Bokeh is better than ever!

Fabio Pliger

fabio.pliger@gmail.com
@b_smoke

EuroPython 2016, Bilbao
Hi!

• Bokeh core dev
• Sw. Engineer @continuumio
• @b_smoke
• fabio.pliger@gmail.com
SUMMARY

• Introductions
• Gimme good stuff :)  
  • Bokeh Server
  • Bokeh Command
  • Client Side Callbacks
  • New Layout
  • Custom Extensions
  • Annotations
  • Geo Support
  • JS/Typescript API
  • JS Transforms
  • [More] WebGL support
• Extras
• What’s next?
Introduction

Show some hands
Introduction

What is Bokeh?

http://bokeh.pydata.org/
https://bokeh.github.io/blog/2016/6/28/release-0-12/
SUMMARY

• Introductions
• Gimme good stuff :)  
  • Bokeh Server
  • Bokeh Command
  • Client Side Callbacks
  • New Layout
  • Custom Extensions
  • Annotations
  • Geo Support
  • JS/Typescript API
  • JS Transforms
  • [More] WebGL support

• Extras
• What’s next?
Bokeh Server (a.k.a. Bokeh Apps)

- Completely rewritten since 0.11
- Powerful and performant
- Based on tornado and web sockets
- Integrated with bokeh command (*bokeh serve*)
- keep the “model objects” in python and in the browser in sync
  - respond to UI and tool events generated in a browser with computations or queries using python
  - automatically push updates the UI (i.e. widgets or plots), in a browser
  - use periodic, timeout, and asynchronous callbacks to drive streaming updates

See the new User’s Guide for much more info:

http://bokeh.pydata.org/en/latest/docs/user_guide/server.html#userguide-server-applications
Bokeh Server (a.k.a. Bokeh Apps)

https://demo.bokehplots.com/
Bokeh Server (a.k.a. Bokeh Apps)

• Running bokeh server:
  • to serve bokeh applications:
    ‣ `bokeh serve app_dir|app_script.py [options]`
  • for output_server or bokeh.client connections:
    ‣ `bokeh serve`

See the new User’s Guide for much more info:

http://bokeh.pydata.org/en/latest/docs/user_guide/cli.html#module-bokeh.command.subcommands.serve
from bokeh.plotting import figure, curdoc

# create a plot and style its properties
p = figure(x_range=(0, 100),
     y_range=(0, 100),
     toolbar_location=None)
p.border_fill_color = 'black'
p.background_fill_color = 'black'
p.outline_line_color = None
p.grid.grid_line_color = None

# add a text renderer to our plot (no data yet)
r = p.text(x=[], y=[], text=[], text_color=[],
     text_font_size="20pt",
     text_baseline="middle",
     text_align="center")

i = 0
rnd = np.random.random
ds = r.data_source
def callback():
    global i
    ds.data['x'].append(rnd()*70 + 15)
    ds.data['y'].append(rnd()*70 + 15)
    ds.data['text_color'].append(RdYlBu3[i%3])
    ds.data['text'].append(str(i))
    ds.trigger('data', ds.data, ds.data)
i = i + 1

# add a text button widget and configure with the # call back
button = Button(label="Press Me")
button.on_click(callback)

curvoc().add_root(column(button, p))
Bokeh Server (Bokeh App - Directory Format)

- A directory with at least a `main.py` file can be used.
- Similar to a single module format but functionality extended to:
  - A `server_lifecycle.py` file that allows optional callbacks to be triggered at different stages of application creation, as described in Lifecycle Hooks.
  - A `static` subdirectory that can be used to serve static resources associated with this application.
  - A `theme.yaml` file that declaratively defines default attributes to be applied to Bokeh model types.
  - A templates subdirectory with `index.html` Jinja template file. The directory may contain additional Jinja templates for `index.html` to refer to. The template should have the same parameters as the FILE template.
SUMMARY

• Introductions
• Gimme good stuff :)  
  • Bokeh Server  
  • **Bokeh Command**  
  • Client Side Callbacks  
  • New Layout  
  • Custom Extensions  
  • Annotations  
  • Geo Support  
  • JS/Typescript API  
  • JS Transforms  
  • [More] WebGL support  
• Extras  
• What’s next?
Bokeh Command

The bokeh html command can create standalone HTML documents from any kind of Bokeh application source: e.g., python scripts, app directories, JSON files, jupyter notebooks and others. For example:

bokeh html myapp.py

The bokeh json command will generate a serialized JSON representation of a Bokeh document from any kind of Bokeh application source. For example:

bokeh json myapp.py

The bokeh serve command let’s you instantly turn Bokeh documents into interactive web applications. For example:

bokeh serve myapp.py

See the new User’s Guide for much more info:

SUMMARY

• Introductions
• Gimme good stuff :)  
  • Bokeh Server  
  • Bokeh Command  
  • Client Side Callbacks  
  • New Layout  
  • Custom Extensions  
  • Annotations  
  • Geo Support  
  • JS/Typescript API  
  • JS Transforms  
  • [More] WebGL support  

• Extras  
• What’s next?
Static Documents, Dynamic Interaction

http://nbviewer.jupyter.org/github/bokeh/bokeh-notebooks/blob/master/tutorial/00%20-%20intro.ipynb#Interaction
JS Callbacks extend capability now, for a marginal cost (a few lines of JS)

[Will] encapsulate common patterns as they are discovered (no more JS!)

A dream: write callbacks in Python, translate automatically !!!

- no python code is ever executed when a CustomJS callback is used
- A CustomJS callback is only executed inside a browser JavaScript interpreter, and can only directly interact JavaScript data and functions (e.g., BokehJS Backbone models).

See the new User’s Guide for much more info:

Client Side “Python” Callbacks

```python
plot.line('x', 'y', source=source, line_width=3, line_alpha=0.6)

def callback(source=source, window=None):
    data = source.get('data')
    A, B = amp.get('value'), offset.get('value')
    k, phi = freq.get('value'), phase.get('value')
    x, y = data['x'], data['y']
    for i in range(len(x)):
        y[i] = B + A * window.Math.sin(k * x[i] + phi)
    source.trigger('change')

# turn our function into a CustomJS object.
# print(callback.code) to see the generated JavaScript
callback = CustomJS.from_py_func(callback)

amp_slider = Slider(start=0.1, end=10, value=1, step=.1,
                    title="Amplitude", callback=callback)

callback.args["amp"] = amp_slider
```
widgets in the GUI can trigger python callback functions that execute in the Jupyter Python kernel
• these callbacks call `push_notebook()` to push updates
• It is currently possible to push updates from python, to BokehJS (i.e., to update plots, etc.) using `push_notebook()` but not the opposite
SUMMARY

• Introductions
• Gimme good stuff :)  
  • Bokeh Server
  • Bokeh Command
  • Client Side Callbacks
• New Layout \o/
• Custom Extensions
• Annotations
• Geo Support
• JS/Typescript API
• JS Transforms
• [More] WebGL support
• Extras
• What’s next?
New Layout
• Bye bye VBox and HBox
• Welcome **Row, Column, and WidgetBox \o/**
  • `row()`, `column()`, and `widgetbox()`
  • let you build a grid of plots and widgets (rows, columns, and plots)
  • support a number of “sizing modes”. These sizing modes allow plots and widgets to resize based on the browser window (all items must have the same sizing mode & Widgets should be inside a widget box)

See the new User’s Guide for much more info:

Layouts (column)

```
output_file("layout.html")

x = list(range(11))
y0 = x
y1 = [10 - i for i in x]
y2 = [abs(i - 5) for i in x]

# create a new plot
s1 = figure(width=250, plot_height=250, title=None)
s1.circle(x, y0, size=10, color="navy", alpha=0.5)

# create another one
s2 = figure(width=250, height=250, title=None)
s2.triangle(x, y1, size=10, color="firebrick", alpha=0.5)

# create another
s3 = figure(width=250, height=250, title=None)
s3.square(x, y2, size=10, color="olive", alpha=0.5)

# put the results in a column and show
show(column(s1, s2, s3))
```
output_file("layout.html")

x = list(range(11))
y0 = x
y1 = [10 - i for i in x]
y2 = [abs(i - 5) for i in x]

# create a new plot
s1 = figure(width=250, plot_height=250, title=None)
s1.circle(x, y0, size=10, color="navy", alpha=0.5)

# create another one
s2 = figure(width=250, height=250, title=None)
s2.triangle(x, y1, size=10, color="firebrick", alpha=0.5)

# create and another
s3 = figure(width=250, height=250, title=None)
s3.square(x, y2, size=10, color="olive", alpha=0.5)

# put the results in a column and show
show(row(s1, s2, s3))
from bokeh.layouts import gridplot

output_file("layout_grid.html")

x = list(range(11))
y0 = x
y1 = [10 - i for i in x]
y2 = [abs(i - 5) for i in x]

# create three plots
p1 = figure(width=250, plot_height=250, title=None)
p1.circle(x, y0, size=10, color=Viridis3[0])
p2 = figure(width=250, height=250, title=None)
p2.triangle(x, y1, size=10, color=Viridis3[1])
p3 = figure(width=250, height=250, title=None)
p3.square(x, y2, size=10, color=Viridis3[2])

# make a grid
grid = gridplot([[p1, p2], [None, p3]])

# show the results
show(grid)
The full code for this plot is available at examples/howto/layouts/dashboard.py in the project GitHub repository.
Layout (examples)

https://bokeh.github.io/blog/2016/6/28/release-0-12/

https://demo.bokehplots.com/apps/movies

https://demo.bokehplots.com/apps/stocks

https://demo.bokehplots.com/apps/selection_histogram
SUMMARY

• Introductions
• Gimme good stuff :)  
  • Bokeh Server
  • Bokeh Command
  • Client Side Callbacks
  • New Layout
  • Custom Extensions
• Annotations
• Geo Support
• JS/Typescript API
• JS Transforms
• [More] WebGL support
• Extras
• What’s next?
Custom Extensions

- It is possible to extend Bokeh by creating custom user extensions to:
  - Modify the behaviour of existing Bokeh models
  - Add new models to connect third-party JavaScript libraries to Python
  - Create highly specialized models for domain specific use-cases
  - do not require setting up a development environment or building anything from source

See the new User's Guide for much more info:

Custom Extensions (Python Models)

- Python Models
  - For the most part, completely declarative classes
  - Custom extensions are created by making a subclass Model and including special class attributes to declare the properties that are mirrored on the JavaScript side

```python
from bokeh.core.properties import String, Instance
from bokeh.models import LayoutDOM, Slider

class Custom(LayoutDOM):
    text = String(default="Custom text")
    range = Instance(Slider)
```
Custom Extensions (JavaScript Models and Views)

- JavaScript side requires code to implement the model
- When appropriate, code for a corresponding view must also be provided.
- Currently BokehJS models and views are subclasses of Models and View from the Backbone JavaScript library.

```python
from bokeh.core.properties import String, Instance
from bokeh.models import LayoutDOM, Slider

class Custom(LayoutDOM):
    __implementation__ = open("custom.coffee").read()
    text = String(default="Custom text")
    slider = Instance(Slider)
```
SUMMARY

• Introductions
• Gimme good stuff :)  
  • Bokeh Server  
  • Bokeh Command  
  • Client Side Callbacks  
  • New Layout  
  • Custom Extensions  
• Annotations
• Geo Support  
• JS/Typescript API  
• JS Transforms  
• [More] WebGL support  

• Extras  
• What’s next?
Annotations

allow users to add supplemental information to their visualizations

Title and Legends are annotations now!

See the new User’s Guide for much more info

http://bokeh.pydata.org/en/latest/docs/user_guide/plotting.html#adding-annotations
from bokeh.models import Arrow, OpenHead, NormalHead, VeeHead

p = figure(plot_width=600, plot_height=600)

p.circle(x=[0, 1, 0.5], y=[0, 0, 0.7], radius=0.1, color=['navy', 'yellow', 'red'], fill_alpha=0.1)
p.add_layout(Arrow(end=OpenHead(line_color='firebrick', line_width=4), x_start=0, y_start=0, x_end=1, y_end=0))
p.add_layout(Arrow(end=NormalHead(fill_color='orange'), x_start=1, y_start=0, x_end=0.5, y_end=0.7))
p.add_layout(Arrow(end=VeeHead(size=35), line_color='red', x_start=0.5, y_start=0.7, x_end=0, y_end=0))
show(p)
from bokeh.models import BoxAnnotation
...

data = data.ix['2010-10-06': '2010-10-13']

p = figure(x_axis_type="datetime", tools=TOOLS)
p.line(data.index.to_series(), data['glucose'], line_color="gray", line_width=1, legend="glucose")

low_box = BoxAnnotation(top=80, fill_alpha=0.1, fill_color='red')
mid_box = BoxAnnotation(bottom=80, top=180, fill_alpha=0.1, fill_color='green')
high_box = BoxAnnotation(bottom=180, fill_alpha=0.1, fill_color='red')

p.add_layout(low_box)
p.add_layout(mid_box)
p.add_layout(high_box)

show(p)
from bokeh.models import ColumnDataSource, Range1d, LabelSet, Label

...  
p = figure(title='Dist. of 10th Grade Students at Lee High',
            x_range=Range1d(140, 275))
p.scatter(x='weight', y='height', size=8, source=source)
p.xaxis[0].axis_label = 'Weight (lbs)'
p.yaxis[0].axis_label = 'Height (in)'

labels = LabelSet(x='weight', y='height', text='names',
                    level='glyph', x_offset=5, y_offset=5, source=source,
                    render_mode='canvas')

citation = Label(x=70, y=70, x_units='screen', y_units='screen',
                    text='Collected by Luke C. 2016-04-01', render_mode='css',
                    border_line_color='black', border_line_alpha=1.0,
                    background_fill_color='white', background_fill_alpha=1.0)

p.add_layout(labels)
p.add_layout(citation)

show(p)
from bokeh.models import Span

p = figure(x_axis_type="datetime", y_axis_type="datetime")

p.line(daylight_warsaw_2013.Date, daylight_warsaw_2013.Sunset,
       line_dash='solid', line_width=2, legend="Sunset")
p.line(daylight_warsaw_2013.Date, daylight_warsaw_2013.Sunrise,
       line_dash='dotted', line_width=2, legend="Sunrise")

...  
daylight_savings_start = Span(location=start_date,
                               dimension='height', line_color='green',
                               line_dash='dashed', line_width=3)
p.add_layout(daylight_savings_start)

daylight_savings_end = Span(
    location=end_date, dimension='height',
    line_color='red', line_dash='dashed', line_width=3)
p.add_layout(daylight_savings_end)

show(p)
SUMMARY

• Introductions
• Gimme good stuff :)  
  • Bokeh Server
  • Bokeh Command
  • Client Side Callbacks
  • New Layout
  • Custom Extensions
  • Annotations
  • Geo Support
  • JS/Typescript API
  • JS Transforms
  • [More] WebGL support
• Extras
• What’s next?
Geo Support (GeoJSON Datasource)

from bokeh.models import GeoJSONDataSource
from bokeh.plotting import figure
from bokeh.sampledata.sample_geojson import geojson

geo_source = GeoJSONDataSource(geojson=geojson)

p = figure()
p.circle(x='x', y='y', alpha=0.9, source=geo_source)
output_file("geojson.html")
show(p)

{
   "type": "Feature",
   "geometry": {
      "type": "Point",
      "coordinates": [125.6, 10.1]
   },
   "properties": {
      "name": "Dinagat Islands"
   }
}
Warning :(  
There is an open issue documenting points appearing to be ~10px off from their intended location.  
Google has its own terms of service for using Google Maps API and any use of Bokeh with Google Maps must be within Google’s Terms of Service
Geo Support (Tile Providers)

Bokeh plots can also consume XYZ tile services which use the Web Mercator projection. The module bokeh.tile_providers contains several pre-configured tile sources with appropriate attribution which can be added to a plot using the .add_tile() method.

```python
from bokeh.io import output_file, show
from bokeh.plotting import figure
from bokeh.tile_providers import STAMEN_TONER

bound = 20000000 # meters
fig = figure(tools='pan, wheel_zoom', x_range=(-bound, bound), y_range=(-bound, bound))
fig.axis.visible = False
fig.add_tile(STAMEN_TONER)
output_file("stamen_toner_plot.html")
show(fig)
```
SUMMARY

• Introductions
• Gimme good stuff :)
  • Bokeh Server
  • Bokeh Command
  • Client Side Callbacks
  • New Layout
  • Custom Extensions
  • Annotations
  • Geo Support
  • JS/Typescript API
  • JS Transforms
  • [More] WebGL support
• Extras
• What’s next?
Bokeh JS

• BokehJS now has its own [real] API
• The BokehJS APIs are new as of version 0.12 and may undergo some changes before a 1.0 release.
• available via CDN and npm
• Low Level Models
• Plotting interface
• Charts interface**
  • bar
  • pie

See the new User’s Guide for much more info:

Bokeh JS

```javascript
var plt = Bokeh.Plotting;

var pie_data = {
    labels: ['Work', 'Eat', 'Commute', 'Sport', 'Watch TV', 'Sleep'],
    values: [8, 2, 2, 4, 0, 8],
};

var p2 = Bokeh.Charts.pie(pie_data, {
    inner_radius: 0.2,
    start_angle: Math.PI / 2
});

var p3 = Bokeh.Charts.pie(pie_data, {
    inner_radius: 0.2,
    start_angle: Math.PI / 6,
    end_angle: 5 * Math.PI / 6
});

var p4 = Bokeh.Charts.pie(pie_data, {
    inner_radius: 0.2,
    palette: "Oranges9",
    slice_labels: "percentages"
});

plt.show(plt.gridplot([p2, p3, p4]));
```
Bokeh JS

```javascript
var plt = Bokeh.Plotting;

var bar_data = [
    ['City', '2010 Population', '2000 Population'],
    ['New York City, NY', 8175000, 8008000],
    ['Los Angeles, CA', 3792000, 3694000],
    ['Chicago, IL', 2695000, 2896000],
    ['Houston, TX', 2099000, 1953000],
    ['Philadelphia, PA', 1526000, 1517000],
];

var p1 = Bokeh.Charts.bar(bar_data, {
    axis_number_format: "0.[00]a"
});

var p2 = Bokeh.Charts.bar(bar_data, {
    axis_number_format: "0.[00]a",
    stacked: true
});

var p3 = Bokeh.Charts.bar(bar_data, {
    axis_number_format: "0.[00]a",
    orientation: "vertical"
});

var p4 = Bokeh.Charts.bar(bar_data, {
    axis_number_format: "0.[00]a",
    orientation: "vertical",
    stacked: true
});

plt.show(plt.gridplot([p1, p2, p3, p4]));
```
SUMMARY

• Introductions
• Gimme good stuff :)
  • Bokeh Server
  • Bokeh Command
  • Client Side Callbacks
  • New Layout
  • Custom Extensions
  • Annotations
  • Geo Support
  • JS/Typescript API
  • JS Transforms
  • [More] WebGL support
• Extras
• What’s next?
Bokeh JS

```python
import numpy as np
from bokeh.models import Jitter
from bokeh.plotting import figure, show, output_file

p = figure(plot_width=500, plot_height=400,
    x_range=(0,3), y_range=(0,10),
    title="Demonstration of Jitter transform")

y1 = np.random.random(2500) * 10
y2 = np.random.normal(size=2500)*2 + 5

p.circle(x={'value': 1, 'transform': Jitter(width=0.4)}, y=y1, color="navy", alpha=0.3)

p.circle(x={'value': 2, 'transform': Jitter(width=0.4)}, y=y2, color="firebrick", alpha=0.3)

output_file("jitter.html")
show(p)
```
SUMMARY

• Introductions
• Gimme good stuff :)  
  • Bokeh Server
  • Bokeh Command
  • Client Side Callbacks
  • New Layout
  • Custom Extensions
  • Annotations
  • Geo Support
  • JS/Typescript API
  • JS Transforms
    • [More] WebGL support
• Extras
• What’s next?
WebGL [Better] Support

- support to all markers
- fixed WebGL bugs
- fast!

See the new User's Guide for much more info:

SUMMARY

• Introductions
• Gimme good stuff :)  
  • Bokeh Server
  • Bokeh Command
  • Client Side Callbacks
  • New Layout
  • Custom Extensions
  • Annotations
  • Geo Support
  • JS/Typescript API
  • JS Transforms
  • [More] WebGL support

• Extras

• What’s next?
HoloViews

```python
import numpy as np
import holoviews as hv
hv.notebook_extension('matplotlib')
fractal = hv.Image(np.load('mandelbrot.npy'))

((fractal * hv.HLine(y=0)).hist() + fractal.sample(y=0))
```

HoloViewsJS, MatplotlibJS successfully loaded in this cell.

more info at [http://holoviews.org/](http://holoviews.org/)
DataShader

http://datashader.readthedocs.io/
SUMMARY

- Introductions
- Gimme good stuff :)  
  - Bokeh Server
  - Bokeh Command
  - Client Side Callbacks
  - New Layout
  - Custom Extensions
  - Annotations
  - Geo Support
  - JS/Typescript API
  - JS Transforms
  - [More] WebGL support
- Extras
- What’s next?
1.0!
THANK YOU!