

---

# **python-nvd3 Documentation**

*Release 0.12.2*

**Arezqui Belaid**

July 23, 2014



<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Overview . . . . .	3
1.2	Documentation . . . . .	4
1.3	Contributing . . . . .	4
1.4	License . . . . .	4
<b>2</b>	<b>Resources</b>	<b>5</b>
2.1	Feedback . . . . .	5
2.2	Source download . . . . .	5
<b>3</b>	<b>Chart Classes Reference</b>	<b>7</b>
3.1	NVD3Chart . . . . .	7
<b>4</b>	<b>Examples of chart types</b>	<b>11</b>
4.1	cumulativeLineChart . . . . .	11
4.2	discreteBarChart . . . . .	11
4.3	lineChart . . . . .	12
4.4	lineWithFocusChart . . . . .	12
4.5	linePlusBarChart . . . . .	13
4.6	linePlusBarWithFocusChart . . . . .	13
4.7	multiBarChart . . . . .	14
4.8	multiBarHorizontalChart . . . . .	14
4.9	pieChart . . . . .	15
4.10	scatterChart . . . . .	15
4.11	stackedAreaChart . . . . .	16
<b>5</b>	<b>Indices and tables</b>	<b>17</b>
	<b>Python Module Index</b>	<b>19</b>



**Release** 0.12.2

**Date** July 23, 2014

**Keywords** python, plot, graph, nvd3, d3

**Author** Arezqui Belaid

**Description** Python wrapper for nvd3, build re-usable charts and chart components for d3.js

NVD3 is an attempt to build re-usable charts and chart components for d3.js without taking away the power that d3.js offers you.

Python-NVD3 makes your life easy! You write Python and the library renders JavaScript for you! These graphs can be part of your web application:

Want to try it yourself? Install python-nvd3, enter your python shell and try this quick demo:

```
>>> from nvd3 import pieChart
>>> type = 'pieChart'
>>> chart = pieChart(name=type, color_category='category20c', height=450, width=450)
>>> xdata = ["Orange", "Banana", "Pear", "Kiwi", "Apple", "Strawberry", "Pineapple"]
>>> ydata = [3, 4, 0, 1, 5, 7, 3]
>>> extra_serie = {"tooltip": {"y_start": "", "y_end": " cal"}}
>>> chart.add_serie(y=ydata, x=xdata, extra=extra_serie)
>>> chart.buildcontent()
>>> print chart.htmlcontent
```

This will output the following HTML to render a live chart. The HTML could be stored into a HTML file, used in a Web application, or even used via Ipython Notebook:

```
<div id="pieChart"><svg style="width:450px;height:450px;"></svg></div>
<script>
data_pieChart=[{"values": [{"value": 3, "label": "Orange"},
                        {"value": 4, "label": "Banana"},
                        {"value": 0, "label": "Pear"},
                        {"value": 1, "label": "Kiwi"},
                        {"value": 5, "label": "Apple"},
                        {"value": 7, "label": "Strawberry"},
                        {"value": 3, "label": "Pineapple"}], "key": "Serie 1"}];

nv.addGraph(function() {
  var chart = nv.models.pieChart();
  chart.margin({top: 30, right: 60, bottom: 20, left: 60});
  var datum = data_pieChart[0].values;
  chart.tooltipContent(function(key, y, e, graph) {
    var x = String(key);
    var y = String(y) + ' cal';
    tooltip_str = '<center><b>'+x+'</b></center>' + y;
    return tooltip_str;
  });
  chart.showLegend(true);
  chart.showLabels(true);
  chart.donut(false);
  chart
    .x(function(d) { return d.label })
    .y(function(d) { return d.value });
  chart.width(450);
  chart.height(450);
  d3.select('#pieChart svg')
```

```
.datum(datum)
.transition().duration(500)
.attr('width', 450)
.attr('height', 450)
.call(chart);
});
</script>
```

Check out the class references for dynamic examples and a full list of supported charts!

Excited !? Learn more here:

---

## Introduction

---

**Version** 0.12.2

**Date** July 23, 2014

**Keywords** python, plot, graph, nvd3, d3

**Author** Arezqui Belaid

**License** MIT

**Description** Python wrapper for nvd3, build re-usable charts and chart components for d3.js

**NVD3** NVD3 <http://nvd3.org/>

– Python-nvd3 is a Python wrapper for NVD3 graph library. NVD3 is an attempt to build re-usable charts and chart components for d3.js without taking away the power that d3.js gives you.

## 1.1 Overview

Python-nvd3 is a Python wrapper for NVD3 graph library. Visit NVD3 website for further information : <http://nvd3.org/>

### 1.1.1 Installation

Install, upgrade and uninstall python-nvd3.py with these commands:

```
$ sudo pip install python-nvd3
$ sudo pip install --upgrade python-nvd3
$ sudo pip uninstall python-nvd3
```

Or if you don't have *pip*:

```
$ sudo easy_install python-nvd3
```

### 1.1.2 Usage

After installation use python-nvd3 as follows

```
from nvd3 import pieChart

#Open File to write the D3 Graph
output_file = open('test-nvd3.html', 'w')
```

```
type = 'pieChart'
chart = pieChart(name=type, color_category='category20c', height=450, width=450)
chart.set_containerheader("\n\n<h2>" + type + "</h2>\n\n")

xdata = ["Orange", "Banana", "Pear", "Kiwi", "Apple", "Strawberry", "Pineapple"]
ydata = [3, 4, 0, 1, 5, 7, 3]

extra_serie = {"tooltip": {"y_start": "", "y_end": " cal"}}
chart.add_serie(y=ydata, x=xdata, extra=extra_serie)
chart.buildhtml()
output_file.write(chart.htmlcontent)

#close Html file
output_file.close()
```

See our examples directory for more usage.

### 1.1.3 Supported nvd3 charts

See the section Chart Classes.

## 1.2 Documentation

Check out the documentation on 'Read the Docs' (<http://python-nvd3.readthedocs.org>) for some live Chart examples!

### 1.2.1 Changelog

Changelog summary : <https://github.com/areski/python-nvd3/blob/master/CHANGELOG.rst>

## 1.3 Contributing

If you've found a bug, add a feature or improve python-nvd3 and think it is useful then please consider contributing. Patches, pull requests or just suggestions are always welcome!

Source code: <http://github.com/areski/python-nvd3>

If you don't like Github and Git you're welcome to send regular patches.

Bug tracker: <http://github.com/areski/python-nvd3/issues>

## 1.4 License

Python-nvd3 is licensed under MIT, see *MIT-LICENSE.txt*.



---

## Resources

---

- [Feedback](#)
- [Source download](#)

### 2.1 Feedback

Your feedback is more than welcome. Write email to [areski@gmail.com](mailto:areski@gmail.com) or post bugs and feature requests on github:

<http://github.com/areski/python-nvd3/issues>

### 2.2 Source download

The source code is currently available on github. Fork away!

<http://github.com/areski/python-nvd3>



---

## Chart Classes Reference

---

Contents:

### 3.1 NVD3Chart

**class** `nvd3.NVD3Chart.NVD3Chart` (\*\*kwargs)  
NVD3Chart Base class

**Attributes:**

- `axislist` - All X, Y axis list
- `charttooltip_dateformat` - date format for tooltip if x-axis is in date format
- `charttooltip` - Custom tooltip string
- `color_category` - Define color category (eg. `category10`, `category20`, `category20c`)
- `color_list` - used by pieChart (eg. ['red', 'blue', 'orange'])
- `container` - Place for graph
- `containerheader` - Header for javascript code
- `count` - chart count
- `custom_tooltip_flag` - False / True
- `date_flag` - x-axis contain date format or not
- `dateformat` - see <https://github.com/mbostock/d3/wiki/Time-Formatting>
- `extras` - extra modifiers. Use this to modify different attributes of the chart.
- `header_css` - False / True
- `header_js` - Custom tooltip string
- `height` - Set graph height
- `htmlcontent` - Contain the htmloutput
- `htmlheader` - Contain the html header
- `jschart` - Javascript code as string
- `margin_bottom` - set the bottom margin
- `margin_left` - set the left margin

- `margin_right` - set the right margin
- `margin_top` - set the top margin
- `model` - set the model (ex. `pieChart`, `LineWithFocusChart`, `MultiBarChart`)
- `resize` - `False / True`
- `series` - Series are list of data that will be plotted
- `stacked` - `False / True`
- `style` - Special style
- `template_page_nvd3` - template variable
- `use_interactive_guideline` - `False / True`
- `width` - Set graph width
- `x_axis_date` - `False / True`
- `show_legend` - `False / True`
- `show_labels` - `False / True`
- `assets_directory` directory holding the assets (`./bower_components/`)
- `x_custom_format` - `False / True` - Used with `x_axis_format`
- `y_custom_format` - `False / True`- Used with `y_axis_format`

**add\_chart\_extras** (*extras*)

Use this method to add extra d3 properties to your chart. For example, you want to change the text color of the graph:

```
chart = pieChart(name='pieChart', color_category='category20c', height=400, width=400)

xdata = ["Orange", "Banana", "Pear", "Kiwi", "Apple", "Strawberry", "Pineapple"]
ydata = [3, 4, 0, 1, 5, 7, 3]

extra_serie = {"tooltip": {"y_start": "", "y_end": " cal"}}
chart.add_serie(y=ydata, x=xdata, extra=extra_serie)
```

The above code will create graph with a black text, the following will change it:

```
text_white="d3.selectAll('#pieChart text').style('fill', 'white');"
chart.add_chart_extras(text_white)
```

The above extras will be appended to the java script generated.

Alternatively, you can use the following initialization:

```
chart = pieChart(name='pieChart',
                 color_category='category20c',
                 height=400, width=400,
                 extras=text_white)
```

**add\_serie** (*y, x, name=None, extra={}, \*\*kwargs*)

add serie - Series are list of data that will be plotted y {1, 2, 3, 4, 5} / x {1, 2, 3, 4, 5}

**Attributes:**

- `name` - set Serie name
- `x` - x-axis data

- y - y-axis data

kwargs:

- shape - for scatterChart, you can set different shapes (circle, triangle etc...)
- size - for scatterChart, you can set size of different shapes
- type - for multiChart, type should be bar
- bar - to display bars in Chart
- color\_list - define list of colors which will be used by pieChart
- color - set axis color
- disabled -

extra:

- tooltip - set tooltip flag
- date\_format - set date\_format for tooltip if x-axis is in date format

**buildcontainer ()**

generate HTML div

**buildcontent ()**

Build HTML content only, no header or body tags. To be useful this will usually require the attribute *jquery\_on\_ready* to be set which will wrap the js in `$(function(){<regular_js>;})`

**buildhtml ()**

Build the HTML page Create the htmlheader with css / js Create html page Add Js code for nvd3

**buildhtmlheader ()**

generate HTML header content

**buildjschart ()**

generate javascript code for the chart

**create\_x\_axis (name, label=None, format=None, date=False, custom\_format=False)**

Create X-axis

**create\_y\_axis (name, label=None, format=None, custom\_format=False)**

Create Y-axis

**set\_containerheader (containerheader)**

Set containerheader

**set\_custom\_tooltip\_flag (custom\_tooltip\_flag)**

Set custom\_tooltip\_flag & date\_flag

**set\_date\_flag (date\_flag=False)**

Set date flag

**set\_graph\_height (height)**

Set Graph height

**set\_graph\_width (width)**

Set Graph width

**slugify\_name (name)**

Slufigy name with underscore



---

## Examples of chart types

---

Contents:

### 4.1 cumulativeLineChart

**class** `nvd3.cumulativeLineChart` (*\*\*kwargs*)

A cumulative line chart is used when you have one important grouping representing an ordered set of data and one value to show, summed over time.

Python example:

```
from nvd3 import cumulativeLineChart
chart = cumulativeLineChart(name='cumulativeLineChart', x_is_date=True)
xdata = [1365026400000000, 1365026500000000, 1365026600000000]
ydata = [6, 5, 1]
y2data = [36, 55, 11]

extra_serie = {"tooltip": {"y_start": "There are ", "y_end": " calls"}}
chart.add_serie(name="Serie 1", y=ydata, x=xdata, extra=extra_serie)

extra_serie = {"tooltip": {"y_start": "", "y_end": " mins"}}
chart.add_serie(name="Serie 2", y=y2data, x=xdata, extra=extra_serie)
chart.buildhtml()
```

Javascript generated:

See the HTML source code of this page, to see the underlying javascript.

### 4.2 discreteBarChart

**class** `nvd3.discreteBarChart` (*\*\*kwargs*)

A discrete bar chart or bar graph is a chart with rectangular bars with lengths proportional to the values that they represent.

Python example:

```
from nvd3 import discreteBarChart
chart = discreteBarChart(name='discreteBarChart', height=400, width=400)

xdata = ["A", "B", "C", "D", "E", "F"]
ydata = [3, 4, 0, -3, 5, 7]
```

```
chart.add_series(y=ydata, x=xdata)
chart.buildhtml()
```

Javascript generated:

See the HTML source code of this page, to see the underlying javascript.

## 4.3 lineChart

**class** `nvd3.lineChart` (*\*\*kwargs*)

A line chart or line graph is a type of chart which displays information as a series of data points connected by straight line segments.

Python example:

```
from nvd3 import lineChart
chart = lineChart(name="lineChart", x_is_date=False, x_axis_format="AM_PM")

xdata = range(24)
ydata = [0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 4, 3, 3, 5, 7, 5, 3, 16, 6, 9, 15, 4, 12]
ydata2 = [9, 8, 11, 8, 3, 7, 10, 8, 6, 6, 9, 6, 5, 4, 3, 10, 0, 6, 3, 1, 0, 0, 0, 1]

extra_series = {"tooltip": {"y_start": "There are ", "y_end": " calls"}}
chart.add_series(y=ydata, x=xdata, name='sine', extra=extra_series, **kwargs1)
extra_series = {"tooltip": {"y_start": "", "y_end": " min"}}
chart.add_series(y=ydata2, x=xdata, name='cose', extra=extra_series, **kwargs2)
chart.buildhtml()
```

Javascript rendered to:

See the source code of this page, to see the underlying javascript.

See the HTML source code of this page, to see the underlying javascript.

## 4.4 lineWithFocusChart

**class** `nvd3.lineWithFocusChart` (*\*\*kwargs*)

A lineWithFocusChart or line graph is a type of chart which displays information as a series of data points connected by straight line segments. The lineWithFocusChart provide a smaller chart that act as a selector, this is very useful if you want to zoom on a specific time period.

Python example:

```
from nvd3 import lineWithFocusChart
chart = lineWithFocusChart(name='lineWithFocusChart', x_is_date=True, x_axis_format="%d %b %Y")
xdata = [13650264000000000, 13650265000000000, 13650266000000000]
ydata = [-6, 5, -1]

extra_series = {"tooltip": {"y_start": "", "y_end": " ext"},
                "date_format": "%d %b %Y"}
chart.add_series(name="Serie 1", y=ydata, x=xdata, extra=extra_series)
chart.buildhtml()
```

Javascript generated:

See the HTML source code of this page, to see the underlying javascript.



## 4.5 linePlusBarChart

**class** `nvd3.linePlusBarChart` (*\*\*kwargs*)

A `linePlusBarChart` Chart is a type of chart which displays information as a series of data points connected by straight line segments and with some series with rectangular bars with lengths proportional to the values that they represent.

Python example:

```
from nvd3 import linePlusBarChart
chart = linePlusBarChart(name="linePlusBarChart",
                          width=500, height=400, x_axis_format="%d %b %Y",
                          x_is_date=True,
                          yaxis2_format="function(d) { return d3.format(',0.3f')(d) }")

xdata = [1338501600000, 1345501600000, 1353501600000]
ydata = [6, 5, 1]
y2data = [0.002, 0.003, 0.004]

extra_serie = {"tooltip": {"y_start": "There are ", "y_end": " calls"},
               "date_format": "%d %b %Y %H:%S" }
chart.add_serie(name="Serie 1", y=ydata, x=xdata, extra=extra_serie,
                bar=True)

extra_serie = {"tooltip": {"y_start": "There are ", "y_end": " min"}}
chart.add_serie(name="Serie 2", y=y2data, x=xdata, extra=extra_serie)
chart.buildcontent()
```

Note that in case you have two data serie with extreme different numbers, that you would like to format in different ways, you can pass a keyword `yaxis1_format` or `yaxis2_format` when creating the graph.

In the example above the graph created presents the values of the second data series with three digits right of the decimal point.

Javascript generated:

See the HTML source code of this page, to see the underlying javascript.

## 4.6 linePlusBarWithFocusChart

**class** `nvd3.linePlusBarWithFocusChart` (*\*\*kwargs*)

A `linePlusBarWithFocusChart` Chart is a type of chart which displays information as a series of data points connected by straight line segments and with some series with rectangular bars with lengths proportional to the values that they represent

Python example:

```
from nvd3 import linePlusBarWithFocusChart
chart = linePlusBarWithFocusChart(name='linePlusBarChart', x_is_date=True, x_axis_format="%d %b %Y",
                                  width=500, height=400, yaxis2_format="function(d) { return d3.format(',0.3f')(d) }")

xdata = [1338501600000, 1345501600000, 1353501600000]
ydata = [6, 5, 1]
y2data = [36, 55, 11]
kwargs = {}
kwargs['bar'] = True
extra_serie = {"tooltip": {"y_start": "There is ", "y_end": " calls"}}
chart.add_serie(name="Serie 1", y=ydata, x=xdata, extra=extra_serie, **kwargs)
```

```
extra_serie = {"tooltip": {"y_start": "There is ", "y_end": " min"}}
chart.add_serie(name="Serie 2", y=y2data, x=xdata, extra=extra_serie)
chart.buildhtml()
```

Javascript generated:

See the HTML source code of this page, to see the underlying javascript.

## 4.7 multiBarChart

**class** `nvd3.multiBarChart` (*\*\*kwargs*)

A multiple bar graph contains comparisons of two or more categories or bars. One axis represents a quantity and the other axis identifies a specific feature about the categories. Reading a multiple bar graph includes looking at extremes (tallest/longest vs. shortest) in each grouping.

Python example:

```
from nvd3 import multiBarChart
chart = multiBarChart(width=500, height=400, x_axis_format=None)
xdata = ['one', 'two', 'three', 'four']
ydata1 = [6, 12, 9, 16]
ydata2 = [8, 14, 7, 11]

chart.add_serie(name="Serie 1", y=ydata1, x=xdata)
chart.add_serie(name="Serie 2", y=ydata2, x=xdata)
chart.buildhtml()
```

Javascript generated:

See the HTML source code of this page, to see the underlying javascript.

## 4.8 multiBarHorizontalChart

**class** `nvd3.multiBarHorizontalChart` (*\*\*kwargs*)

A multiple horizontal bar graph contains comparisons of two or more categories or bars.

Python example:

```
from nvd3 import multiBarHorizontalChart
chart = multiBarHorizontalChart(name='multiBarHorizontalChart', height=400, width=400)
xdata = [-14, -7, 7, 14]
ydata = [-6, 5, -1, 9]
y2data = [-23, -6, -32, 9]

extra_serie = {"tooltip": {"y_start": "", "y_end": " balls"}}
chart.add_serie(name="Serie 1", y=ydata, x=xdata, extra=extra_serie)

extra_serie = {"tooltip": {"y_start": "", "y_end": " calls"}}
chart.add_serie(name="Serie 2", y=y2data, x=xdata, extra=extra_serie)
chart.buildcontent()
```

Javascript generated:

See the HTML source code of this page, to see the underlying javascript.

## 4.9 pieChart

**class** `nvd3.pieChart` (*\*\*kwargs*)

A pie chart (or a circle graph) is a circular chart divided into sectors, illustrating numerical proportion. In chart, the arc length of each sector is proportional to the quantity it represents.

Python example:

```
from nvd3 import pieChart
chart = pieChart(name='pieChart', color_category='category20c', height=400, width=400)

xdata = ["Orange", "Banana", "Pear", "Kiwi", "Apple", "Strawberry", "Pineapple"]
ydata = [3, 4, 0, 1, 5, 7, 3]

extra_serie = {"tooltip": {"y_start": "", "y_end": " cal"}}
chart.add_serie(y=ydata, x=xdata, extra=extra_serie)
chart.buildhtml()
```

Javascript generated:

See the HTML source code of this page, to see the underlying javascript.

## 4.10 scatterChart

**class** `nvd3.scatterChart` (*\*\*kwargs*)

A scatter plot or scattergraph is a type of mathematical diagram using Cartesian coordinates to display values for two variables for a set of data. The data is displayed as a collection of points, each having the value of one variable determining the position on the horizontal axis and the value of the other variable determining the position on the vertical axis.

Python example:

```
from nvd3 import scatterChart
chart = scatterChart(name='scatterChart', height=400, width=400)
xdata = [3, 4, 0, -3, 5, 7]
ydata = [-1, 2, 3, 3, 15, 2]
ydata2 = [1, -2, 4, 7, -5, 3]

kwargs1 = {'shape': 'circle', 'size': '1'}
kwargs2 = {'shape': 'cross', 'size': '10'}

extra_serie = {"tooltip": {"y_start": "", "y_end": " call"}}
chart.add_serie(name="series 1", y=ydata, x=xdata, extra=extra_serie, **kwargs1)

extra_serie = {"tooltip": {"y_start": "", "y_end": " min"}}
chart.add_serie(name="series 2", y=ydata2, x=xdata, extra=extra_serie, **kwargs2)
chart.buildhtml()
```

Javascript generated:

See the HTML source code of this page, to see the underlying javascript.

## 4.11 stackedAreaChart

**class** `nvd3.stackedAreaChart` (*\*\*kwargs*)

The stacked area chart is identical to the area chart, except the areas are stacked on top of each other, rather than overlapping. This can make the chart much easier to read.

Python example:

```
from nvd3 import stackedAreaChart
chart = stackedAreaChart(name='stackedAreaChart', height=400, width=400)

xdata = [100, 101, 102, 103, 104, 105, 106,]
ydata = [6, 11, 12, 7, 11, 10, 11]
ydata2 = [8, 20, 16, 12, 20, 28, 28]

extra_serie = {"tooltip": {"y_start": "There is ", "y_end": " min"}}
chart.add_serie(name="Serie 1", y=ydata, x=xdata, extra=extra_serie)
chart.add_serie(name="Serie 2", y=ydata2, x=xdata, extra=extra_serie)
chart.buildhtml()
```

Javascript generated:

See the HTML source code of this page, to see the underlying javascript.

---

## Indices and tables

---

- *genindex*
- *modindex*
- *search*



## n

[nvd3](#), [12](#)  
[nvd3.cumulativeLineChart](#), [11](#)  
[nvd3.discreteBarChart](#), [11](#)  
[nvd3.lineChart](#), [12](#)  
[nvd3.linePlusBarWithFocusChart](#), [13](#)  
[nvd3.lineWithFocusChart](#), [12](#)  
[nvd3.multiBarChart](#), [14](#)  
[nvd3.multiBarHorizontalChart](#), [14](#)  
[nvd3.NVD3Chart](#), [7](#)  
[nvd3.pieChart](#), [14](#)  
[nvd3.scatterChart](#), [15](#)  
[nvd3.stackedAreaChart](#), [15](#)