D23 BOKEH - 輕鬆以網頁呈現視覺化圖表

簡報閱讀 範例與作業 問題討論 學習心得(完成)

重要知識點

- 完成今日課程後你應該可以了解
  # 使用Bokeh **將數據**轉換為可視化
  # **自定義和組織**可視化
  # 為可視化**添加交互性**

Bokeh

Bokeh 為一個 Python 函式庫，提供了各式各樣的視覺化必須的輔助函式，同時也將網頁前端的技術細節包裝成一個個的 Python 函式與參數供我們呼叫，讓我們不再需要編輯 HTML 與 JavaScript 便能製作網頁前端視覺化。

Bokeh 相關套件安裝

相關套件安裝

- NumPy, Jinja2, Six, Requests, Tornado
- PyYaml, DateUtil, pandas, bokeh, panel

需額外注意套件版本：

- Jinja2 >=2.7
- numpy >=1.7.1
- packaging >=16.8
- pillow >=4.0
- python-dateutil >=2.1
- PyYAML >=3.10
- six >=1.5.2
- tornado >=4.3

要注意的是bokeh會預設連BokehJS cdn，但連線有時不是很穩定，這時可多加"INLINE"環境變數設定，讓BokehJS驅動於local python env。

```
from bokeh.resources import INLINE
bokeh.io.output_notebook(INLINE)
```

Bokeh可以在Jupyter呈現開發也可以跳轉出html檔，可自由設定，預設是跳轉html檔(output_file())。

若要更改預設，必須加上bokeh.io.reset_output()重設環境預設。

```
output_notebook() # jupyter呈現
output_file() #html呈現
```

fig物件可設定tools參數，圖表會自帶縮放、重整、儲存等功能。

載入套件函數

- from bokeh.plotting import figure
- from bokeh.plotting import output_file
- from bokeh.plotting import show
- from bokeh.models import widgets
- from bokeh.io import output_notebook

讓網頁直接輸出在NOTEBOOK

- output_notebook()

設定資料與輸出檔案

- output_file("out.html")

利用 Bokeh 繪製圖表

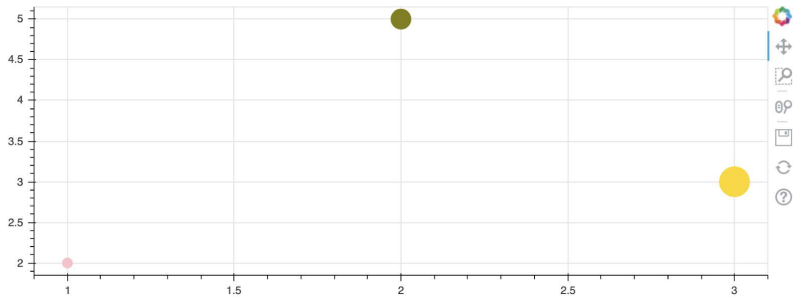
- p = figure()
- p.line([1,2,3,4,5], [5,4,3,2,1])

開啟產生的 HTML 檔 (HTML + JavaScript ，自動生成)

- show(p)

Bokeh 也提供了一些預先建置好的圖表供我們使用，例如用 circle 繪製點圖：

```
from bokeh.plotting import figure, output_file, show
p = figure(width=800,height=300)
p.circle([1,2,3],[2,5,3], size=[10,20,30], color=["pink","olive","gold"])
show(p)
```



除了繪製點圖的 circle 函式以外，另外還有以下不同圖形的繪製函數：

bokeh.charts.HeatMap — 製作熱圖
bokeh.charts.Donut — 製作甜甜圈圖

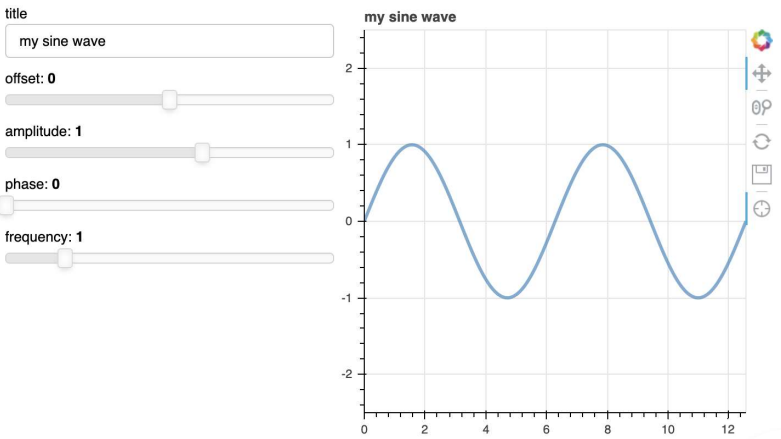
製作各式圖表， Bokeh 將函式庫大略分成三組：

- bokeh.model — 製作圖表的基本元素，例如軸線、形狀等等。用來打造各種元件。
- bokeh.plotting — 為我們處理掉一些基本細節 (例如格點與軸線) ，但保留客製化的彈性。
- bokeh.charts — 直接使用各種完整圖表，例如長條圖、盒鬚圖等等。

網頁元件與互動圖表

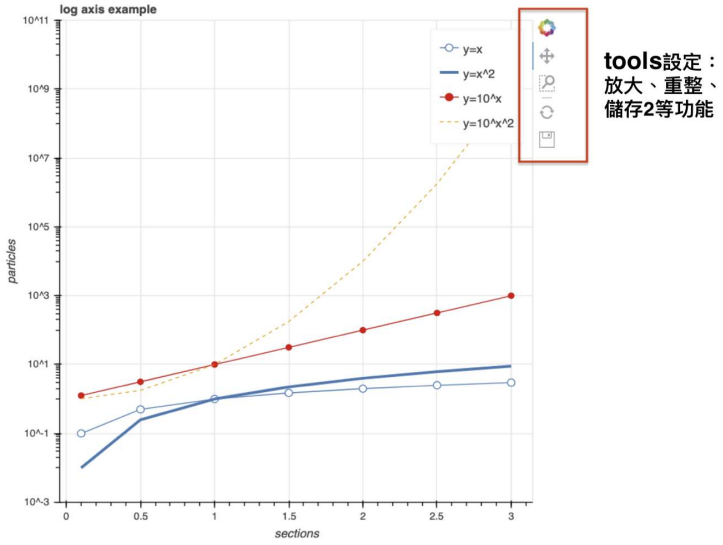
我們先透過 IFrame 直接呼叫 bokeh 範例來看看

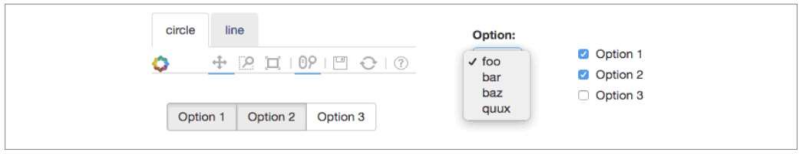
```
from IPython.display import IFrame
IFrame('https://demo.bokeh.org/sliders',
width=900, height=500)
```



製作互動圖表的能力

由於 Python 並沒有在網頁上執行的能力，這個其實是透過 HTML + Bootstrap 兜起來的元件





這些元件都位於 bokeh.models.widgets 下，可以透過 import 語法將其匯入。

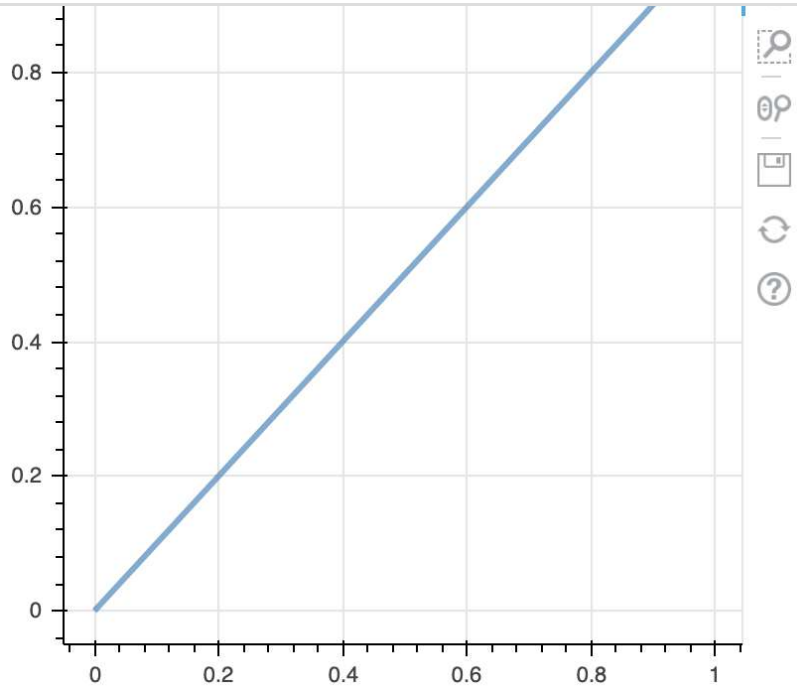
製作互動圖表 - CustomJS 回調

再來我們製作一個具有互動功能的圖表，透過 CustomJS 回調 JavaScript 代碼，我們讓小部件控制執行圖表操作。

```
# 載入函數
from bokeh.layouts import column
```

```
from bokeh.models import Slider
from bokeh.plotting import Figure, output_file,
show

# 指定html檔案名稱
output_file("js_on_change.html")

---

# 建立資料
x = [x*0.005 for x in range(0, 200)]
y = x

# 指定資料來源
source = ColumnDataSource(data=dict(x=x, y=y))

# 製作一個空畫布並繪製線段
plot = Figure(plot_width=400, plot_height=400)
plot.line('x', 'y', source=source,
          line_width=3, line_alpha=0.6)

# 呈現結果
show(plot)
```

此時我們得到一個單純線圖，再來我們要加上一些可控部件。

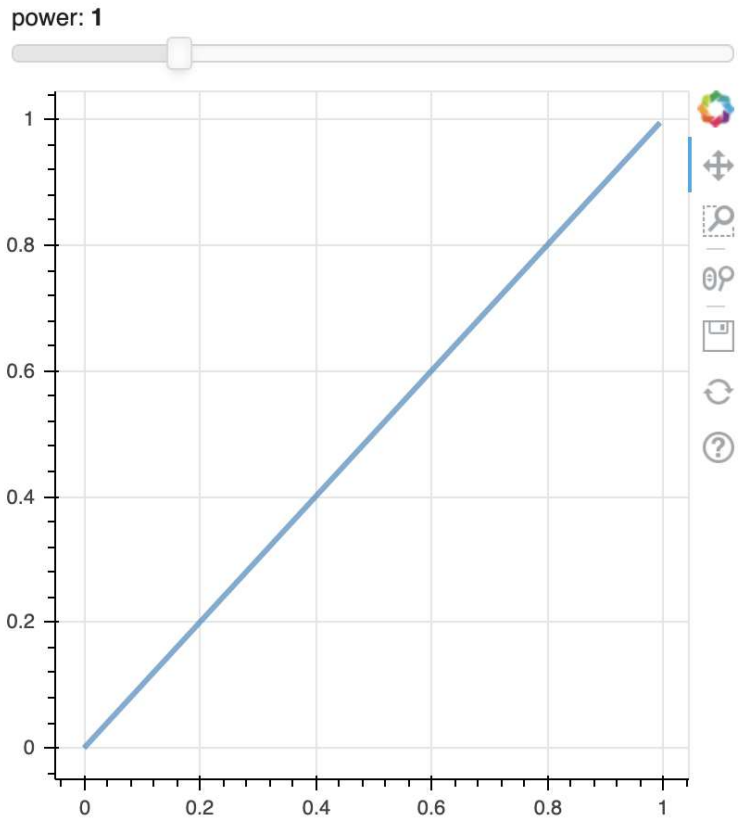



```
# 定義互動過程 (code="""JavaScript代碼""")
callback = CustomJS(args=dict(source=source),
code="""
  var data = source.data;
  var f = cb_obj.value
  var x = data['x']
  var y = data['y']
  for (var i = 0; i < x.length; i++) {
      y[i] = Math.pow(x[i], f)
  }
  source.change.emit();
""")

#建立並給定部件名稱
slider = Slider(start=0.1, end=4, value=1, step=.1,
title="power")
slider.js_on_change('value', callback)
```

並且定義透過 slider 物件，來改變 callback 裡面的 value
而 CustomJS 觸發回調的模型，以 cb_obj 物件存取。

最後將我們的兩大物件
(plot 畫布 & slider 控制部件) 進行組裝

```
# 建立頁面框架
layout = column(slider, plot)

# 結果呈現
show(layout)
```



- 另一個常見的情況是，當選擇更改時，希望指定要執行的相同類型的回調。作為簡單的演示，下面的示例僅將第一個圖上的選定點複製到第二個圖上。但是，可以採用類似的方式輕鬆構造更複雜的動作和計算。

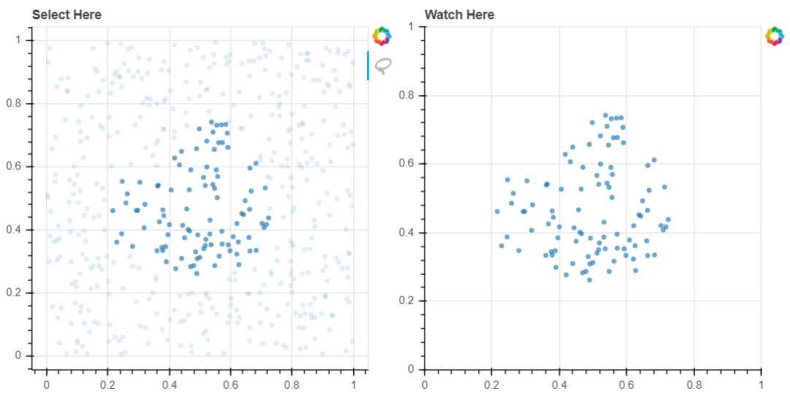
設定 datasource：

- s1 = ColumnDataSource(data=dict(x=x, y=y))

s1.selected.js_on_change('indices', CustomJS(args=dict(s1=s1, s2=s2).....)

- s2 = ColumnDataSource(data=dict(x=[], y=[]))

- 建立互動執行



邊緣和節點渲染器 GraphRenderer

關鍵功能是它為圖節點和圖邊緣維護單獨的 sub-GlyphRenderer。

- 這樣可以通過修改 GraphRenderer 的 node_renderer 屬性來自定義節點。可以將

- 同樣，可以通過 edge_renderer 屬性修改邊緣的樣式屬性。

交互策略

通過設置 GraphRenderer 的 selection_policy 和 inspection_policy 屬性，可以配置圖形的選擇或檢查行為。這些策略屬性接受特殊的 GraphHitTestPolicy 模型實例。

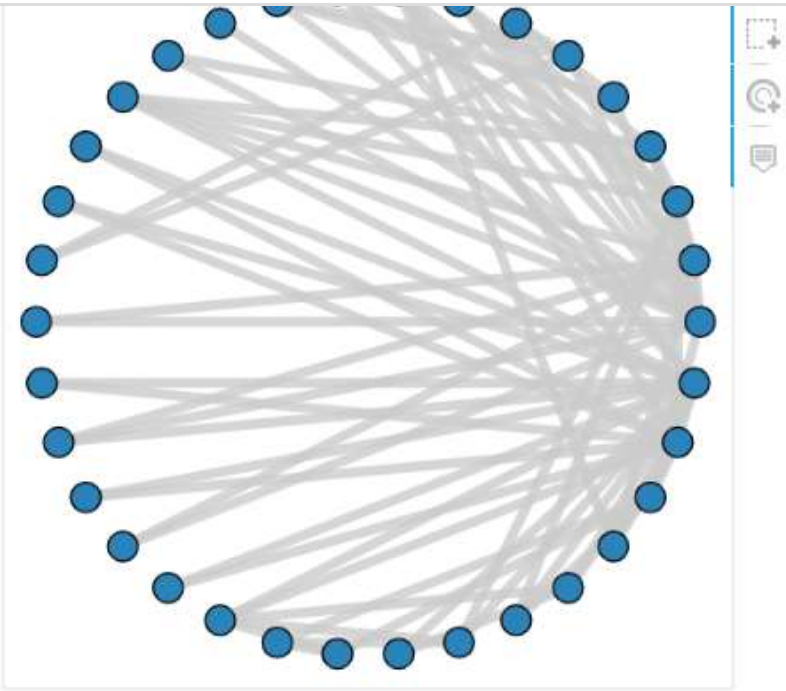
例如，設置 selection_policy = NodesAndLinkedEdges（）將使所選節點也選擇關聯的邊。類似地，設置 inspection_policy = EdgesAndLinkedNodes（）將導致在使用 HoverTool 懸停邊線時也檢查邊線的起點和終點。

用戶可能希望自定義邊緣和節點子渲染器的 selection_glyph，nonselection_glyph 和 / 或 hover_glyph 屬性，以便將動態視覺元素添加到其圖形交互中。

```
import networkx as nx

#建立互動網路圖

G=nx.karate_club_graph()
```

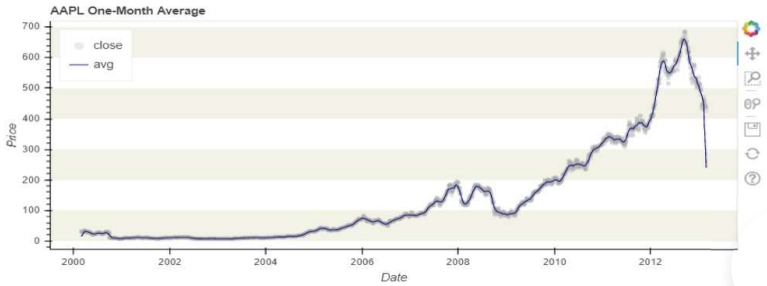
```
#載入相關的套件

import networkx as nx
```

```
from bokeh.models import (BoxSelectTool,
Circle, EdgesAndLinkedNodes,
HoverTool,MultiLine, NodesAndLinkedEdges,
Plot, Range1d, TapTool,)

from bokeh.palettes import Spectral4

from bokeh.plotting import from_networkx

---

#建立互動網路圖
G=nx.karate_club_graph()

#建立節點交互
graph_renderer.node_renderer.glyph
#建立邊緣回饋交互
graph_renderer.edge_renderer.glyph

#選擇策略
graph_renderer.selection_policy =
NodesAndLinkedEdges()
graph_renderer.inspection_policy =
EdgesAndLinkedNodes()
#繪製GRAPH
plot.renderers.append(graph_renderer)
#輸出
output_file("interactive_graphs.html")
```



知識點回顧

- Bokeh以成為交互式數據可視化的庫而自豪。
- 不同於Matplotlib和Seaborn等Python可視化領域的流行同行，Bokeh使用HTML和JavaScript渲染其圖形。這使其非常適合構建基於Web的儀表板和應用程序。但是，它是用於探索和理解數據或為項目或報告創建漂亮的自定義圖表的功能同樣強大的工具。
- 範例本身也提供一個股票線形圖

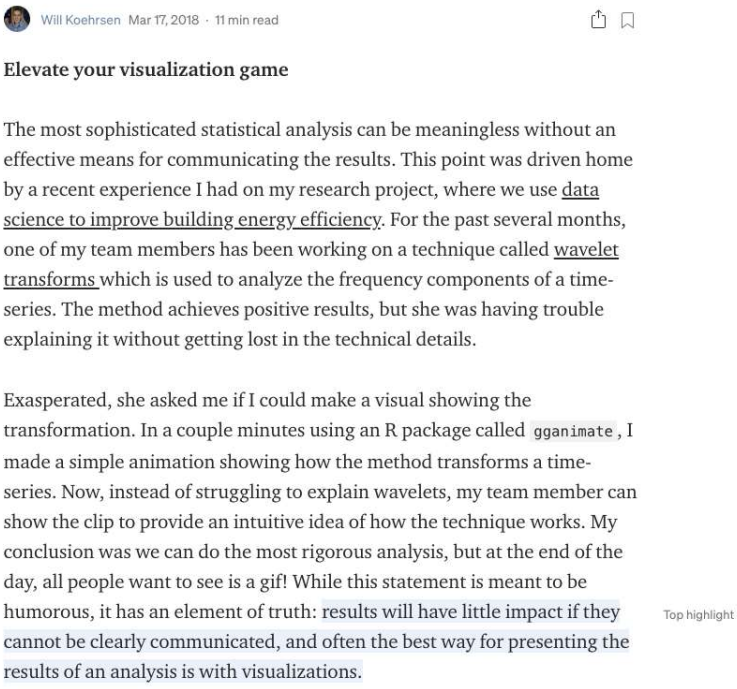


Bokeh In Python

Bokeh 的主要概念是，圖形一次只能建立一層。我們首先創建一個圖形，然後向該圖形添加稱為字形的元素。（對於使用 ggplot 的人來說，字形的思想與一次添加到一個“層”的圖形中的幾何圖形的思想相同。）字形可以根據所需的用途採用多種形狀：圓，線，補丁，條形，弧形等。讓我們通過製作帶有正方形和圓形的基本圖表來說明字形的想法。首先，我們使用 figure 方法繪製圖，然後通過調用適當的方法並傳入數據，將字形附加到圖上。最後，我們顯示我們的繪圖

Python 中使用 Bokeh 進行數據可視化，第一部分：入門

網站：towardsdatascience



Will Koehrsen Mar 17, 2018 · 11 min read

Elevate your visualization game

The most sophisticated statistical analysis can be meaningless without an effective means for communicating the results. This point was driven home by a recent experience I had on my research project, where we use data science to improve building energy efficiency. For the past several months, one of my team members has been working on a technique called wavelet transforms which is used to analyze the frequency components of a time-series. The method achieves positive results, but she was having trouble explaining it without getting lost in the technical details.

Exasperated, she asked me if I could make a visual showing the transformation. In a couple minutes using an R package called `gganimate`, I made a simple animation showing how the method transforms a time-series. Now, instead of struggling to explain wavelets, my team member can show the clip to provide an intuitive idea of how the technique works. My conclusion was we can do the most rigorous analysis, but at the end of the day, all people want to see is a gif! While this statement is meant to be humorous, it has an element of truth: results will have little impact if they cannot be clearly communicated, and often the best way for presenting the results of an analysis is with visualizations.

Top highlight

利用 Python 中 Bokeh 實現資料視覺化，第二部分：互動

網站：[itread01]

建立可互動的小部件

一旦我們在 Bokeh 中建立一個基礎圖形，通過小部件新增互動就相對簡單了。我們需要的第一個小部件是允許使用者選擇要顯示的航空公司的選擇框。這是一個允許根據需要進行儘可能多的選擇的複選框控制元件，在 Bokeh 中稱為T `CheckboxGroup.`。為了製作這個可選工具，我們需要匯入 `CheckboxGroup` 類來建立帶有兩個引數的例項，`labels`：我們希望顯示每個框旁邊的值以及 `active`：檢查選中的初始框。以下建立的 `CheckboxGroup` 程式碼中附有所需的運營商。

```
from bokeh.models.widgets import CheckboxGroup

# 建立複選框可選元素，可用的載體是
# 資料中所有航空公司組成的列表
carrier_selection = CheckboxGroup(labels=available_carriers,
                                  active = [0, 1])
```

複製程式碼

CheckboxGroup 部件

Bokeh 複選框中的標籤必須是字串，但啟用值需要的是整型。這意味著在在影象 'AirTran Airways Corporation' 中，啟用值為 0，而 'Alaska Airlines Inc.' 啟用值為 1。當我們想要將選中的複選框與 airlines 想匹配時，我們需要確保所選的**整型**啟用值能匹配與之對應的**字串**。我們可以使用部件的 `.labels` 和 `.active` 屬性來實現。

```
# 從選擇值中選擇航空公司的名稱
[carrier_selection.labels[i] for i in carrier_selection.active]

['AirTran Airways Corporation', 'Alaska Airlines Inc.']
```

複製程式碼

下一步：閱讀範例與完成作業