Exercise Vz

Data visualization

Prior Knowledge
Unix Command Line Shell
Simple Python
Completion of Exercise 4, part B

Learning Objectives
To try out creating charts in Python
Connecting your data processing to visualizations

Software Requirements
(see separate document for installation of these)

- Python 2.7.x
- Sublime text editor or any other text editor
- A Web browser

bokeh

Bokeh is a Python library for developing charts, including interactive ones. Lucky for us, they provide some really nice introductory exercises (the originals can be found at http://goo.gl/3y0vAD). In Part A we'll start with one of these exercises.

PART A – Plot a basic scatter graph with Bokeh

First, we want to get familiar with how to do a first basic chart using Bokeh in Python to load some built-in sample data that comes with the library. Let's make sure the `bokeh` package in the VM's Python environment is installed. Open up a terminal window and run

```
>>> sudo pip install bokeh
```

You should see a lot of logging and warnings while it is installing and eventually (it seems to take a while to install on the VM) ending with something like

```
Successfully installed bokeh...
```

And you'll also need to install `pandas` in order for Bokeh to make some other stuff work:

```
>>> sudo pip install pandas
```

That's all! We're ready to get started!

Start up a Python console by just running python on the command line. It's just like using the pyspark interactive console, but without connecting to Spark and just doing some stuff in Python.

We need to import a couple of functions to let us render our output.

```
>>> from bokeh.io import output_file, show
```

Next, we can tell Bokeh where to write our output from the interactive Python console. We can specify to write out to an HTML file called clo.html

```
>>> output_file('clo.html', title='Bokeh Plot')
```

We're now ready to try plotting some charts! Bokeh includes some sample data that we can import and use straight away. The next two lines import some data about flowers, and the head() function prints out the first few rows of a pandas.DataFrame (like a table).

```
>>> from bokeh.sampledata.iris import flowers
>>> flowers.head()
```

should output

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In Bokeh, charts can read in a range of data types including:

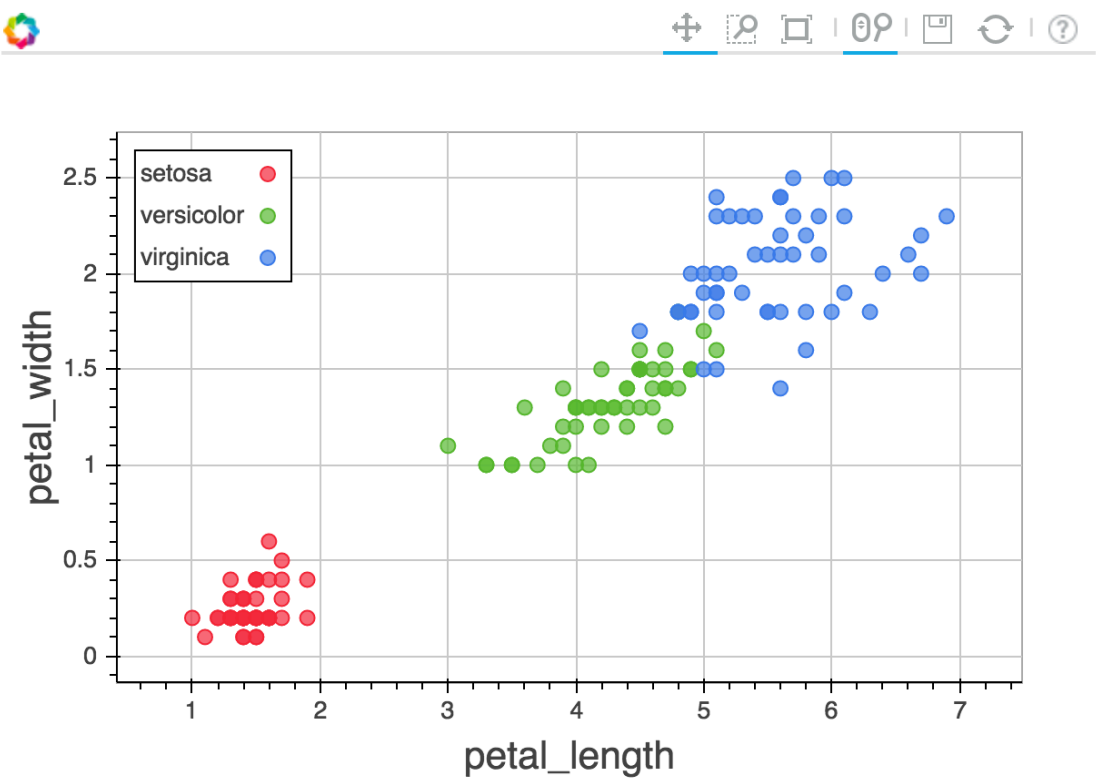
- Array-like objects - list, tuple, numpy.ndarray, pandas.Series
- Table-like objects - records: list(dict), columns: dict(list), pandas.DataFrame

We can now plot this data using one of many charts. Let's use a scatterplot.

```
>>> from bkcharts import Scatter
>>> p = Scatter(flowers, x='petal_length', y='petal_width', color='species', \ legend='top_left')
>>> show(p)
```

The first line imports our Scatter class that we use to build a chart. The next line creates the plot using the flowers DataFrame, and selecting the petal_length and petal_width as the x and y axes respectively. In the dataset there are three categories of species, and Bokeh can automatically assign each a colour simply by providing the species list. The final line show(p) renders the HTML and launches the browser automatically with the rendered chart.

In your Web browser, you should see a chart like this:



Try interact with the chat using the toolbar at the top right and also directly clicking/dragging on the chart. All very easy, right!

PART B – Using Bokeh with your wind data

Remember your wind-mapper.py from Exercise 4, part B? We can modify this a bit to plot a chart showing the wind speeds for each time point by adding a little bit of code. This time we'll use a line graph. To create a line graph, you import figure to use, and you use the line() function to plot a line using some data. In this case x and y are Python lists of numbers, where we use x as a label for a time point (so wind speed reading 1, 2, 3 and so on) and y to plot the wind speed reading. There's a nice example of a line chart found at https://goo.gl/iZu4sw. You will need to figure out how to create the lists in order to pass the plotting data to figure.

```
from bokeh.plotting import figure
p = figure(title="Wind speeds", x_axis_label='x', \ y_axis_label='y')
p.line(x, y, legend="Mtr per sec.", line_width=2)
show(p)
```

If you run your wind mapper (using only one of your input wind data files), you should now you should be able to render a simple chart using Bokeh in Python displaying some wind data. Take care to make sure it doesn't try to render until the end of the logic in your code.

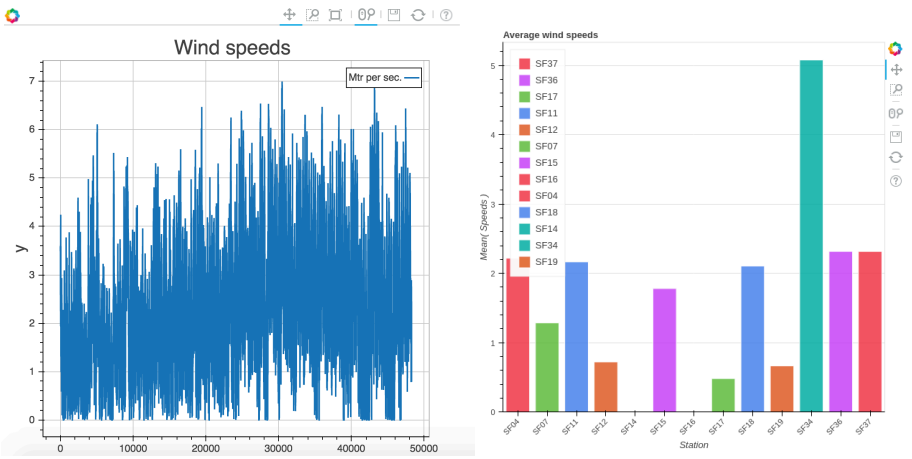
Try out plotting a chart using the output from the second part of Exercise 4, part B (the summarized wind speed averages and maximums that are output from wind-reducer.py) using a Bar chart to display the wind speed averages for each station. To create a bar chart, we can do something like this:

```
from bkcharts import Bar
p = Bar(data, 'station', values='speeds', title="Average wind
speeds", width=400, height=400)
```

where data is formatted as two columns (a dict of lists):

```
data = dict(stations=list(), speeds=list())
```

You'll need to figure out how to put the data into these columns yourself. Test this second part with the full set of wind data files (probably worth commenting out show() in wind-mapper.py, as rendering the full dataset may kill to death your browser). If all went well, we should end up with two charts looking something like the one of the left hand side (plotting speeds of all time points) and the one on the right (plotting average speeds per station):



You can even try adding the Python code to the end of your pyspark scripts to render the HTML output as a chart. Also, there's plenty of documentation on how to decorate your charts differently. Check out the full Bokeh documentation here: http://bokeh.pydata.org/en/latest/

Remember, bokeh charts can take as input lists and tuples, dicts of lists and lists of dicts, so these are probably easiest to put any series of data into from your analysis exercises.

If you're feeling really adventurous, try out integrating charts into the analyses using Spark from Exercise 6. Bokeh supports pandas.DataFrame objects, and in Spark we also have DataFrames – they both have the same interfaces so it *should* work.

For Exercise 6, you should be able recreate what we did in this exercise running in Spark, writing your output to file.

For Exercise 8, you can also try and create a visualization over Google Maps. Here's a great example of overlaying graphical shapes on maps (code and data are both included in the repository): https://github.com/queise/Berlin_maps

Note that it's not so straightforward getting it to work with Exercise 6b with Jupyter in Spark, as Livy runs your Jupyter Python context in pyspark inside your Spark master node, meaning your master node needs to be running the relevant Jupyter packages as well.

Congratulations! You have completed this lab and are now on your way to becoming a dataviz wiz.