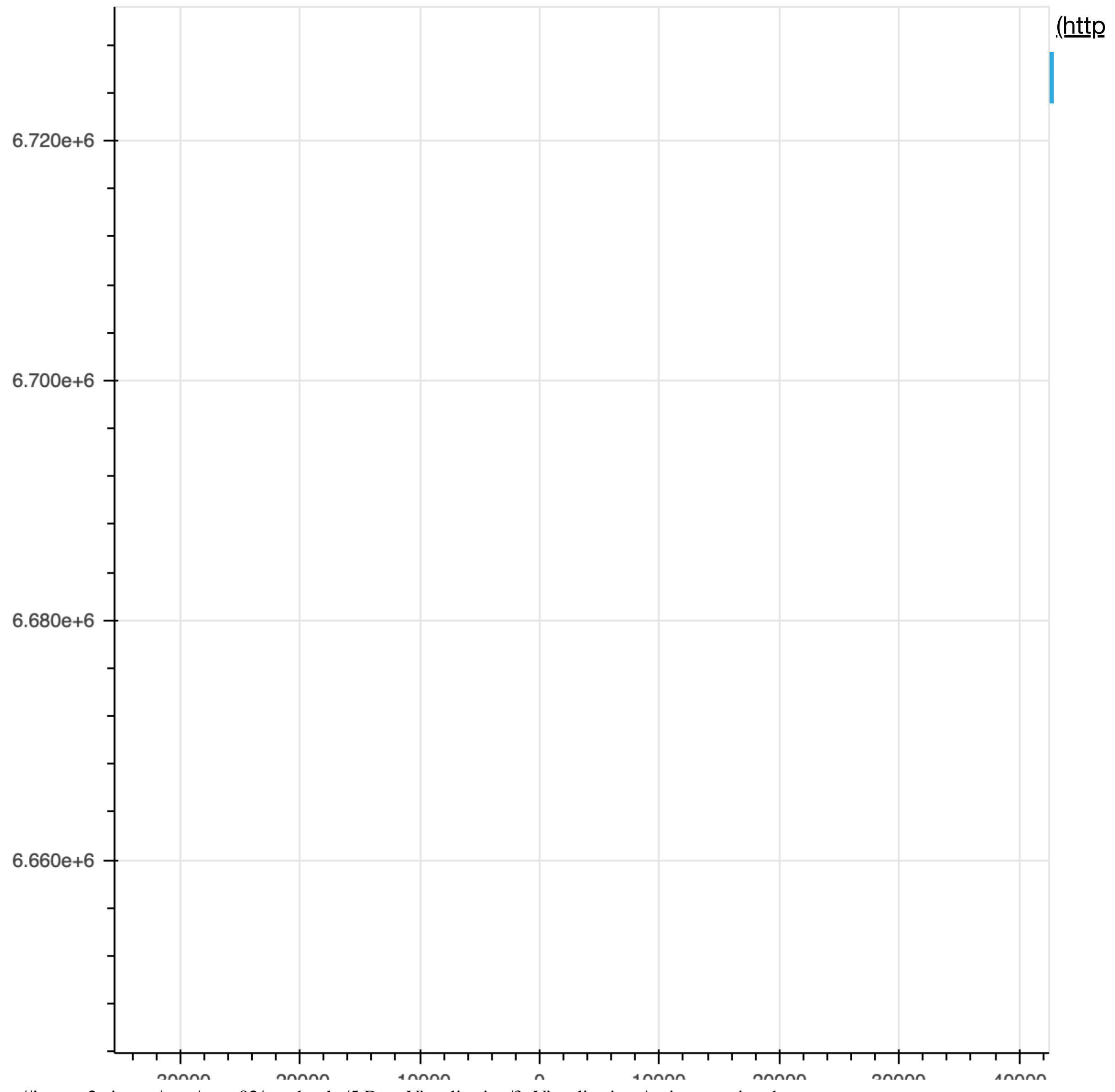
Create a function **`get_map`**, which **returns** a map of London using the `STAMEN_TERRAIN` (http://bokeh.pydata.org/en/latest/docs/reference/tile_providers.html#bokeh.tile_providers.STAMEN_TERRAIN) tile

- The function should **return** a **type** `Figure`
- The figure should include **all** the available **Pan/Drag tools (http://bokeh.pydata.org/en/latest/docs/user_guide/tools.html#pan-drag-tools)**, the **reset** tool, and the **mouse wheel zoom** tool
- The map should display London, at an appropriate zoom level for the data

Hint: To get the `x,y` values surrounding London, look at the smallest and largest `x` and `y` values in the data
Hint: Week 4, Guided Exercise 3, Residual Analysis (attributes)
Hint: Week 5, Guided Exercise 2, Data Sources (Bokeh Map Tiling)

In [9]:

```
from bokeh.io import output_file, show
from bokeh.layouts import column
from bokeh.plotting import figure


def get_map(data):
    """
    In this function you return a figure with a map background.  The background shou
    on London at an appropriate zoom level
    """
    # YOUR CODE HERE
    url = 'http://tile.stamen.com/terrain/{z}/{x}/{y}.jpg'
    x_range,y_range = ((-8123.87379911,15202.9028576), (6675917.19275,6699112.66832
    fig = figure(tools='pan, reset, wheel_zoom', x_range=x_range, y_range=y_range)
    fig.add_tile(WMTSTileSource(url = url))



    return fig
    raise NotImplementedError()

show(get_map(get_source(get_data())))
```



Question 1(d) [3 marks]

Write code which creates and shows a figure `london_map` using the `get_map()` function, obtains a data source for the figure using `get_data()`, then uses the `circle` method to add the data to the map.

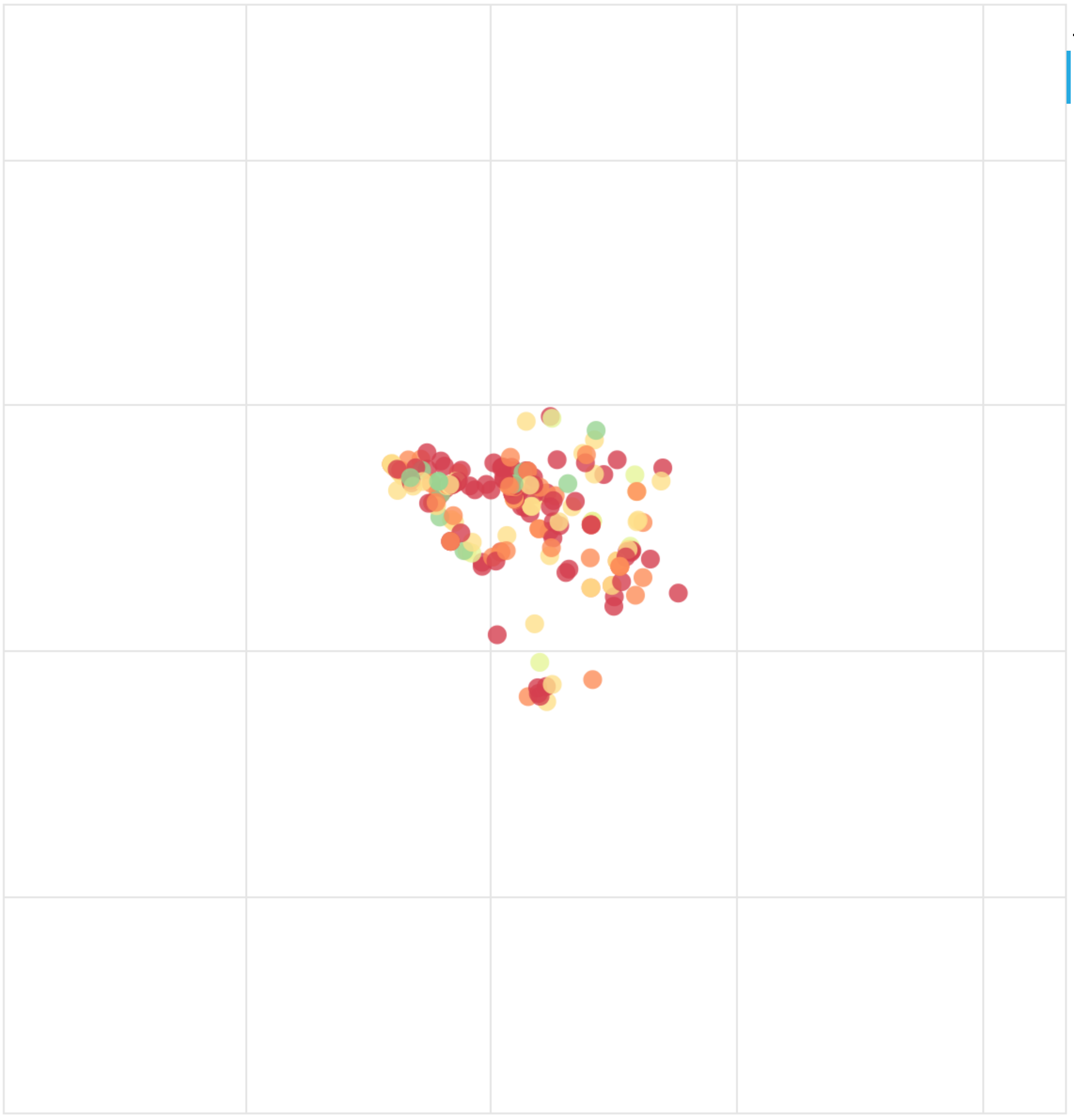
- You should call the output of the `circle` function `data_points`.
- The dots you add to the map should have a size of 10, no border, and a `fill_alpha` of 0.8
- You should call your map `london_map`
- Your code should contain a variable name for the dots added to the map

N.B. You are not being asked to create a function for this question

Hint: Week 4, Guided Exercise 3, Fitting a Model - Residual Analysis
Hint: Week 5, Guided Exercise 2, Widgets

In [10]:

```
source = get_source(get_data())
london_map = get_map(source)


data_points = london_map.circle(source = source, x = 'x', y = 'y',
                                fill_color = 'Colour', fill_alpha = 0.8, size = 10, l

london_map.axis.visible = False
show(london_map)
```



(http

In [11]:

```
fig_source = london_map.select(GlyphRenderer)[0].data_source
#print(vars(fig_source))
assert_equal(fig_source.data['x'][0], source['x'][0])
assert_equal(fig_source.data['y'][0], source['y'][0])
assert_equal(fig_source.data['Colour'][0], source['Colour'][0])
#assert_equal(fig_source.data['fill_color'][0], ds.data['Colour'][0])


glyph = london_map.select(GlyphRenderer)[0].glyph
assert_equal(glyph.line_color, None)
assert_equal(glyph.size, 10)
assert_equal(glyph.fill_alpha, 0.8)
#glyph.fill_color == (ds.data['Colour'][0])
print('All tests passed successfully')
```

```
All tests passed successfully
```

Question 2: Make it Interactive

Question 2(a) [2 marks]

Create a function **`callback`** for later use, which updates the visible businesses on the map according to their **`RatingValue`**. The function should have parameter **`rating`** which specifies the value to filter by, calling **`get_source`**. Use the **`source`** variable from Question 1(d) to update the map.

Hint: Week 4, Guided Exercise 2, Bokeh Charts
Hint: Week 5, Guided Exercise 2, Data Sources - Widgets (update figure)

In [15]:

```
def callback(rating):
    # YOUR CODE HERE
    source = get_source(get_data())
    london_map = get_map(source)

    if rating == -1:
        data = source
        Colour = data['Colour']
        data_points = london_map.circle(source = source, x='x' , y='y',
                                        fill_color=Colour, size=10, fill_alpha=0.8,

        data_points.data_source.data['x'] = data['x']
        data_points.data_source.data['y'] = data['y']
        push_notebook()
        london_map.axis.visible = False
        show(london_map, notebook_handle=True)
        return london_map
    else:
        data = source.loc[source['RatingValue'] == rating]
        colour = source.loc[source['RatingValue'] == rating]['Colour'].iloc[0]
        data_points = london_map.circle(source = source, x = 'x', y = 'y',
                                       fill_color = colour, fill_alpha = 0.8, size =

        data_points.data_source.data['x'] = data['x']
        data_points.data_source.data['y'] = data['y']
        push_notebook()
        london_map.axis.visible = False
        show(london_map, notebook_handle=True)
        return london_map
    raise NotImplementedError()
```

Question 2(b) [2 marks]

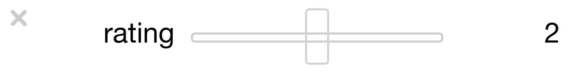
Using **`ipywidgets`**, create an interactive (http://ipywidgets.readthedocs.io/en/latest/examples/Using%20Interact.html) `IntSlider` widget, which calls the **`callback`** function when it updates.

Return the interactive widget in the function **`set_interactive()`**

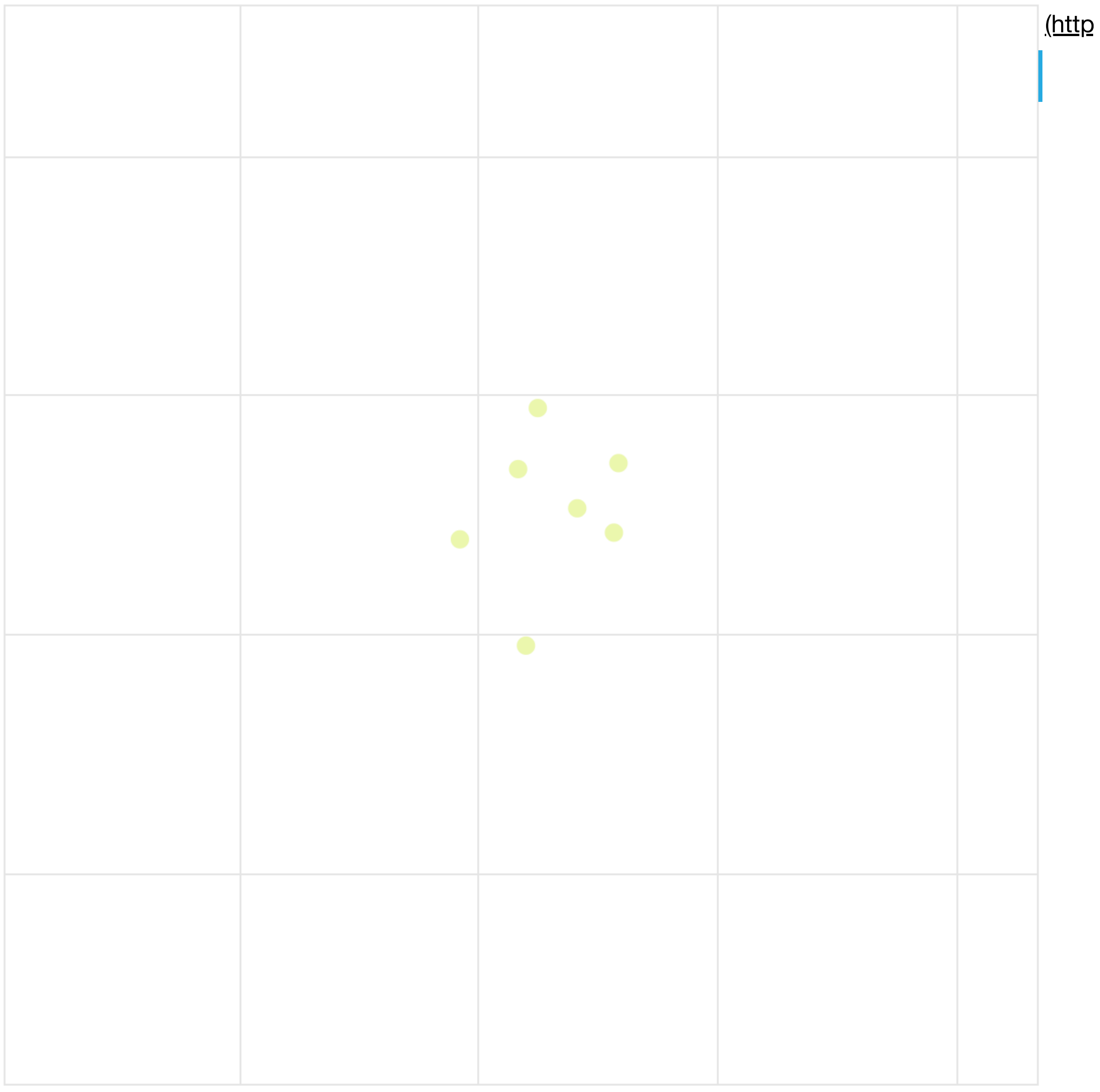
Hint: Week 5, Guided Exercise 2, Widgets

In [19]:

```
def set_interactive():
    # YOUR CODE HERE
    i = interactive(callback, rating=(-1,5))

    return i
    raise NotImplementedError()

set_interactive()
```



```
/opt/conda/lib/python3.5/site-packages/bokeh/models/sources.py:81: Bok
ehUserWarning: ColumnDataSource's columns must be of the same length
  lambda: warnings.warn("ColumnDataSource's columns must be of the sam
e length", BokehUserWarning))
/opt/conda/lib/python3.5/site-packages/bokeh/models/sources.py:81: Bok
ehUserWarning: ColumnDataSource's columns must be of the same length
  lambda: warnings.warn("ColumnDataSource's columns must be of the sam
e length", BokehUserWarning))
```



Figure(id = '68105a91-7f42-4d5e-af6b-8211e7f7ab8b', …)

In [20]:

```
# You don't need to write anything here
old_callback = callback
del callback

try:
    set_interactive()
except NameError as e:
    pass
else:
    raise AssertionError('You have not called the callback function in your code')
finally:
    callback = old_callback
    del old_callback
assert_equal(type(set_interactive()), ipywidgets.widgets.widget_box.Box)
```

Question 3: Extend the Visualisation

Applying question 2 solutions

Now you have created an initial visualisation, you are going to add the following components to it:

- **Hover** text, so that each dot will give information about the business when you hover
- A **drop down menu** to limit the type of business

You will also be asked to explain a possible use-case for this chart, and offer a suggestion as to how it could be improved.

NOTE: There are discretionary marks available for good visualisation practice

Question 3(a) [5 marks]

Create a function **`get_hover`**, which returns a HoverTool (http://bokeh.pydata.org/en/latest/docs/user_guide/tools.html#hover-tool) to be added to the map. When the cursor hovers over any circle, the following information should be displayed:

- The **name** of the establishment
- The **type** of the establishment
- The **`RatingValue`** of the establishment

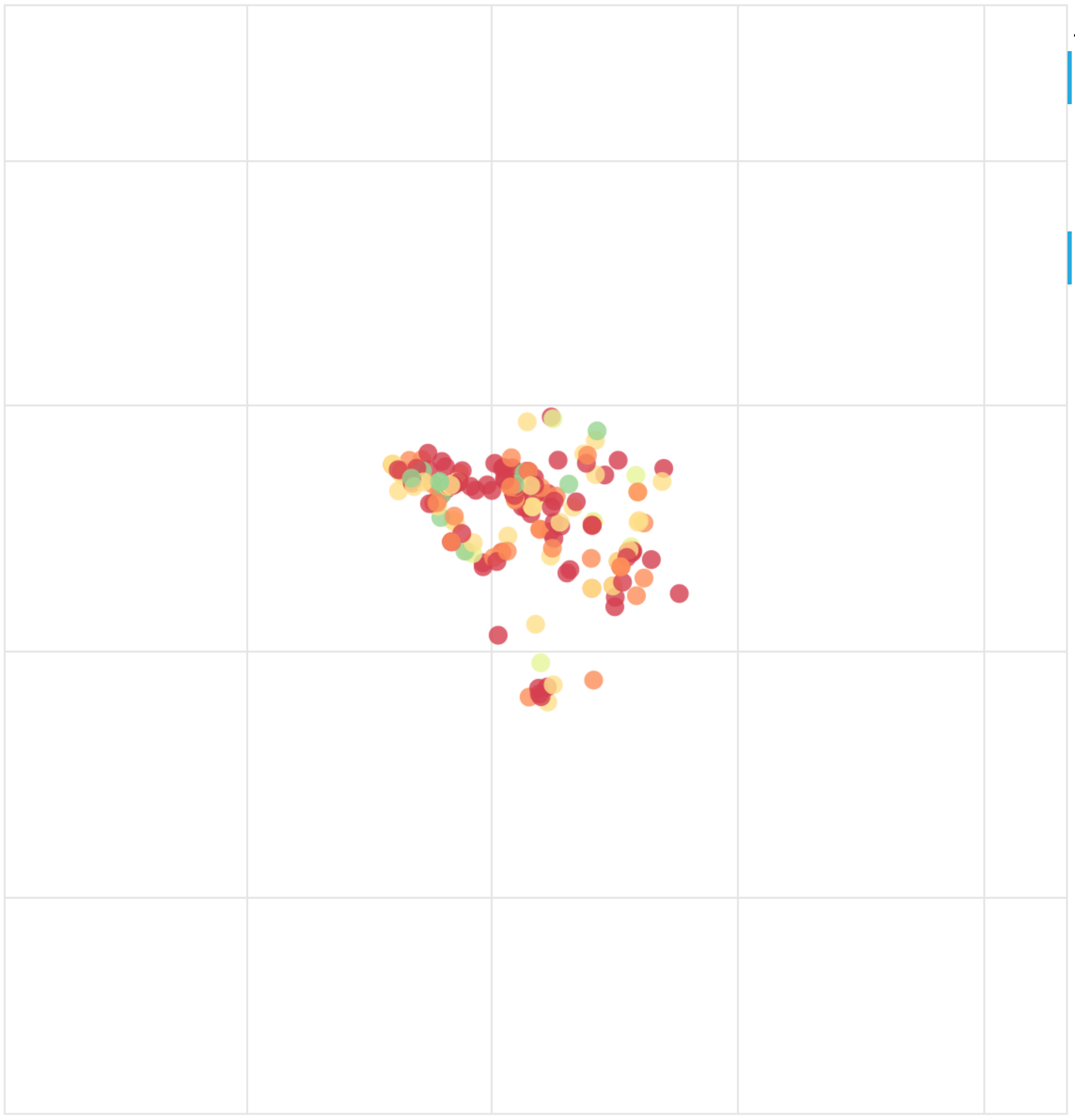
Your function should **return** the HoverTool (http://bokeh.pydata.org/en/latest/docs/user_guide/tools.html#hover-tool).

N.B You may need to read the documentation available to ensure that you match a suitable title for the **field** with the **field value** using **`@`**.

Hint: Week 5, Guided Exercise 2, Tools

In [21]:

```
def get_hover():
    # YOUR CODE HERE
    hover_tool = HoverTool(
    tooltips=[
        ( 'BusinessName',   '@BusinessName'            ),
        ( 'BusinessType',  '@BusinessType' ), # use @{ } for field names with spaces
        ( 'RatingValue', '@RatingValue'      )
    ],

    # display a tooltip whenever the cursor is vertically in line with a glyph
    #mode='vline'
    )

    return hover_tool
    raise NotImplementedError()

london_map.add_tools(get_hover())
show(london_map)
```



(http

Question 3(b) [5 marks]

Using `interactive` (http://ipywidgets.readthedocs.io/en/latest/examples/Using%20Interact.html), create a **dropdown menu** which allows the user to choose between different **types of business.** The map should update automatically as you select.

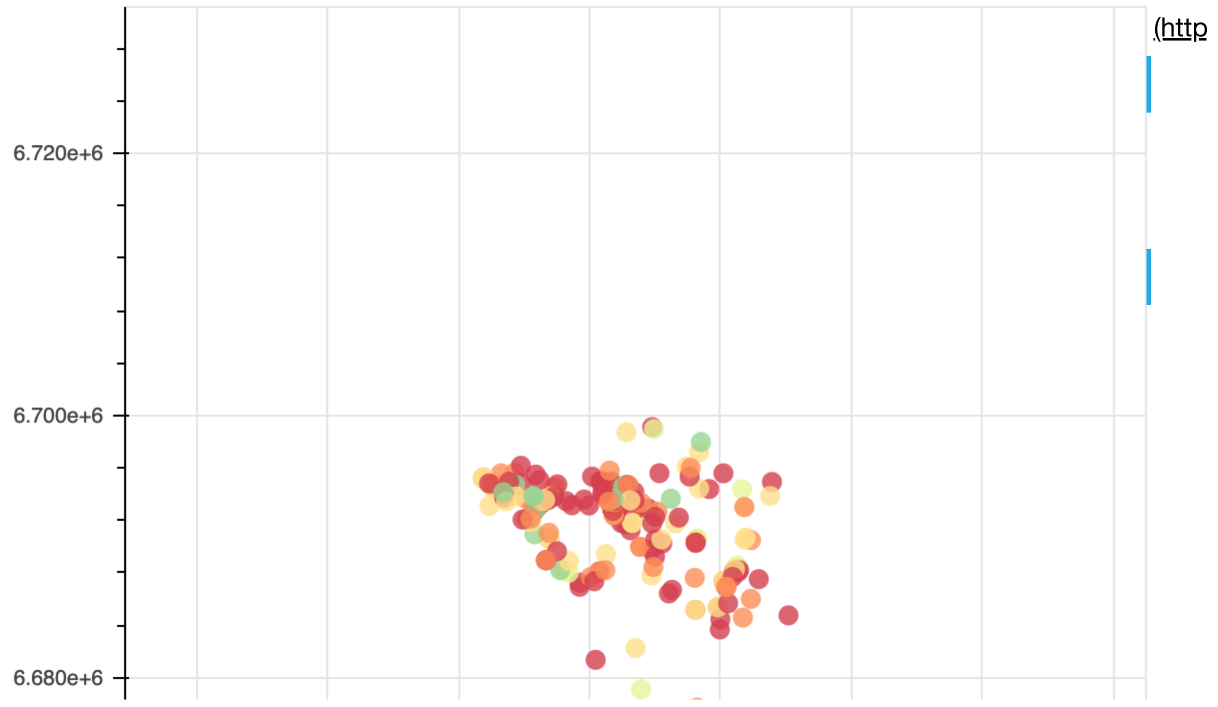
- You should create two functions **`filter_business_types`**, which the dropdown menu will call when it changes. This function should update the data source
- You will need to obtain a list of the different types of business in the database
- You should **return** an **`interactive`** object, which should call the function **`filter_business_types`**, which updates the **`source`** variable accordingly

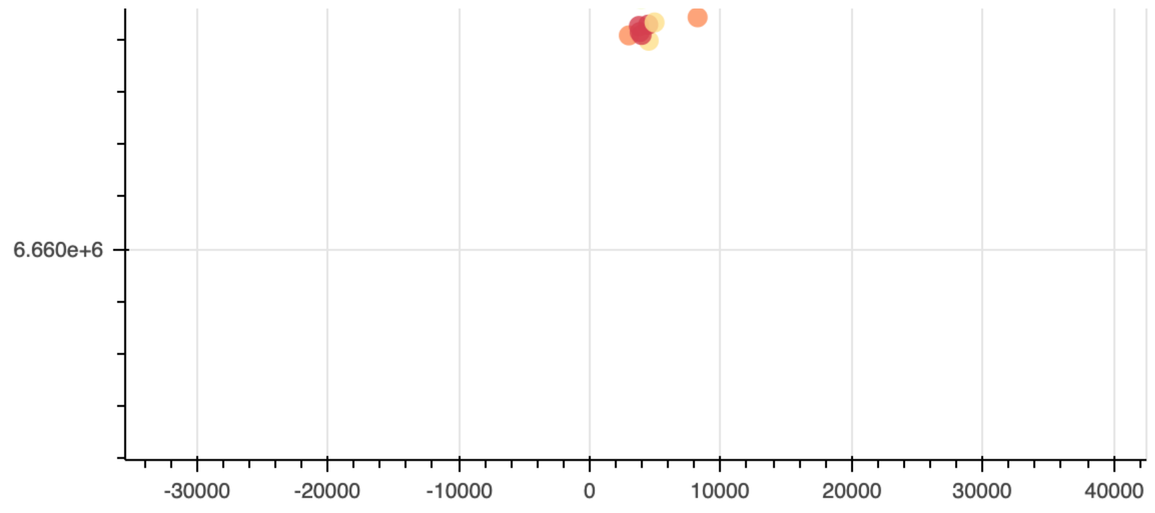
N.B. Should you use the original data set?

Hint: Week 5, Guided Exercise 2, Data Sources - Tools

In [24]:

```
def filter_business_types(business_type):
    # YOUR CODE HERE
    source = get_source(get_data())
    london_map = get_map(source)

    data = source.loc[source['BusinessType'] == business_type]
    Colour = data['Colour'].iloc[0]
    data_points = london_map.circle(source=source, x='x', y='y',
                                    fill_color='Colour', size=10, fill_alpha=0.8, l

    data_points.data_source.data['x'] = data['x']
    data_points.data_source.data['y'] = data['y']
    push_notebook()

    show(london_map, notebook_handle=True)
    return london_map

    raise NotImplementedError()

def get_dropdown_list():
    # YOUR CODE HERE

    j = source['BusinessType'].drop_duplicates()
    listie = j[0], j[1], j[2], j[24], j[26], j[28], j[39], j[40], j[63], j[78], j[1
    kkk = interactive(filter_business_types, business_type = (listie) )
    return kkk
    raise NotImplementedError()

# DISPLAY THE MAP
data = get_data()
source = get_source(data)
mappy = get_map(data)
dots = mappy.circle(source=source, x='x', y='y',fill_color='Colour', size=10, fill_
mappy.add_tools(get_hover())
dropdown = get_dropdown_list()
slider = set_interactive()
show(mappy, notebook_handle=True)
```



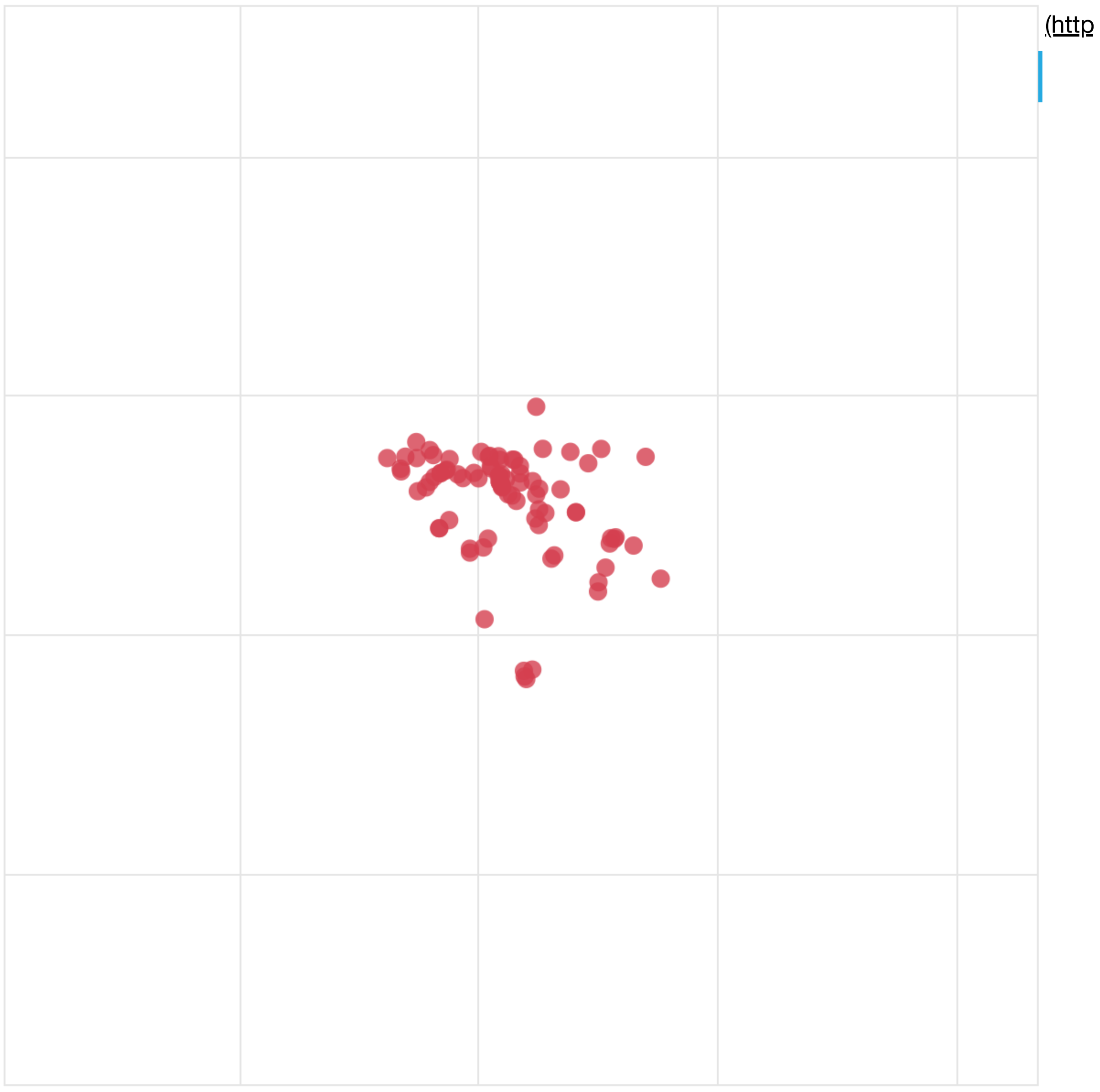


```
Out[24]:

<Bokeh Notebook handle for In[24]>
```

In [25]:

```
# You don't have to write anything here
# Display the widgets
HBox([slider, dropdown])

# Why do you think we might get a Bokeh warning diplayed when we use our data?
```

```
/opt/conda/lib/python3.5/site-packages/bokeh/models/sources.py:81: Bok
ehUserWarning: ColumnDataSource's columns must be of the same length
  lambda: warnings.warn("ColumnDataSource's columns must be of the sam
e length", BokehUserWarning))
/opt/conda/lib/python3.5/site-packages/bokeh/models/sources.py:81: Bok
ehUserWarning: ColumnDataSource's columns must be of the same length
  lambda: warnings.warn("ColumnDataSource's columns must be of the sam
e length", BokehUserWarning))
```



Figure(id = '117787ed-def2-4b8b-9e03-4020f78fb287', …)

Question 3(c) [5 marks]

Describe a use case for which an application like this would be useful, and suggest one way which it could be improved.

이와 같은 유용한 어플리케이션 사례는 코로나 맵이 있습니다. 요즈음 코로나 바이러스 문제로 인하여 확진자가 발생하면 동선을 나타나는 맵을 나라별, 지역별 코로나 맵을 만들어서 어플리케이션 이용하는 사람들에게 확진자가 어디를 다녀왔는지 시각적으로 알아 볼 수 있습니다. 이를 개선시킬 방법으로는 레이팅 벨류가 0부터 5까지 구분 되는 것 처럼, 일자별로 구분되게 필터링을 만들어서 좀 더 이용자가 알아보기 쉽고 유용하게 어플리케이션을 사용하는 방법이 있습니다.

```
In [ ]:
```