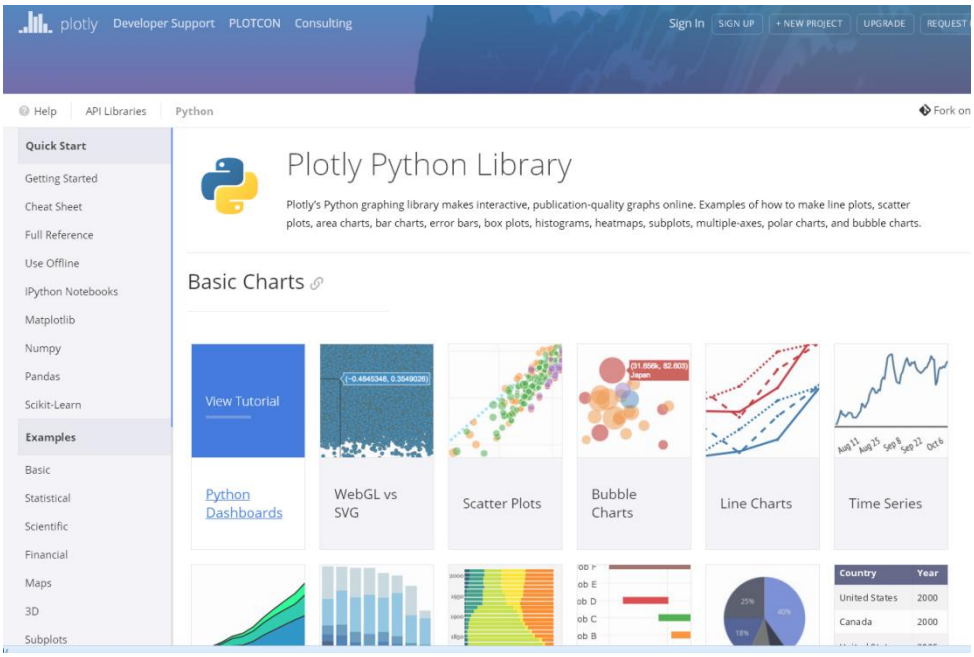
TopQuant.vip 极宽量化开源课件系列文库

《Plotly 绘图模块中文指南》第 1 期

作者：TopQuant 极宽量化开源团队

2016,12,25 圣诞节



- http://topquant.vip,Top 极宽量化社区（原 zw 量化,ziwang.com），国内第一家专业 Python 量化社区!
- 字王 Git 项目总览：github.com/ziwang-com/
- QQ 群：124134140 （千人大群，AI 量化，大数据、云字库、zwPython）
- 网盘下载：pan.baidu.com/s/1jIg944u

前言

尽管 plotly2016 年 6 月，才正式发布 python-api 文档，尽管 2017 年 1 月，plotly 1.0 才正式发布。

但 plotly，以其强度无比的功能，特别是强大的 web 交互式绘图功能，彻底替代了 matplotlib、seaborn、bokeh 等多种传统的 Python 每天模块，成为新一代的绘图模块当中的王中之王。

如今，很多人都听说了 plotly 的大名，很多人都渴望早日接触、早日使用 plotly 绘图包括。

不过，也有很少的一部分人，在默默 的工作，从 2016 年圣诞、2017 年元旦、2017 年春节。

这些很少的很少的一部分网友，默默地利用自己的节假日时间，默默地引进、翻译 plolty 的各种文档。

这些人，就是 TopQuant 极宽量化开源组的 plotly 翻译小组。

这些人是中国 Quant 的 Top 团队。

plolty 的文档翻译，可能是极宽量化开源团队，面对的最大一次挑战，时间方面，恰遇新年，文档规模，更是超过了前面的各种课件。

为此，极宽开源组只能采用折中的办法，分期进行翻译。

如今，第一期文档已经完成，中英双语，超过 240 页的 16 开版面，完全是一部图书的尺度。

何海群

TopQuant 极宽量化开源团队发起人

更多介绍请参见：

Plotly 互动型 python 绘图神器 - plotly 与数据可视化 – http://ziwang.com/forum.php?mod=viewthread&tid=41&extra=page%3D1

TopQuant-极宽量化社区（原 zw 量化），国内第一家专业 Python 量化社区

附录：任务分配

plotly 翻译任务分几期进行，这次为第一期，仅仅进行最简单的翻译任务。只翻译 getting_start, basic, 与 Financial Charts （ plotl offline 部分后来由 youngle sunny 1535327967 个人兴趣而临时加进去）这三部分内容。

任务分配如下：

组长：

youngle sunny 1535327967 basic 1,2 + 任务分配+ 校对，组长

成员：

余勤 441499022 校对 + 机动安排任务

leon, 华子（尹少华 32509167） geting_start 1/2

啦啦啦 505512828 geting_start 2/2

禎 948280670 basic 5,6

L. 1248515039 basic7,8

Rikimaru 11766429 basic 9,10

iris 704699640 13,14

信平 759949947 basic 3,4 + 校对

吴娜 2184934 basic20-2 + 校对 + 机动安排任务

周涛 510548099 11,12

zw木子 719735825 basic 15,16

非洲兔 85011284 basic 17,18

十二月 378258849 basic 19,20-1

大朱 775941748 Financial Charts 1

我爱作文你信吗 571171954 Financial Charts 2

补充其他：

Youngle sunny 1535327967 plot offline，该部分内容不在本次翻译计划范围之内，属于个人兴趣之作。

快速入门

Getting Started with Plotly for Python

Installation and Initialization Steps for Using Plotly in Python.

在 Python 中使用 Plotly 的安装和初始化步骤

安装

安装

To install Plotly's python package, use the package manager **pip** inside your terminal.

在终端上使用包管理器 pip 来安装 Plotly 的 python 包

If you don't have pip installed on your machine, click here for pip's installation instructions.

如果你的机器上没有安装过 pip，点击这里看安装说明。

```
$ pip install plotly
or
$ sudo pip install plotly
```

Plotly's Python package is updated frequently! To upgrade, run:

Plotly 的 Python 包会经常更新，若需要更新到最新版，运行：

```
$ pip install plotly --upgrade
```

在线初始化

Initialization for Online Plotting

在线初始化 Plotting

Plotly provides a web-service for hosting graphs! Create a free account to get started. Graphs are saved inside your online Plotly account and you control the privacy. Public hosting is free, for private hosting, check out our paid plans.

Plotly 提供了一个在线托管图表的 web 服务！创建免费账户开始使用。图表保存在你的在线 Plotly 账户，你来控制权限。公共方式托管是免费的，也提供私有方式托管，查看我们的收费计划。

After installing the Plotly package, you're ready to fire up python:

安装好 Plotly 软件包后，你就可以启动 python 了：

```
$ python
```

and set your credentials:

并设置你的凭据：

In [1]:

```
import plotly
plotly.tools.set_credentials_file(username='DemoAccount',
api_key='lr1c37zw81')
```

You'll need to replace 'DemoAccount' and 'lr1c37zw81' with your Plotly username and API key.

你需要将'DemoAccount' 和 'lr1c37zw81'替换为你的 Plotly 用户名和 API key。Find your API key here.

在这里可以找到你的 API key。

The initialization step places a special .plotly/.credentials file in your home directory. Your ~/.plotly/.credentials file should look something like this:

初始化步骤会在你的家目录中存放一个特殊文件.plotly/.credentials，这个文件看起来像这样：

```
{
  "username": "DemoAccount",
```

```
    "stream_ids": ["ylosqsyet5", "h2ct8btk1s", "oxz4fm883b"],
    "api_key": "lr1c37zw81"
}
```

在线画图隐私

Online Plot Privacy

在线画图隐私

Plot can be set to three different type of privacies: public, private or secret.

有三种不同的隐私设置类型：公共，私有，秘密。

- **public**:
- **公共：**

Anyone can view this graph. It will appear in your profile and can appear in search engines. You do not need to be logged in to Plotly to view this chart.

任何人都可以查看此图表，它将显示在你的个人资料中，并可以显示在搜索引擎中。

你不需要登录 Plotly 就可以查看这些图表。

- **private**:
- **私有**

Only you can view this plot. It will not appear in the Plotly feed, your profile, or search engines. You must be logged in to Plotly to view this graph. You can privately share this graph with other Plotly users in your online Plotly account and they will need to be logged in to view this plot.

只有你自己可以查看此图表，

它不会显示在 Plotly 的 feed，你的个人资料，和搜索引擎中。

你需要登录才能查看这些图表。

你可以将图表在线私下分享给 Plotly 的其他账户，他们需要登录后才能查看这些图表。

- **secret**:
- **私密**

Anyone with this secret link can view this chart. It will not appear in the Plotly feed, your profile, or search engines. If it is embedded inside a webpage or an IPython notebook, anybody who is viewing that page will be able to view the graph. You do not need to be logged in to view this plot.

任何拥有此秘密链接的人都可以查看此图表。

它不会显示在 Plotly 的 feed，你的个人资料，和搜索引擎中。

如果它嵌入在 WEB 或 IPython notebook 中，任何在查看该网页的人能够看到该图表。

你不需要登录就可以查看这些图表。

By default all plots are set to **public**. Users with free account have the permission to keep one private plot. If you have additional private storage need, please visit Plotly products page.

默认情况下，所有图表都设置为公共 public 模式。免费账户可以有一个私有图表。如果你有额外的私有存储需求，请访问 Plotly 的产品页面。

If you're a PRO USER and would like the default setting for your plots to be private, you can edit your Plotly configuration:

如果你是一个专业用户，并且希望你的默认设置为私有模式，你可以编辑你的 Plotly 配置：

In [12]:

```
import plotly
plotly.tools.set_config_file(world_readable=False,
                             sharing='private')
```

For more examples on privacy settings please visit Python privacy documentation

有关隐私设置的更多示例，请访问此处隐私权文档

内部部署用户的特殊说明

Special Instructions for Plotly On-Premise Users

内部部署用户的特殊说明

Your API key for account on the public cloud will be different than the API key in Plotly On-Premise. Visit https://plotly.your-company.com/settings/api/ to find your Plotly On-Premise API key. Remember to replace "your-company.com" with the URL of your Plotly On-Premise server. If your company has a Plotly On-Premise server, change the Python API endpoint so that it points to your company's Plotly server instead of Plotly's cloud.

你在公共云上账户的 API key 将不同于内部部署的 API key。访问 https://plotly.your-company.com/settings/api/来找到你 Plotly 的内部部署的 API key，注意将“your-company”替换为内部部署的服务器。

如果你的公司有一台 Plotly 内部服务器，请修改 Python API 的端点参数，使其指向你公司的 Plotly 服务器，而不是 Plotly 云的。

In python, enter:

在 python 下，键入：

In [3]:

```
import plotly
plotly.tools.set_config_file(plotly_domain='https://plotly.your-company.com',
                             plotly_streaming_domain='stream-plotly.your-company.com',
                             plotly_api_domain='https://api-plotly.your-company.com')
```

Make sure to replace "your-company.com" with the URL of your Plotly On-Premise server.

确保将 **"your-company.com"**替换成了你的内部 Plotly 服务器。

Additionally, you can set your configuration so that you generate private plots by default. For more information on privacy settings see: https://plot.ly/python/privacy/

同时，你可以设置配置来缺省生成私有图，更多隐私设置请看 https://plot.ly/python/privacy/

In python, enter:

在 python 下，键入：

In [6]:

```
import plotly
plotly.tools.set_config_file(plotly_domain='https://plotly.your-company.com',
                             plotly_streaming_domain='stream-plotly.your-company.com',
                             plotly_api_domain='https://api-plotly.your-company.com',
                             world_readable=False,
                             sharing='private')
```

开始在线绘图

Start Plotting Online

开始在线绘图

When plotting online, the plot and data will be saved to your cloud account. There are two methods for plotting online: py.plot() and py.iplot(). Both options create a unique url for the plot and save it in your Plotly account.

在线绘图时，绘图和数据都保存在你的云账户。有两个方法进行在线绘图：py.plot() 和 py.iplot()。这两个操作都将创建唯一的网址，将其存放在你的 Plotly 账户中。

- Use py.plot() to return the unique url and optionally open the url.
- Use py.iplot() when working in a Jupyter Notebook to display the plot in the notebook.
- 使用 py.plot()返回一个唯一网址，并可选择打开网址。
- 在 Jupyter Notebook 中工作时，使用 py.iplot()进行绘图

Copy and paste one of the following examples to create your first hosted Plotly graph using the Plotly Python library:

复制粘贴以下示例之一，使用 Plotly 的 Python 库创建你的第一个托管图：

In [1]:

```
import plotly.plotly as py
from plotly.graph_objs import *

trace0 = Scatter(
   x=[1, 2, 3, 4],
   y=[10, 15, 13, 17]
)
trace1 = Scatter(
   x=[1, 2, 3, 4],
   y=[16, 5, 11, 9]
)
data = Data([trace0, trace1])

py.plot(data, filename = 'basic-line')
```

Out[1]:

```
u'https://plot.ly/~chelsea_lyn/8758'
```

Checkout the docstrings for more information:

查看 docstrings 可获得更多信息：

In [2]:

```
import plotly.plotly as py
help(py.plot)
Help on function plot in module plotly.plotly.plotly:
```

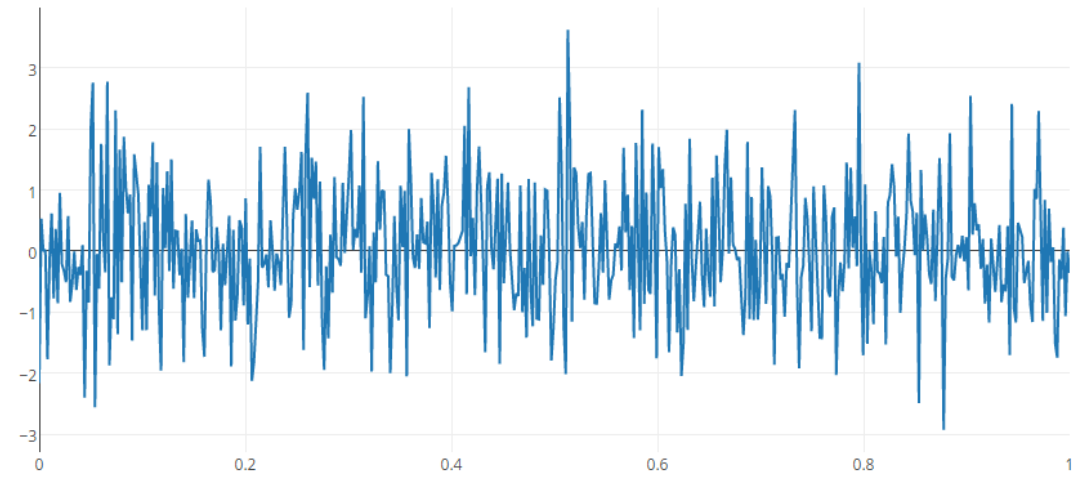
```
plot(figure_or_data, validate=True, **plot_options)
    Create a unique url for this plot in Plotly and optionally open url.

    plot_options keyword agruments:
    filename (string) -- the name that will be associated with this figure
    fileopt ('new' | 'overwrite' | 'extend' | 'append') -- 'new' creates a
        'new': create a new, unique url for this plot
        'overwrite': overwrite the file associated with `filename` with this
        'extend': add additional numbers (data) to existing traces
        'append': add additional traces to existing data lists
    auto_open (default=True) -- Toggle browser options
        True: open this plot in a new browser tab
        False: do not open plot in the browser, but do return the unique url
    sharing ('public' | 'private' | 'secret') -- Toggle who can view this
                                                 graph
        - 'public': Anyone can view this graph. It will appear in your profile
                    and can appear in search engines. You do not need to be
                    logged in to Plotly to view this chart.
        - 'private': Only you can view this plot. It will not appear in the
                     Plotly feed, your profile, or search engines. You must be
                     logged in to Plotly to view this graph. You can privately
                     share this graph with other Plotly users in your online
                     Plotly account and they will need to be logged in to
                     view this plot.
        - 'secret': Anyone with this secret link can view this chart. It will
                    not appear in the Plotly feed, your profile, or search
                    engines. If it is embedded inside a webpage or an IPython
                    notebook, anybody who is viewing that page will be able to
                    view the graph. You do not need to be logged in to view
                    this plot.
    world_readable (default=True) -- Deprecated: use "sharing".
                                     Make this figure private/public
```

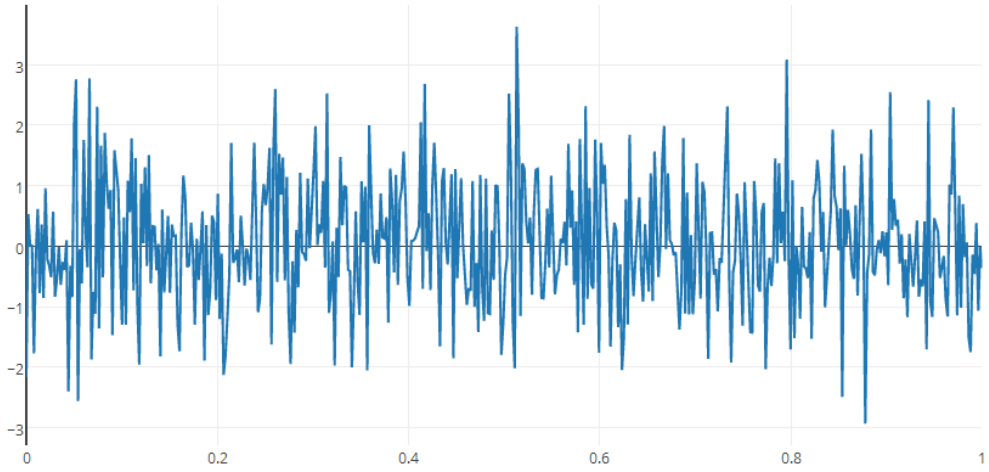
In [3]:

```
import plotly.plotly as py
```

```
from plotly.graph_objs import *

trace0 = Scatter(
    x=[1, 2, 3, 4],
    y=[10, 15, 13, 17]
)
trace1 = Scatter(
    x=[1, 2, 3, 4],
    y=[16, 5, 11, 9]
)
data = Data([trace0, trace1])

py.iplot(data, filename = 'basic-line')
```

Out[3]:





See more examples in our IPython notebook documentation or check out the py.iplot() docstring for more information.

获取更多示例请参考 IPython 笔记文档或使用 py.iplot()以获取更多信息。

In [4]:

```
import plotly.plotly as py
help(py.iplot)
Copy to clipboard!
Help on function iplot in module plotly.plotly.plotly:

iplot(figure_or_data, **plot_options)
    Create a unique url for this plot in Plotly and open in IPython.
    # 为 Plotly 中的图像创建一个独一无二的链接并在 IPython 中打开。

    plot_options keyword agruments:
    filename (string) -- the name that will be associated with this figure
    fileopt ('new' | 'overwrite' | 'extend' | 'append')
        - 'new': create a new, unique url for this plot # 创建一个新的独一无二的链接
        - 'overwrite': overwrite the file associated with `filename` with this # 重写文件
        - 'extend': add additional numbers (data) to existing traces # 添加额外的数据到现有路径中
        - 'append': add additional traces to existing data lists # 添加额外的路径到数据列表中

    sharing ('public' | 'private' | 'secret') -- Toggle who can view this graph
        - 'public': Anyone can view this graph. It will appear in your profile
            and can appear in search engines. You do not need to be
                logged in to Plotly to view this chart.
    # 任何人都可以访问这个图像，并且这一图像会出现在你的配置文件和搜索引擎中，你
    # 不需要去登入 Plotly 去访问。

        - 'private': Only you can view this plot. It will not appear in the
                Plotly feed, your profile, or search engines. You must be
                logged in to Plotly to view this graph. You can privately
                share this graph with other Plotly users in your online
                Plotly account and they will need to be logged in to
                view this plot.
    # 只有你可以访问这个图像，这一图像不会出现在你的配置文件和搜索引擎中，你需
    # 你需要去登入 Plotly 去访问。当然，你可以把图标分享给其它用户，但是他们必须
    # 登入 Plotly 才可以访问这一图像。
```

```
        - 'secret': Anyone with this secret link can view this chart. It will
                    not appear in the Plotly feed, your profile, or search
                    engines. If it is embedded inside a webpage or an IPython
                    notebook, anybody who is viewing that page will be able to
                    view the graph. You do not need to be logged in to view
                    this plot.
    # 任何有这个隐蔽链接的人都可以访问这个图像，但是图像不会出现在你的 Plotly feed
    # 配置文件或是搜索引擎中。如果图像被放到了网页或是 IPython 笔记中，那么任何人
不
    # 需要登入就可以看到这一图像。

    world_readable (default=True) -- Deprecated: use "sharing".
                                     Make this figure private/public
```

You can also create plotly graphs with matplotlib syntax. Learn more in our matplotlib documentation.

您可以使用 *matplotlib* 库绘图，更多信息请见 matplotlib documentation

离线绘图初始化

Initialization for Offline Plotting

初始化 Plotly 单机绘图模块

Plotly Offline allows you to create graphs offline and save them locally. There are also two methods for plotting offline: plotly.offline.plot () and plotly.offline.iplot ().

- Use plotly.offline.plot () to create and standalone HTML that is saved locally and opened inside your web browser.

- Use plotly.offline.iplot () when working offline in a Jupyter Notebook to display the plot in the notebook.

Check your Plotly version, version 1.9.4+ is needed for offline plotting:

单机版 Plotly 允许你在没有网络的情况下画图，并把图像保存到本地。有两种方法可以实现上述功能：plotly.offline.plot () 和 plotly.offline.iplot ()

- 使用 plotly.offline.plot () 去创建一个独立的 HTML 接口并保存到本地，这一 HTML 可以在你的浏览器中打开。
- 使用 plotly.offline.iplot () 当你单机使用 Jupyter Notebook 时，可以把图像显示到 Jupyter 中。

1.94 以上的 Plotly 版本支持上述功能，使用以下代码查看版本信息。

In [5]:

```
import plotly
plotly.__version__
```

Out[5]:

'1.12.9'

Copy and paste one of the following examples to create your first offline Plotly graph using the Plotly Python library:

拷贝粘贴下列的任意一个例子，使用 Plotly 去创建你的第一个 Offline Plotly 图像。

In [6]:

```
import plotly
from plotly.graph_objs import Scatter, Layout

plotly.offline.plot({
    "data": [Scatter(x=[1, 2, 3, 4], y=[4, 3, 2, 1])],
    "layout": Layout(title="hello world")
})
```

Out[6]:

'file:///Users/Chelsea/Repos/documentation/_posts/python/getting-started/temp-plot.html'

Learn more by calling help ():

更多信息请见帮助文件

In [7]:

```
import plotly
help(plotly.offline.plot)

Help on function plot in module plotly.offline.offline:

plot(figure_or_data, show_link=True, link_text='Export to plot.ly', validate=True, output_type='file', include_plotlyjs=True, filename='temp-plot.html', auto_open=True, image=None, image_filename='plot_image', image_width=800, image_height=600)
```

```
        Create a plotly graph locally as an HTML document or string.

        Example:
        ```
        from plotly.offline import plot
        import plotly.graph_objs as go

        plot([go.Scatter(x=[1, 2, 3], y=[3, 2, 6])], filename='my-graph.html')
        # We can also download an image of the plot by setting the image parameter
        # to the image format we want
        # 我们也可以通过把参数 filename 设置为我们想要的图片格式('jpg')，直接下载图
像
        plot([go.Scatter(x=[1, 2, 3], y=[3, 2, 6])], filename='my-graph.html'
            image='jpeg')
        ```
        More examples below

        figure_or_data -- a plotly.graph_objs.Figure or plotly.graph_objs.Data or
                          dict or list that describes a Plotly graph.
                          See https://plot.ly/python/ for examples of
                          graph descriptions.
        # plotly.graph_objs.Figure 或是 plotly.graph_objs.Data 或是 dict 或是 list 可以描述
一个
        # Plotly 图像

        Keyword arguments:
        show_link (default=True) -- display a link in the bottom-right corner of
            of the chart that will export the chart to Plotly Cloud or
            Plotly Enterprise
        # 把链接放置到图像的右下角，可以把图像导出到 Plotly Cloud 或是 Plotly Enterprise
中

        link_text (default='Export to plot.ly') -- the text of export link
        # 链接的文本文档

        validate (default=True) -- validate that all of the keys in the figure
            are valid? omit if your version of plotly.js has become outdated
            with your version of graph_reference.json or if you need to include
            extra, unnecessary keys in your figure.
```

```
        # 验证图像中所有的关键字是否有效，删除关键字，如果你的 plotly.js 版本太旧或是你
的
        # 图像中包含了无关紧要的关键字

        output_type ('file' | 'div' - default 'file') -- if 'file', then
            the graph is saved as a standalone HTML file and `plot`
            returns None.
            If 'div', then `plot` returns a string that just contains the
            HTML <div> that contains the graph and the script to generate the
            graph.
            Use 'file' if you want to save and view a single graph at a time
            in a standalone HTML file.
            Use 'div' if you are embedding these graphs in an HTML file with
            other graphs or HTML markup, like a HTML report or an website.
    # 如果是’file’，那么图像保存为单独的网页，plot 的返回值为 None
    # 如果是’div’，那么 plot 的返回一个字符串，包含了一个包括图像和脚本的 HTML
    # 使用’file’，如果你想要一个单独的图像，并且想要能够在任何时刻以网页形式打开
    # 使用’div’，如果你想把这些图像和其它图像与网页标记嵌入一个 HTML 文件中
    # 类似于 HTML 报告或是一个网站。

        include_plotlyjs (default=True) -- If True, include the plotly.js
            source code in the output file or string.
            Set as False if your HTML file already contains a copy of the plotly.js
            library.

    # 如果参数为’True’，那么在输出的文件或字符串中会包含 plotly.js 的源程序。
    # 设置参数为’False’，如果你的 HTML 文件中已经包含了 plotly.js 库的信息。

        filename (default='temp-plot.html') -- The local filename to save the
            outputted chart to. If the filename already exists, it will be
            overwritten. This argument only applies if `output_type` is 'file'.
    # 保存文件到本地，如果文件已经存在，那么就把新内容更新到这个文件。
    # 本语句只适用于输出类型为’file’的情况。

        auto_open (default=True) -- If True, open the saved file in a
            web browser after saving.
            This argument only applies if `output_type` is 'file'.
    #  ‘True’：在文件保存之后在浏览器中打开
    # 本语句只适用于输出类型为’file’的情况。
```

```
        image (default=None |'png' |'jpeg' |'svg' |'webp') -- This parameter sets
            the format of the image to be downloaded, if we choose to download an
            image. This parameter has a default value of None indicating that no
            image should be downloaded.
    # 默认参数是 None，表示不会去下载图像。当参数是’ png’,’ jpeg’,’ svg’,’
webp’ 时，下载相应
    # 格式的图像。

        image_filename (default='plot_image') -- Sets the name of the file your image
            will be saved to. The extension should not be included.
    # 设置你要保存的图片名称，请不要加上后缀名

        image_height (default=600) -- Specifies the height of the image in `px`.
    # 指定图像的高，默认值为 600 像素

        image_width (default=800) -- Specifies the width of the image in `px`.
    # 指定图像的宽，默认值为 800 像素
```

Out[7]:

When using plotly.offline.iplot to plot offline in Jupyter Notebooks, there is an additional initialization step of running: plotly.offline.init_notebook_mode () at the start of each notebook session.

当使用 plotly.offline.iplot 在 Jupyter 中画图时，需要又一个附加的初始化步骤：在开始使用前输入 plotly.offline.init_notebook_mode ()

See the example below:

下面是一个例子：

In [9]:

```
import plotly
from plotly.graph_objs import Scatter, Layout

plotly.offline.init_notebook_mode()

plotly.offline.iplot({
    "data": [Scatter(x=[1, 2, 3, 4], y=[4, 3, 2, 1])],
    "layout": Layout(title="hello world")
})
```

Out[9]:

In [10]:

```
import plotly
help(plotly.offline.iplot)

Help on function iplot in module plotly.offline.offline:

iplot(figure_or_data, show_link=True, link_text='Export to plot.ly', validate=True,
image=None, filename='plot_image', image_width=800, image_height=600)

    Draw plotly graphs inside an IPython notebook without
    connecting to an external server.
# 在 IPython notebook 中使用 Plotly 画图不连接外部服务器

    To save the chart to Plotly Cloud or Plotly Enterprise, use
    `plotly.plotly.iplot`.
# 使用`plotly.plotly.iplot`保存图表到 Plotly Cloud 或 Plotly Enterprise 中

    To embed an image of the chart, use `plotly.image.ishow`.
 # 使用`plotly.image.ishow`把图像嵌入图表中

    figure_or_data -- a plotly.graph_objs.Figure or plotly.graph_objs.Data or
                      dict or list that describes a Plotly graph.
                      See https://plot.ly/python/ for examples of
                      graph descriptions.

    Keyword arguments:
    show_link (default=True) -- display a link in the bottom-right corner of
                                of the chart that will export the chart to
                                Plotly Cloud or Plotly Enterprise
    # 把链接放置到图像的右下角，可以把图像导出到 Plotly Cloud 或是 Plotly Enterprise
中

    link_text (default='Export to plot.ly') -- the text of export link
    # 链接的文本文档

    validate (default=True) -- validate that all of the keys in the figure
                               are valid? omit if your version of plotly.js
                               has become outdated with your version of
                               graph_reference.json or if you need to include
```

```
                    extra, unnecessary keys in your figure.
        # 验证图像中所有的关键字是否有效，删除关键字，如果你的 plotly.js 版本太旧或是你
的
        # 图像中包含了无关紧要的关键字

        image (default=None |'png' |'jpeg' |'svg' |'webp') -- This parameter sets
            the format of the image to be downloaded, if we choose to download an
            image. This parameter has a default value of None indicating that no
            image should be downloaded.
    # 默认参数是 None，表示不会去下载图像。当参数是’ png’,’ jpeg’,’ svg’,’
webp’ 时，下载相应
    # 格式的图像。

        filename (default='plot') -- Sets the name of the file your image
            will be saved to. The extension should not be included.
    # 设置你要保存的图片名称，请不要加上后缀名

        image_height (default=600) -- Specifies the height of the image in `px`.
    # 指定图像的高，默认值为 600 像素

        image_width (default=800) -- Specifies the width of the image in `px`.
    # 指定图像的宽，默认值为 800 像素

        Example:
        ```
        from plotly.offline import init_notebook_mode, iplot
        init_notebook_mode()
        iplot([{'x': [1, 2, 3], 'y': [5, 2, 7]}])
        # We can also download an image of the plot by setting the image to the
        format you want. e.g. `image='png'`
        iplot([{'x': [1, 2, 3], 'y': [5, 2, 7]}], image='png')
```

**Out[10]:**

For more examples on plotting offline with Plotly in python please visit our offline documentation.

想要获取更多的 Plotly 离线画图的示例，请访问 offline documentation 获取信息。
```

更多示例

MORE EXAMPLES
更多示例
Check out more examples and tutorials for using Plotly in python here!

点击这里获取 Plotly 更多的示例与教程。

基础图形

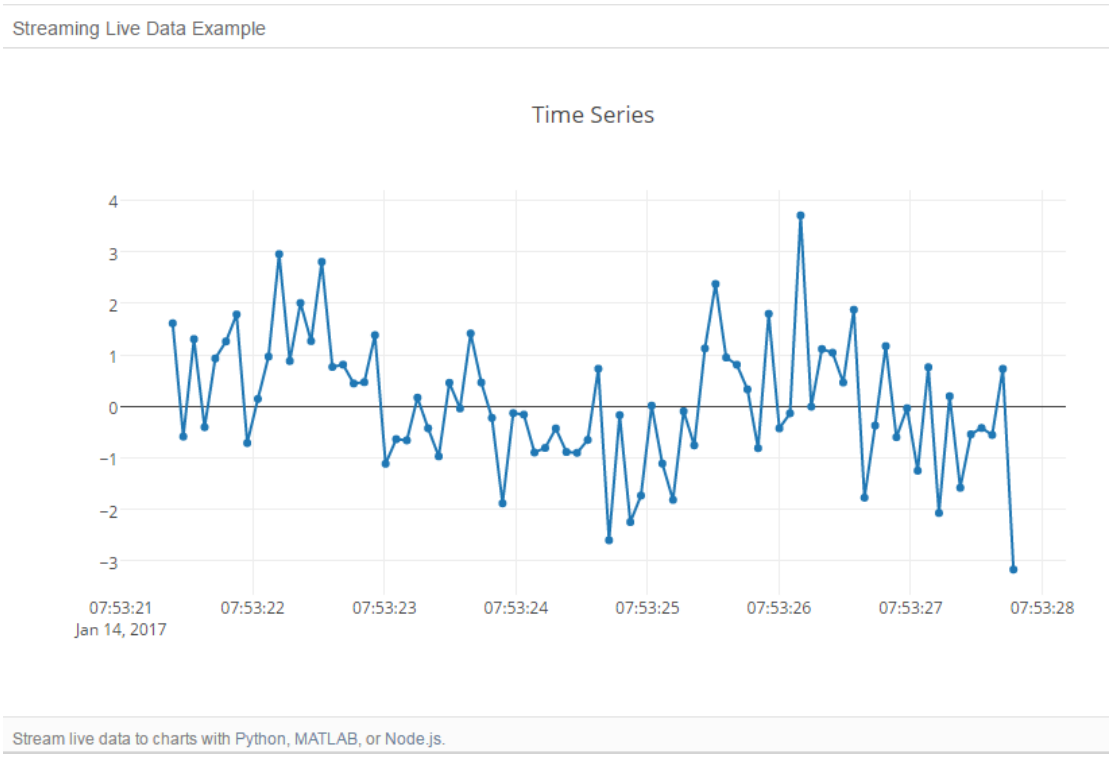
Basic Charts

导航预览

Dashboard
导航预览

动态数据示例

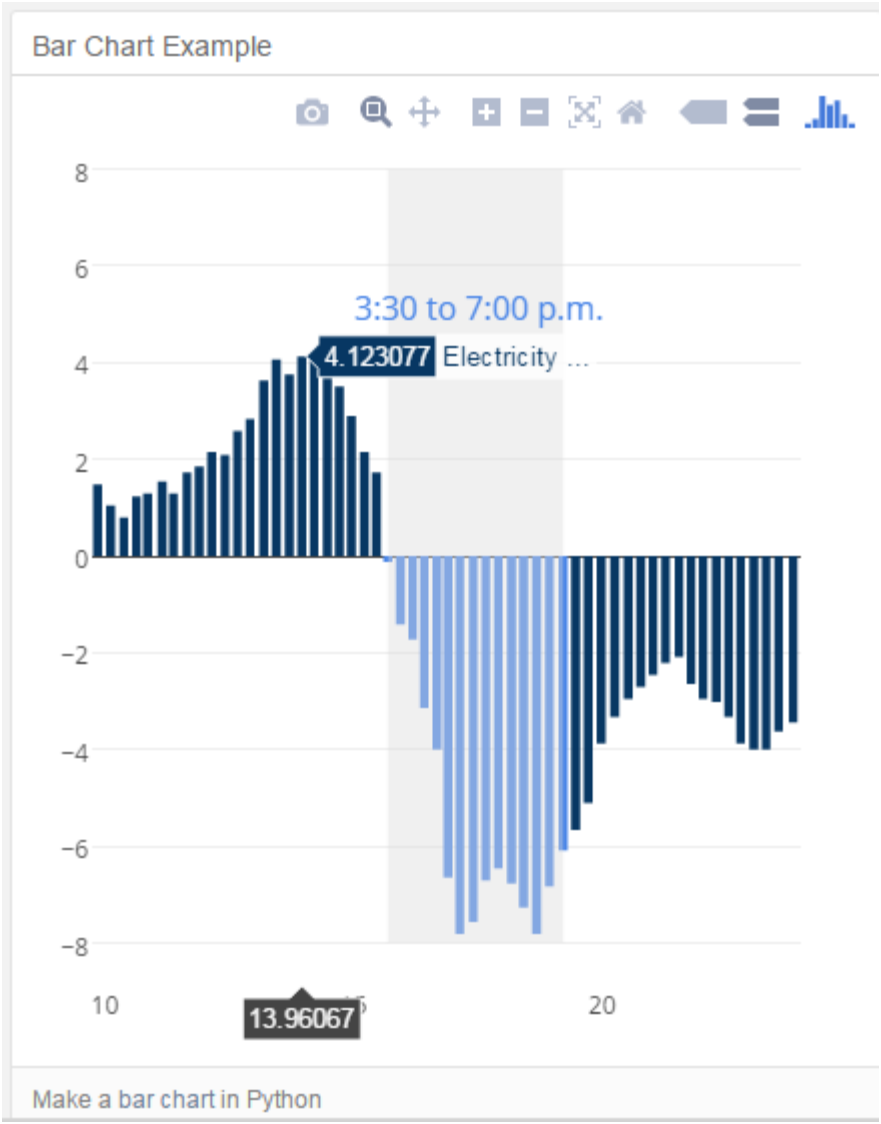
Streaming Live Data Example
动态数据示例



柱状图示例

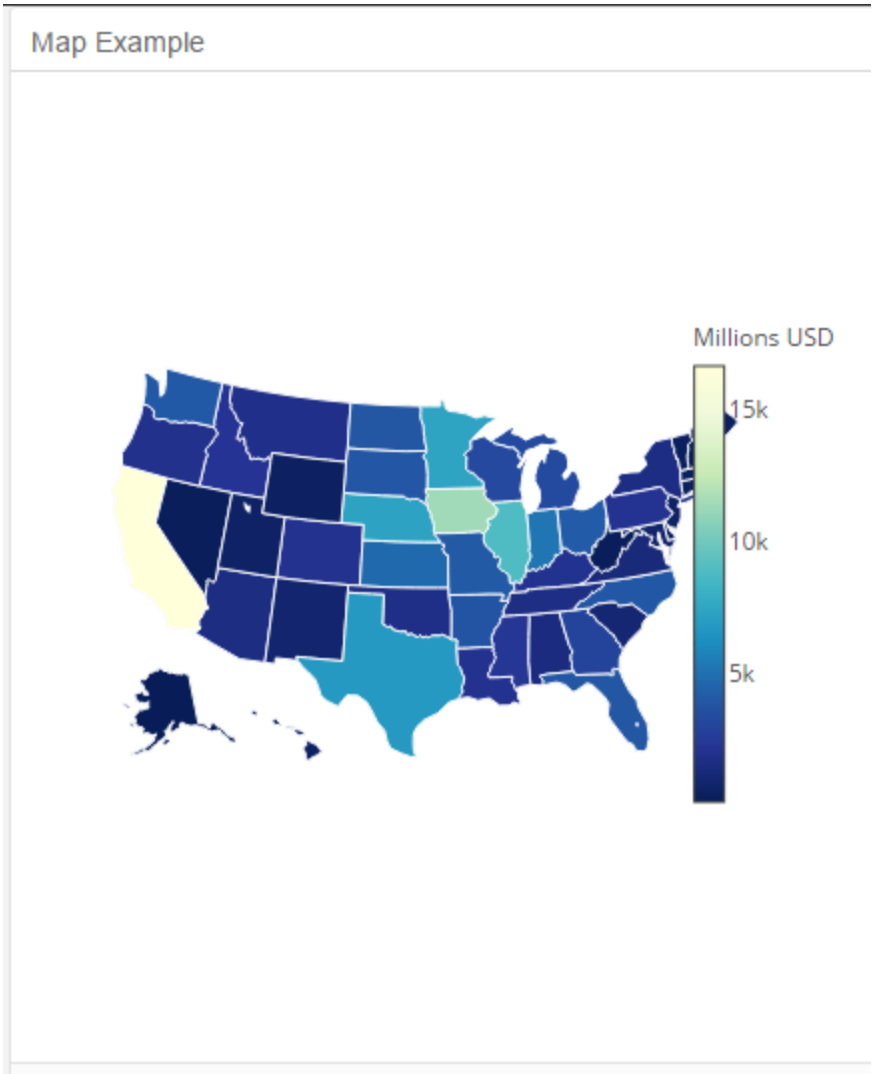
Bar Chart Example

柱状图示例



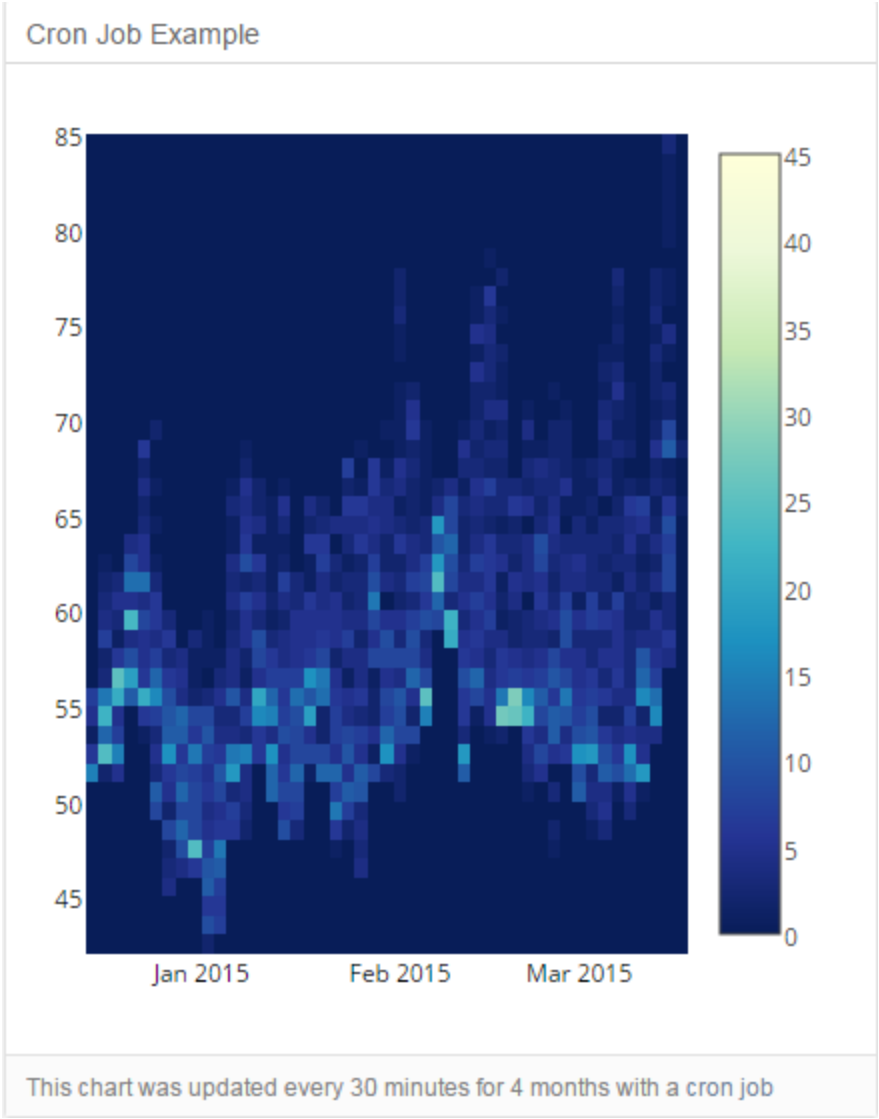
地图示例

Map Example



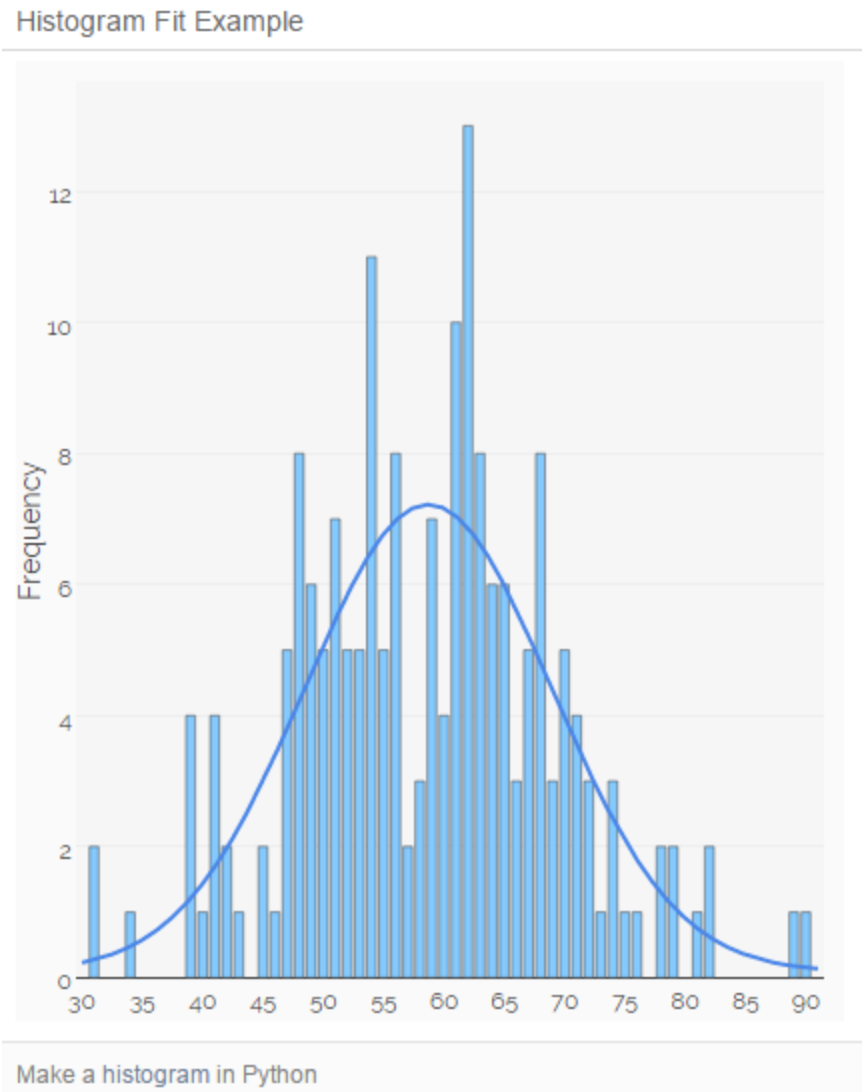
定时计划任务案例

Cron Job Example



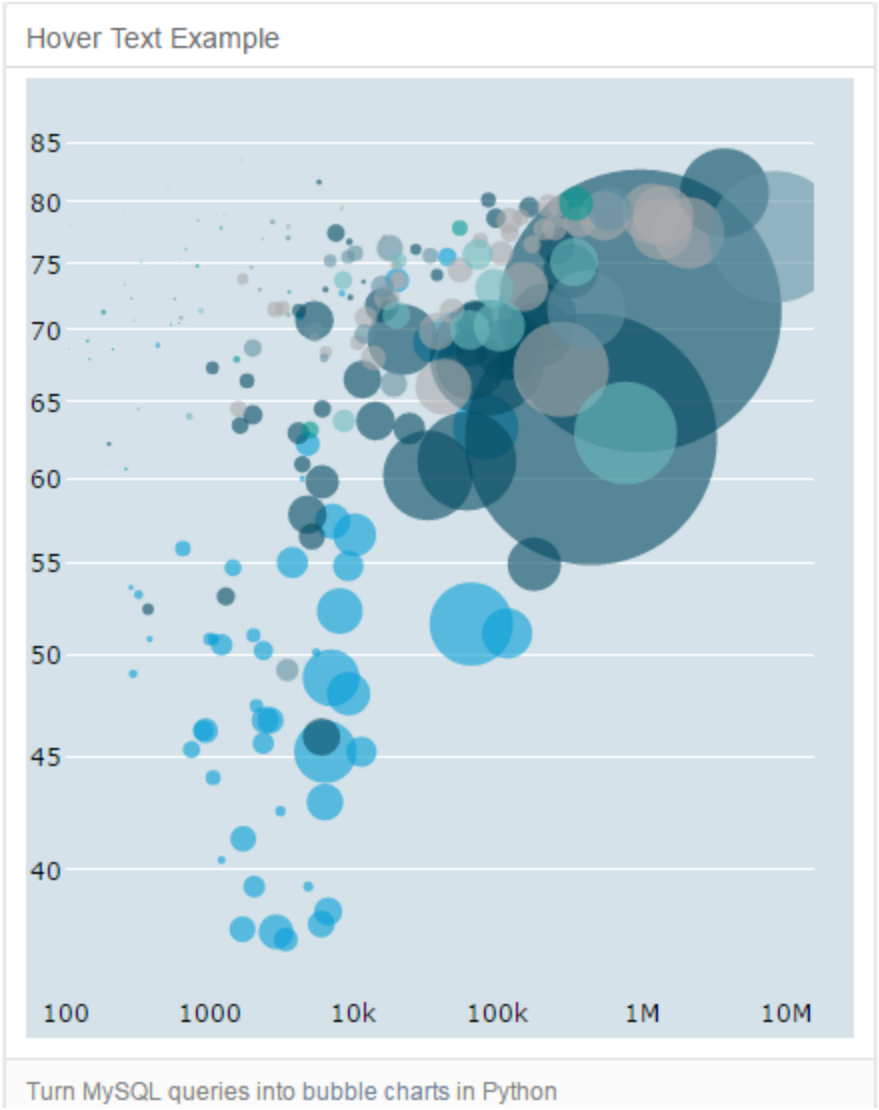
直方图

Histogram Fit Example



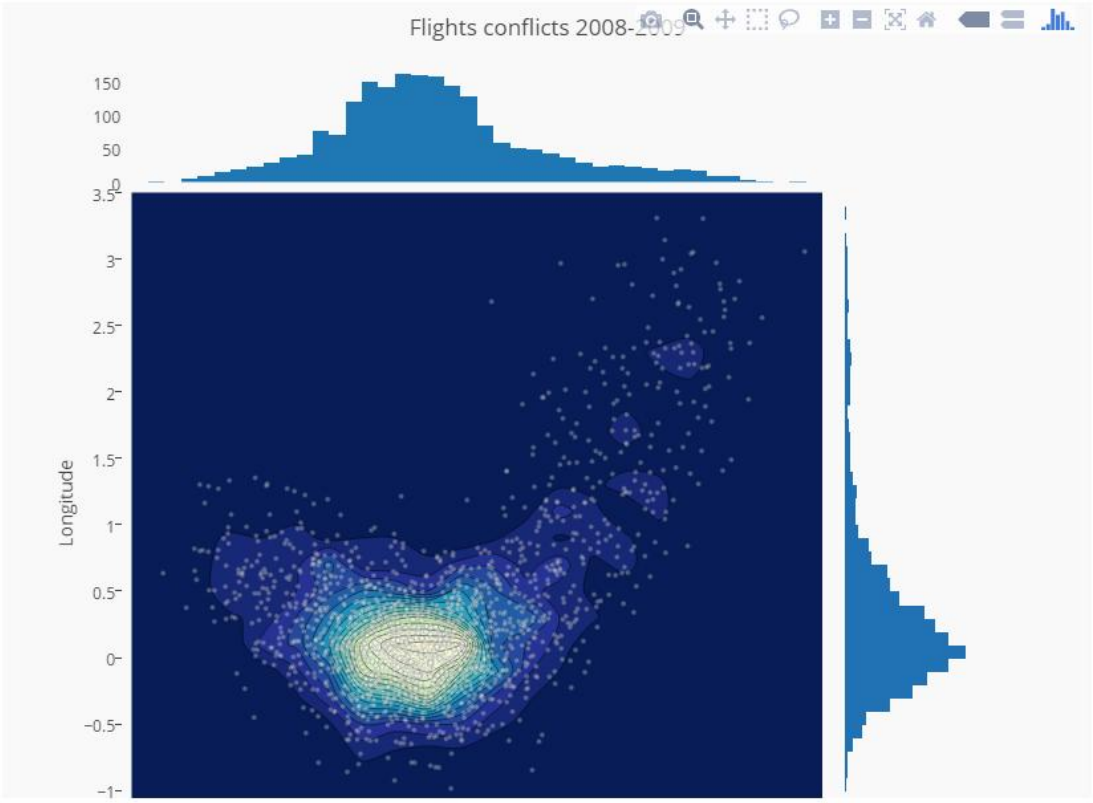
悬浮文字说明

Hover Text Example



自定义 js 控制案例

Custom JavaScript Controls Example



WebGL 与 SVG 绘图绘图

WebGL vs SVG in Python

Implement WebGL for increased speed, improved interactivity, and the ability to plot even more data!

实现 WebGL 加速,提高交互性,并且处理更多数据的能力!

刚开始接触 Plotly ?

New to Plotly?

Plotly's Python library is free and open source! Get started by downloading the client and reading the primer.

You can set up Plotly to work in online or offline mode, or in jupyter notebooks.

We also have a quick-reference cheatsheet (new!) to help you get started!!

Plotly 的 Python 库是免费并且开源的。Get started by downloading the client and reading the primer。

你可以设置 Plotly 以 online 或者 offline 的方式运行，也可以在 jupyter notebooks 中运行。

我们同样也有一个快速参考的 cheatsheet (new!)来帮助新手学习。

对比 WebGL 与 SVG

Compare WebGL and SVG

Checkout this notebook to compare WebGL and SVG scatter plots with 75,000 random data points

查看 this notebook，用 75,000 个随机数据点来对比 WebGL 和 SVG 散点图。

100,000 个点的 WebGL

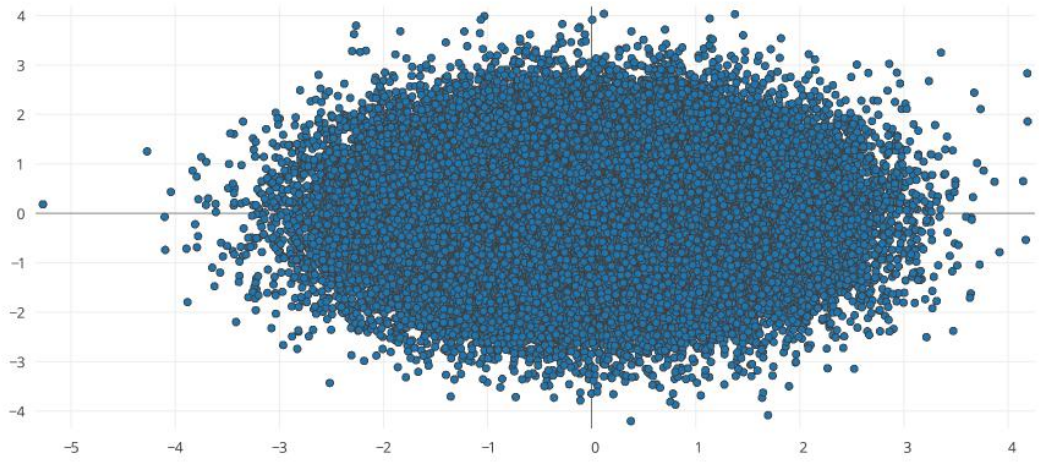
WebGL with 100,000 points

In [2]:

```
import plotly.plotly as py
import plotly.graph_objs as go

import numpy as np

N = 100000
trace = go.Scattergl(
    x = np.random.randn(N),
    y = np.random.randn(N),
    mode = 'markers',
    marker = dict(
        line = dict(
            width = 1,
            color = '#404040')
    )
)
```

```
data = [trace]
py.iplot(data, filename='WebGL100000')
```

Out[2]:



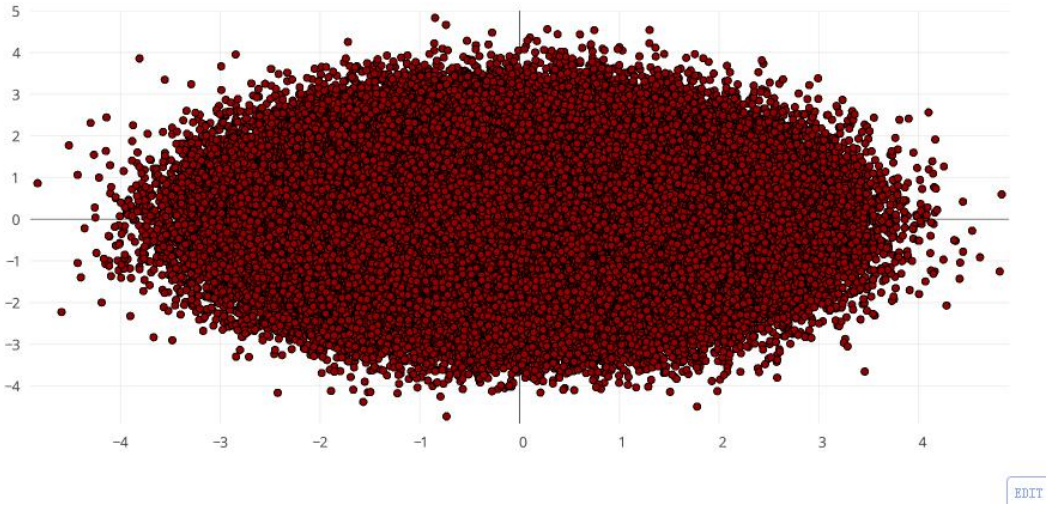
1 百万个点的 WebGL

WebGL with 1 Million Points

In [3]:

```
import plotly.plotly as py
import plotly.graph_objs as go

import numpy as np

N = 1000000
trace = go.Scattergl(
    x = np.random.randn(N),
    y = np.random.randn(N),
    mode = 'markers',
    marker = dict(
        color = 'rgb(152, 0, 0)',
        line = dict(
            width = 1,
            color = 'rgb(0,0,0)')
    )
)
data = [trace]
py.iplot(data, filename='WebGLmillion')
```

Out[3]:



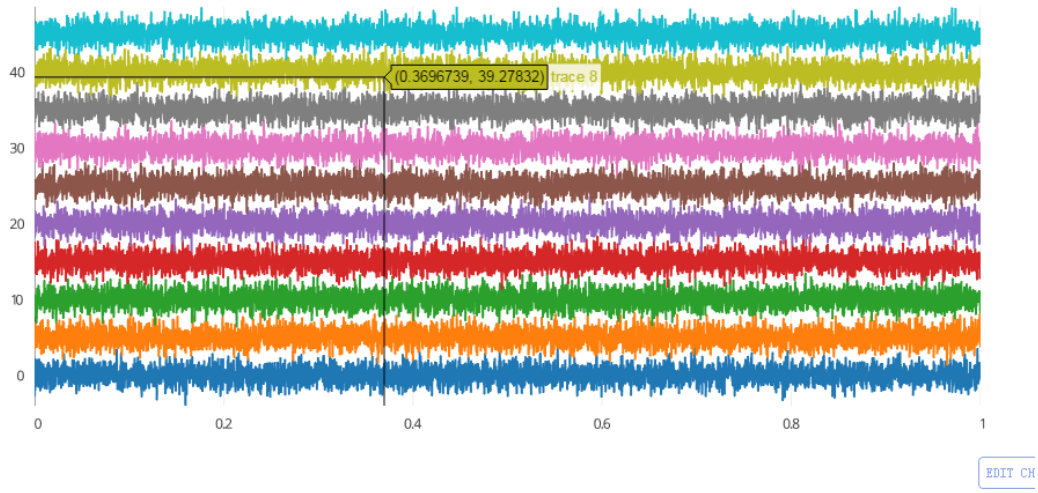
多线条的 WebGL

WebGL with many traces

In [4]:

```
import plotly.plotly as py
import plotly.graph_objs as go

import numpy as np

data = []
trace_num = 10
point_num = 5000
for i in range(trace_num):
    data.append(go.Scattergl(
        x = np.linspace(0, 1, point_num),
        y = np.random.randn(point_num)+(i*5)
    )
)
layout = dict(showlegend=False)
fig=dict(data=data, layout=layout)
py.iplot(fig, filename='WebGL_line')
```

Out[4]:



参考文档

Reference

See https://plot.ly/python/reference/#scattergl for more information and chart attribute options!

查看 https://plot.ly/python/reference/#scattergl，获取更多的信息与图表属性的选项！

散点图

Scatter Plots in Python

刚开始接触 Plotly？

New to Plotly?

Plotly's Python library is free and open source! Get started by downloading the client and reading the primer.
You can set up Plotly to work in online or offline mode, or in jupyter notebooks.
We also have a quick-reference cheatsheet (new!) to help you get started!!

Plotly 的 Python 库是免费并且开源的。Get started by downloading the client and reading the primer。

你可以设置 Plotly 以 online 或者 offline 的方式运行，也可以在 jupyter notebooks 中运行。

我们同样也有一个快速参考的 cheatsheet (new!)来帮助新手学习。

在线输出静态图形

Export Static Images Online

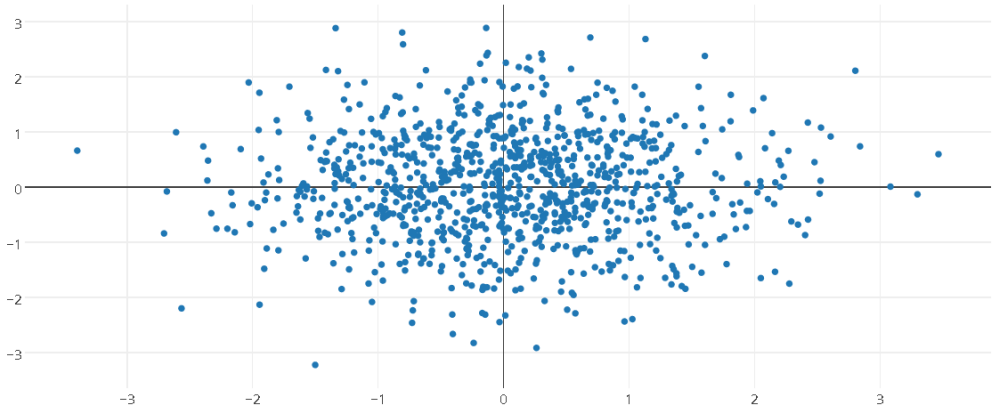
To save the image, you need to login to plotly using your credentials (username and API Key).

你需要在 plotly 上注册你的凭证（用户名和 API 码）才可以保存静态图形

In [1]:

```
import plotly.plotly as py
import plotly.graph_objs as go

# Create random data with numpy
import numpy as np

N = 1000
random_x = np.random.randn(N)
random_y = np.random.randn(N)

# Create a trace
trace = go.Scatter(
    x = random_x,
    y = random_y,
    mode = 'markers'
)

data = [trace]

# Plot and embed in ipython notebook!
py.iplot(data, filename='basic-scatter')
```

Out[1]:



线和散点图

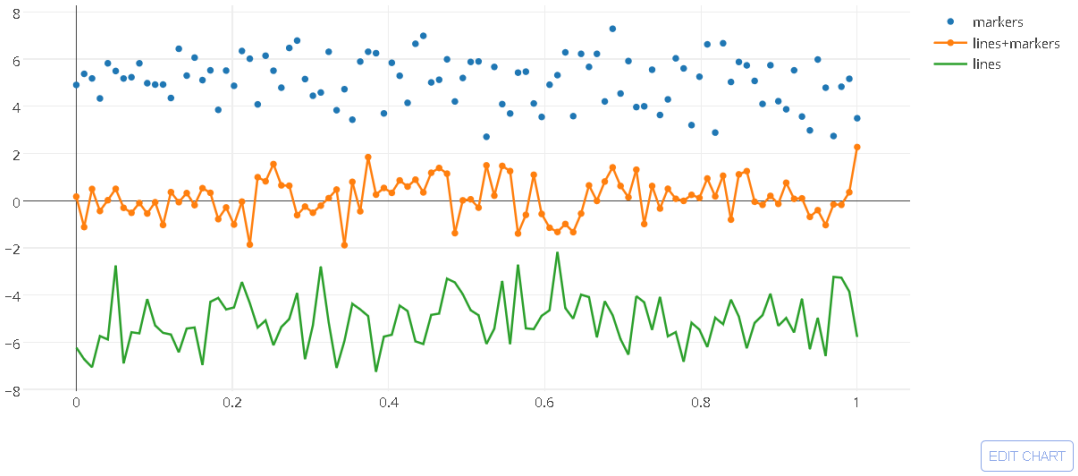
Line and Scatter Plot

In [2]:

```
import plotly.plotly as py
import plotly.graph_objs as go

# Create random data with numpy
import numpy as np

N = 100
random_x = np.linspace(0, 1, N)
random_y0 = np.random.randn(N)+5
random_y1 = np.random.randn(N)
random_y2 = np.random.randn(N)-5

# Create traces
trace0 = go.Scatter(
    x = random_x,
    y = random_y0,
    mode = 'markers',
    name = 'markers'
)
trace1 = go.Scatter(
    x = random_x,
    y = random_y1,
    mode = 'lines+markers',
    name = 'lines+markers'
)
trace2 = go.Scatter(
    x = random_x,
    y = random_y2,
    mode = 'lines',
    name = 'lines'
)
```

```
data = [trace0, trace1, trace2]
py.iplot(data, filename='scatter-mode')
```

Out[2]:



散点图的样式

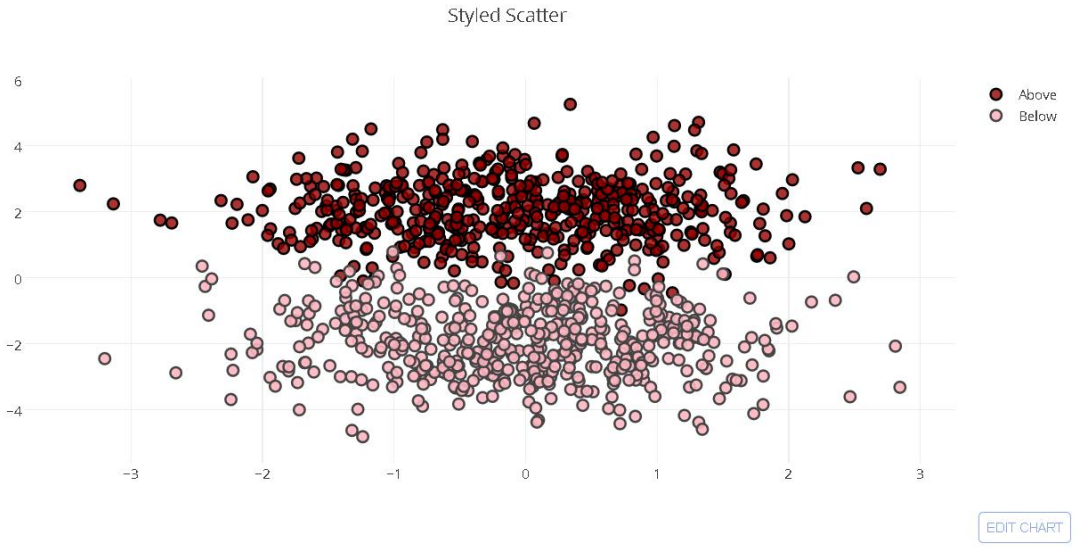
Style Scatter Plots

In [3]:

```
    import plotly.plotly as py
import plotly.graph_objs as go

import numpy as np

N = 500

trace0 = go.Scatter(
    x = np.random.randn(N),
    y = np.random.randn(N)+2,
    name = 'Above',
    mode = 'markers',
    marker = dict(
        size = 10,
        color = 'rgba(152, 0, 0, .8)',
        line = dict(
            width = 2,
            color = 'rgb(0, 0, 0)'
        )
    )
)

trace1 = go.Scatter(
    x = np.random.randn(N),
    y = np.random.randn(N)-2,
    name = 'Below',
    mode = 'markers',
    marker = dict(
```

```
            size = 10,
            color = 'rgba(255, 182, 193, .9)',
            line = dict(
                width = 2,
            )
        )
    )

    data = [trace0, trace1]

    layout = dict(title = 'Styled Scatter',
                  yaxis = dict(zeroline = False),
                  xaxis = dict(zeroline = False)
                 )

    fig = dict(data=data, layout=layout)
    py.iplot(fig, filename='styled-scatter')
```

Out[3]:



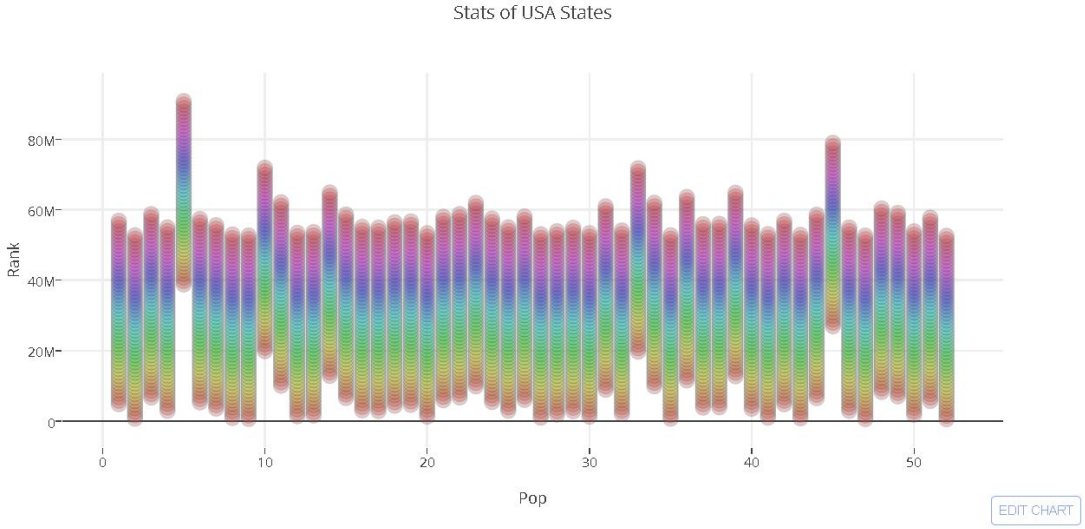
悬浮数据标签

Data Labels on Hover

In [4]:

```
    import plotly.plotly as py
    import plotly.graph_objs as go
    import random
    import numpy as np
    import pandas as pd

    l= []
    y= []
    data=
pd.read_csv("https://raw.githubusercontent.com/plotly/datasets/master/2014_usa_states.csv")
    # Setting colors for plot.
    N= 53
```

```
c= ['hsl('+str(h)+',50%'+',50%)' for h in np.linspace(0, 360, N)]

for i in range(int(N)):
    y.append((2000+i))
    trace0= go.Scatter(
        x= data['rank'],
        y= data['pop']+(i*1000000),
        mode= 'markers',
        marker= dict(size= 14,
                     line= dict(width=1),
                     color= c[i],
                     opacity= 0.3
                    ),name= y[i],
        text= data['state']) # The hover text goes here...
    l.append(trace0);

layout= go.Layout(
    title= 'Stats of USA States',
    hovermode= 'closest',
    xaxis= dict(
        title= 'Pop',
        ticklen= 5,
        zeroline= False,
        gridwidth= 2,
    ),
    yaxis=dict(
        title= 'Rank',
        ticklen= 5,
        gridwidth= 2,
    ),
    showlegend= False
)
fig= go.Figure(data=l, layout=layout)
py.iplot(fig)
```

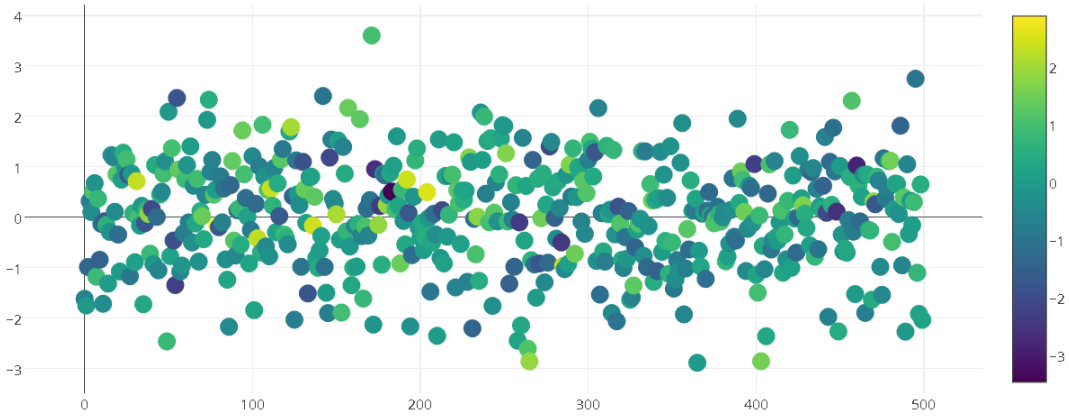


具有颜色尺寸的散点图

Scatter with a Color Dimension

In [5]:

```
import plotly.graph_objs as go
import plotly.plotly as py

import numpy as np

trace1 = go.Scatter(
    y = np.random.randn(500),
    mode='markers',
    marker=dict(
        size='16',
        color = np.random.randn(500), #set color equal to a variable
        colorscale='Viridis',
        showscale=True
    )
)
data = [trace1]

py.iplot(data, filename='scatter-plot-with-colorscale')
```



分类散点图

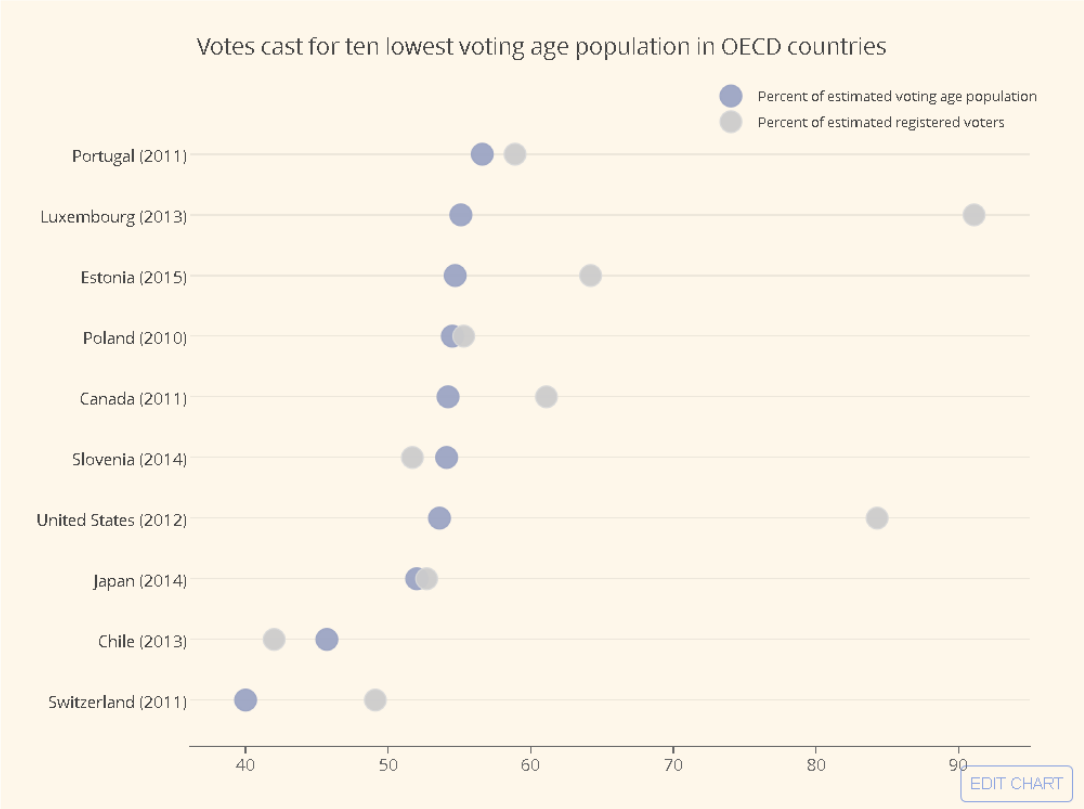
Categorical Dot Plot

In [6]:

```
import plotly.plotly as py
import plotly.graph_objs as go

country = ['Switzerland (2011)', 'Chile (2013)', 'Japan (2014)', 'United States (2012)', 'Slovenia
(2014)', 'Canada (2011)', 'Poland (2010)', 'Estonia (2015)', 'Luxembourg (2013)', 'Portugal (2011)']
voting_pop = [40, 45.7, 52, 53.6, 54.1, 54.2, 54.5, 54.7, 55.1, 56.6]
reg_voters = [49.1, 42, 52.7, 84.3, 51.7, 61.1, 55.3, 64.2, 91.1, 58.9]

trace0 = go.Scatter(
    x=voting_pop,
```

```
        y=country,
        mode='markers',
        name='Percent of estimated voting age population',
        marker=dict(
            color='rgba(156, 165, 196, 0.95)',
            line=dict(
                color='rgba(156, 165, 196, 1.0)',
                width=1,
            ),
            symbol='circle',
            size=16,
        )
    )
    trace1 = go.Scatter(
        x=reg_voters,
        y=country,
        mode='markers',
        name='Percent of estimated registered voters',
        marker=dict(
            color='rgba(204, 204, 204, 0.95)',
            line=dict(
                color='rgba(217, 217, 217, 1.0)',
                width=1,
            ),
            symbol='circle',
            size=16,
        )
    )
    data = [trace0, trace1]
    layout = go.Layout(
        title="Votes cast for ten lowest voting age population in OECD countries",
        xaxis=dict(
            showgrid=False,
            showline=True,
            linecolor='rgb(102, 102, 102)',
            titlefont=dict(
                color='rgb(204, 204, 204)'
            ),
            tickfont=dict(
                color='rgb(102, 102, 102)',
            ),
            autotick=False,
            dtick=10,
            ticks='outside',
            tickcolor='rgb(102, 102, 102)',
        ),
        margin=dict(
            l=140,
            r=40,
            b=50,
            t=80
        ),
        legend=dict(
            font=dict(
                size=10,
            ),
            yanchor='middle',
            xanchor='right',
        ),
        width=800,
```

```
    height=600,
    paper_bgcolor='rgb(254, 247, 234)',
    plot_bgcolor='rgb(254, 247, 234)',
    hovermode='closest',
)
fig = go.Figure(data=data, layout=layout)
py.iplot(fig, filename='lowest-oecd-votes-cast')
.
```



大数据集合

Large Data Sets

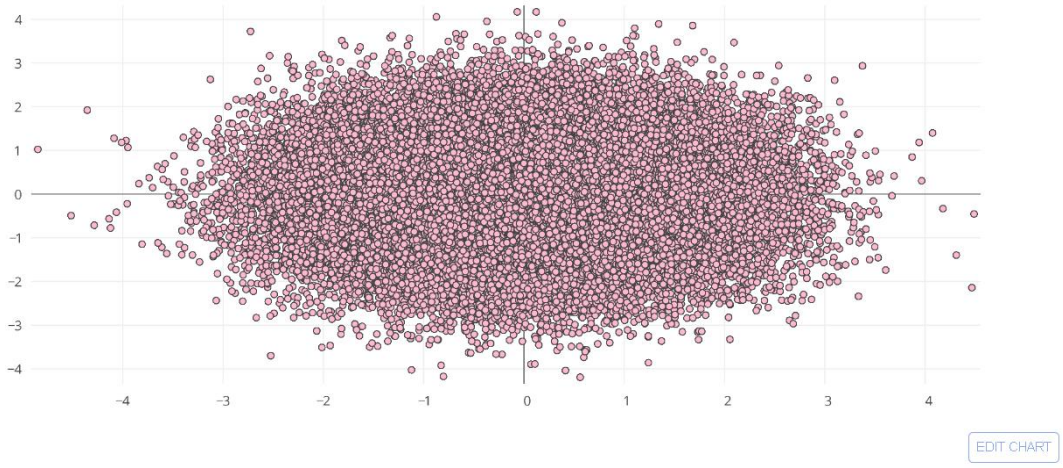
Now in Ploty you can implement WebGL with Scattergl() in place of Scatter() for increased speed, improved interactivity, and the ability to plot even more data!

现在在 Ploy1y,你可以使用 WebGL 的 Scattergl()方法替换 scatter()方法，为提高速度，提高交互性，并有能力绘制更多的数据！

In [7]:

```
import plotly.plotly as py
import plotly.graph_objs as go
import numpy as np

N = 100000
trace = go.Scattergl(
```

```
        x = np.random.randn(N),
        y = np.random.randn(N),
        mode = 'markers',
        marker = dict(
            color = 'FFBAD2',
            line = dict(width = 1)
        )
    )
    data = [trace]
    py.iplot(data, filename='compare_webgl')
    .
```



Reference

参考

See https://plot.ly/python/reference/#scatter or https://plot.ly/python/reference/#scattergl for more information and chart attribute options!

气泡图

Bubble Charts in Python

刚开始接触 Plotly？

New to Plotly?

Plotly's Python library is free and open source! Get started by downloading the client and reading the primer.

You can set up Plotly to work in online or offline mode, or in jupyter notebooks.
We also have a quick-reference cheatsheet (new!) to help you get started!!

Plotly 的 Python 库是免费并且开源的。Get started by downloading the client and reading the primer。

你可以设置 Plotly 以 online 或者 offline 的方式运行，也可以在 jupyter notebooks 中运行。

我们同样也有一个快速参考的 cheatsheet (new!)来帮助新手学习。

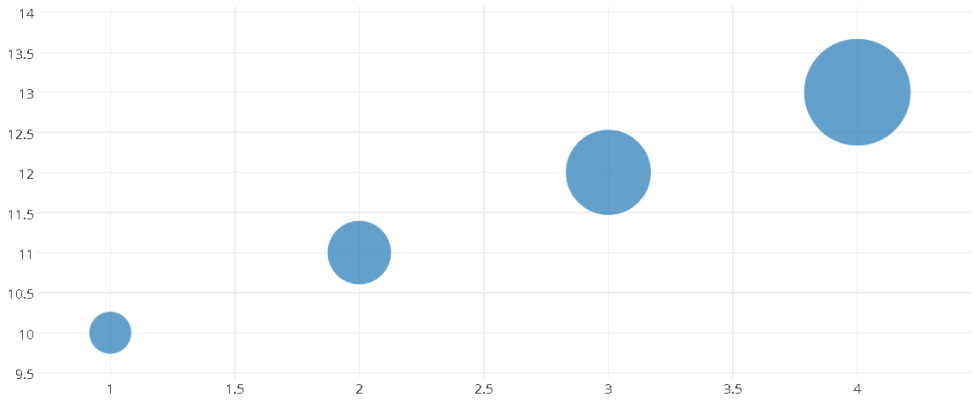
简单气泡图

Simple Bubble Chart

In [1]:

```
import plotly.plotly as py
import plotly.graph_objs as go

trace0 = go.Scatter(
    x=[1, 2, 3, 4],
    y=[10, 11, 12, 13],
    mode='markers',
    marker=dict(
        size=[40, 60, 80, 100],
    )
)

data = [trace0]
py.iplot(data, filename='bubblechart-size')
```

Out[1]:



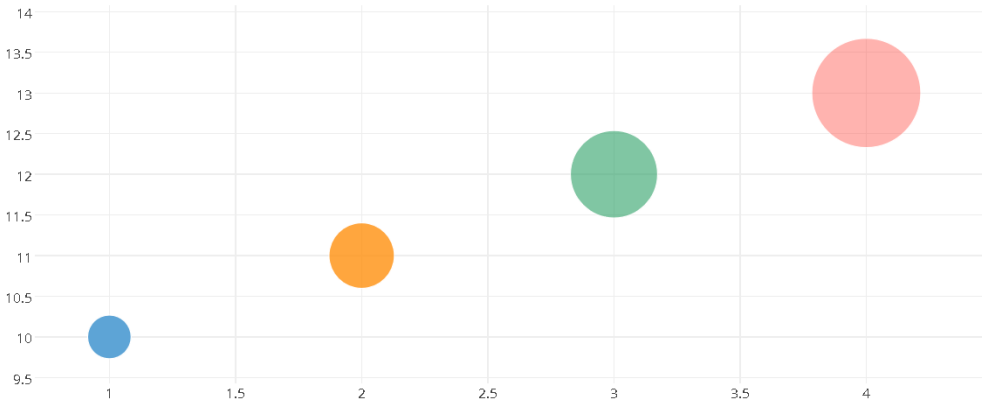
设置标记大小和颜色

Setting Marker Size and Color

In [2]:

```
import plotly.plotly as py
import plotly.graph_objs as go

trace0 = go.Scatter(
    x=[1, 2, 3, 4],
    y=[10, 11, 12, 13],
    mode='markers',
    marker=dict(
        color=['rgb(93, 164, 214)', 'rgb(255, 144, 14)',
               'rgb(44, 160, 101)', 'rgb(255, 65, 54)'],
        opacity=[1, 0.8, 0.6, 0.4],
        size=[40, 60, 80, 100],
    )
)

data = [trace0]
py.iplot(data, filename='bubblechart-color')
```

Out[2]:



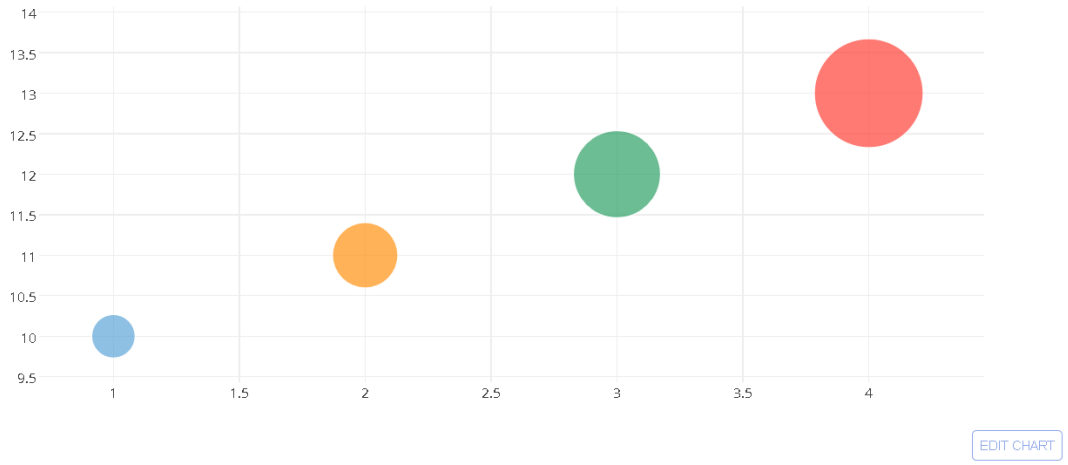
气泡图中的悬浮文本

Hover Text with Bubble Charts

In [3]:

```
import plotly.plotly as py
import plotly.graph_objs as go

trace0 = go.Scatter(
    x=[1, 2, 3, 4],
    y=[10, 11, 12, 13],
    text=['A<br>size: 40', 'B<br>size: 60', 'C<br>size: 80', 'D<br>size: 100'],
    mode='markers',
    marker=dict(
        color=['rgb(93, 164, 214)', 'rgb(255, 144, 14)',  'rgb(44, 160, 101)', 'rgb(255, 65, 54)'],
        size=[40, 60, 80, 100],
    )
)

data = [trace0]
py.iplot(data, filename='bubblechart-text')
```

Out[3]:



可缩放尺寸的气泡图

Scaling the Size of Bubble Charts

Setting 'sizeref' to greater than 1, decreases the rendered marker sizes, while setting 'sizeref' to less than 1, increases the rendered marker sizes. See https://plot.ly/python/reference/#scatter-marker-sizeref for more information.

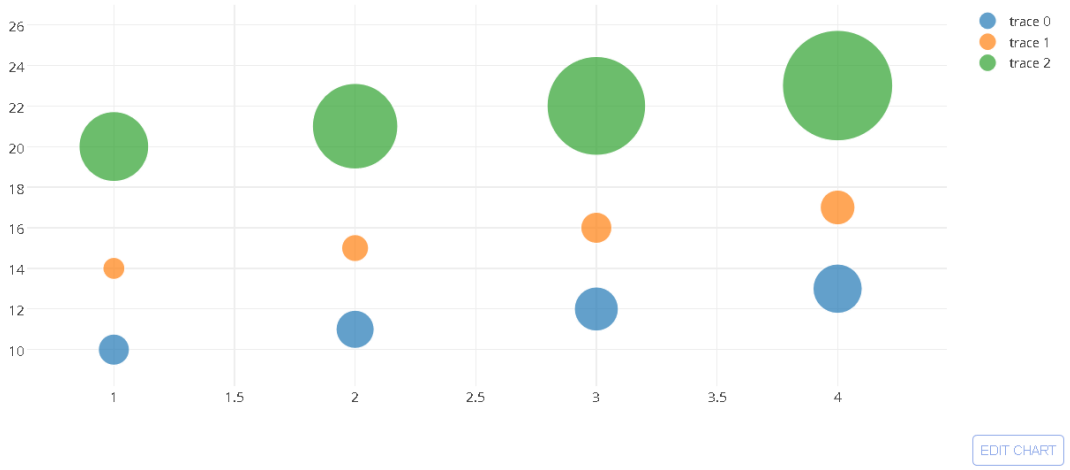
设置“sizeref”大于 1，减少呈现的标记的大小，而设置的 sizeref”小于 1，增加呈现的标记的大小。详情参考页面 https://plot.ly/python/reference/#scatter-marker-sizeref

In [2]:

```
import plotly.plotly as py
import plotly.graph_objs as go
```

```
    trace0 = go.Scatter(
        x=[1, 2, 3, 4],
        y=[10, 11, 12, 13],
        text=['A</br>size: 40</br>default', 'B</br>size: 60</br>default', 'C</br>size: 80</br>default',
'D</br>size: 100</br>default'],
        mode='markers',
        marker=dict(
            size=[400, 600, 800, 1000],
            sizemode='area',
        )
    )
    trace1 = go.Scatter(
        x=[1, 2, 3, 4],
        y=[14, 15, 16, 17],
        text=['A</br>size: 40</br>sixeref: 0.2', 'B</br>size: 60</br>sixeref: 0.2', 'C</br>size:
80</br>sixeref: 0.2', 'D</br>size: 100</br>sixeref: 0.2'],
        mode='markers',
        marker=dict(
            size=[400, 600, 800, 1000],
            sizeref=2,
            sizemode='area',
        )
    )
    trace2 = go.Scatter(
        x=[1, 2, 3, 4],
        y=[20, 21, 22, 23],
        text=['A</br>size: 40</br>sixeref: 2', 'B</br>size: 60</br>sixeref: 2', 'C</br>size:
80</br>sixeref: 2', 'D</br>size: 100</br>sixeref: 2'],
        mode='markers',
        marker=dict(
            size=[400, 600, 800, 1000],
            sizeref=0.2,
            sizemode='area',
        )
    )

    data = [trace0, trace1, trace2]
    py.iplot(data, filename='bubblechart-size-ref')
```

Out[1]:



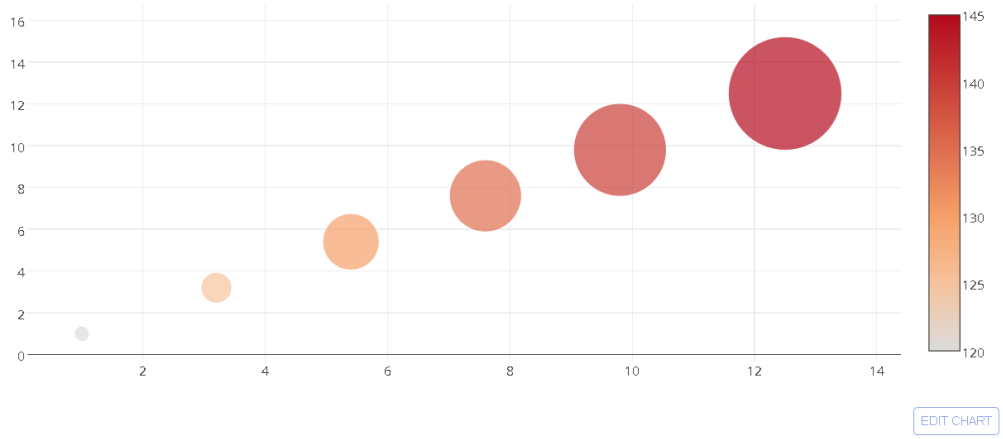
带颜色的气泡图

Bubble Charts with Colorscale

In [2]:

```
import plotly.plotly as py
import plotly.graph_objs as go

data = [
    {
        'x': [1, 3.2, 5.4, 7.6, 9.8, 12.5],
        'y': [1, 3.2, 5.4, 7.6, 9.8, 12.5],
        'mode': 'markers',
        'marker': {
            'color': [120, 125, 130, 135, 140, 145],
            'size': [15, 30, 55, 70, 90, 110],
            'showscale': True
        }
    }
]

py.iplot(data, filename='scatter-colorscale')
```

Out[1]:



分类气泡图

Categorical Bubble Charts

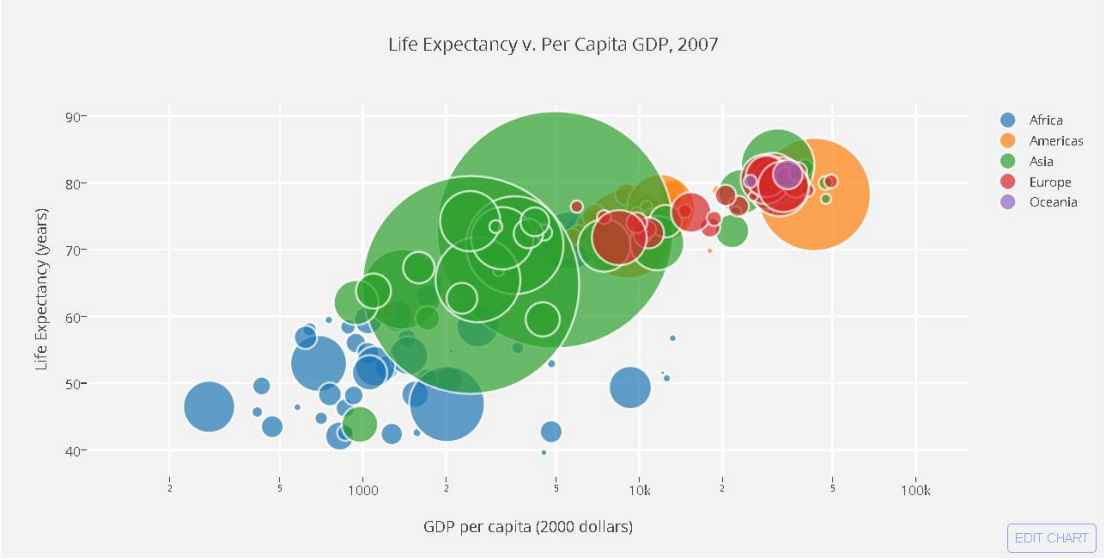
In [2]:

```
import plotly.plotly as py
import plotly.graph_objs as go

import pandas as pd
import math

data =
pd.read_csv("https://raw.githubusercontent.com/plotly/datasets/master/gapminderDataFiveYear.csv")
df_2007 = data[data['year']==2007]
df_2007 = df_2007.sort_values(['continent', 'country'])
slope = 2.666051223553066e-05
hover_text = []
bubble_size = []

for index, row in df_2007.iterrows():
    hover_text.append(('Country: {country}<br>'+
                       'Life Expectancy: {lifeExp}<br>'+
                       'GDP per capita: {gdp}<br>'+
                       'Population: {pop}<br>'+
                       'Year: {year}').format(country=row['country'],
                                              lifeExp=row['lifeExp'],
                                              gdp=row['gdpPercap'],
                                              pop=row['pop'],
                                              year=row['year']))
    bubble_size.append(math.sqrt(row['pop']*slope))

df_2007['text'] = hover_text
df_2007['size'] = bubble_size

trace0 = go.Scatter(
    x=df_2007['gdpPercap'][df_2007['continent'] == 'Africa'],
    y=df_2007['lifeExp'][df_2007['continent'] == 'Africa'],
    mode='markers',
    name='Africa',
    text=df_2007['text'][df_2007['continent'] == 'Africa'],
```

```
        marker=dict(
            symbol='circle',
            sizemode='diameter',
            sizeref=0.85,
            size=df_2007['size'][df_2007['continent'] == 'Africa'],
            line=dict(
                width=2
            ),
        )
    )
    trace1 = go.Scatter(
        x=df_2007['gdpPercap'][df_2007['continent'] == 'Americas'],
        y=df_2007['lifeExp'][df_2007['continent'] == 'Americas'],
        mode='markers',
        name='Americas',
        text=df_2007['text'][df_2007['continent'] == 'Americas'],
        marker=dict(
            sizemode='diameter',
            sizeref=0.85,
            size=df_2007['size'][df_2007['continent'] == 'Americas'],
            line=dict(
                width=2
            ),
        )
    )
    trace2 = go.Scatter(
        x=df_2007['gdpPercap'][df_2007['continent'] == 'Asia'],
        y=df_2007['lifeExp'][df_2007['continent'] == 'Asia'],
        mode='markers',
        name='Asia',
        text=df_2007['text'][df_2007['continent'] == 'Asia'],
        marker=dict(
            sizemode='diameter',
            sizeref=0.85,
            size=df_2007['size'][df_2007['continent'] == 'Asia'],
            line=dict(
                width=2
            ),
        )
    )
    trace3 = go.Scatter(
        x=df_2007['gdpPercap'][df_2007['continent'] == 'Europe'],
        y=df_2007['lifeExp'][df_2007['continent'] == 'Europe'],
        mode='markers',
        name='Europe',
        text=df_2007['text'][df_2007['continent'] == 'Europe'],
        marker=dict(
            sizemode='diameter',
            sizeref=0.85,
            size=df_2007['size'][df_2007['continent'] == 'Europe'],
            line=dict(
                width=2
            ),
        )
    )
    trace4 = go.Scatter(
        x=df_2007['gdpPercap'][df_2007['continent'] == 'Oceania'],
        y=df_2007['lifeExp'][df_2007['continent'] == 'Oceania'],
        mode='markers',
        name='Oceania',
```

```
        text=df_2007['text'][df_2007['continent'] == 'Oceania'],
        marker=dict(
            sizemode='diameter',
            sizeref=0.85,
            size=df_2007['size'][df_2007['continent'] == 'Oceania'],
            line=dict(
                width=2
            ),
        )
    )

    data = [trace0, trace1, trace2, trace3, trace4]
    layout = go.Layout(
        title='Life Expectancy v. Per Capita GDP, 2007',
        xaxis=dict(
            title='GDP per capita (2000 dollars)',
            gridcolor='rgb(255, 255, 255)',
            range=[2.003297660701705, 5.191505530708712],
            type='log',
            zerolinewidth=1,
            ticklen=5,
            gridwidth=2,
        ),
        yaxis=dict(
            title='Life Expectancy (years)',
            gridcolor='rgb(255, 255, 255)',
            range=[36.12621671352166, 91.72921793264332],
            zerolinewidth=1,
            ticklen=5,
            gridwidth=2,
        ),
        paper_bgcolor='rgb(243, 243, 243)',
        plot_bgcolor='rgb(243, 243, 243)',
    )

    fig = go.Figure(data=data, layout=layout)
    py.iplot(fig, filename='life-expectancy-per-GDP-2007')
```

Out[1]:



参考

Reference

See https://plot.ly/python/reference/#scatter for more information and chart attribute options!

线条图

Line Charts

简单的折线图

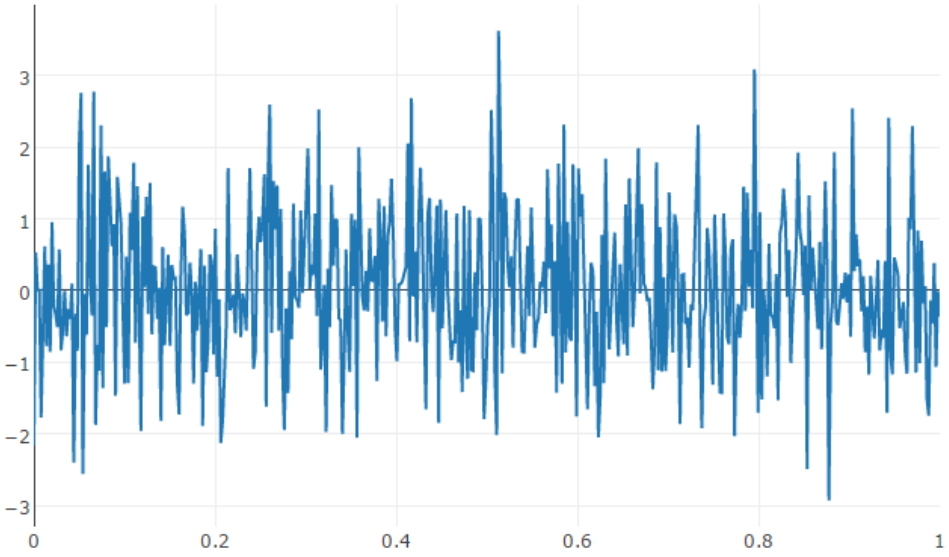
Simple Line Plot

In [2]:

```
import plotly.plotly as py
import plotly.graph_objs as go

# Create random data with numpy
import numpy as np

N = 500
random_x = np.linspace(0, 1, N)
random_y = np.random.randn(N)

# Create a trace
trace = go.Scatter(
    x = random_x,
    y = random_y
)

data = [trace]
```

```
# Plot and embed in ipython notebook!
py.iplot(data, filename='basic-line')

# or plot with: plot_url = py.plot(data, filename='basic-line')
```

Out[2]:



设置划线模式

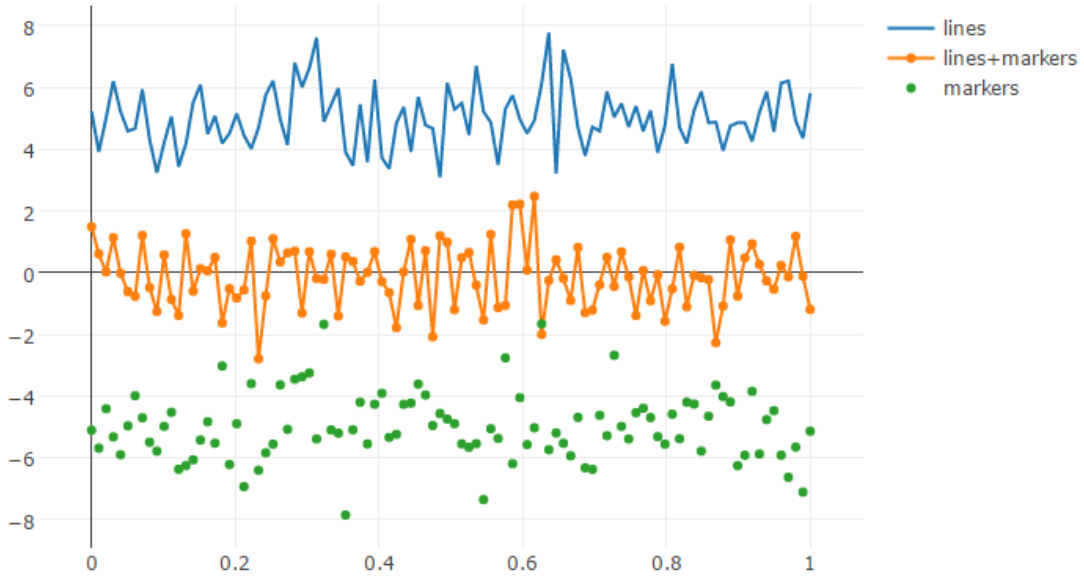
Line Plot Modes

In [2]:

```
import plotly.plotly as py
import plotly.graph_objs as go

# Create random data with numpy
import numpy as np

N = 100
random_x = np.linspace(0, 1, N)
random_y0 = np.random.randn(N)+5
random_y1 = np.random.randn(N)
random_y2 = np.random.randn(N)-5

# Create traces
```

```
trace0 = go.Scatter(
    x = random_x,
    y = random_y0,
    mode = 'lines',
    name = 'lines'
)
trace1 = go.Scatter(
    x = random_x,
    y = random_y1,
    mode = 'lines+markers',
    name = 'lines+markers'
)
trace2 = go.Scatter(
    x = random_x,
    y = random_y2,
    mode = 'markers',
    name = 'markers'
)
data = [trace0, trace1, trace2]

# Plot and embed in ipython notebook!
py.iplot(data, filename='line-mode')
```

Out[2]:



设置画线类型

Style Line Plots

This example styles the color and dash of the traces, adds trace names, modifiys line width, and adds plot and axes titles

这一示例教你如何改变线的颜色、虚实、给线命名、设置线宽，以及增加图和坐标的标题

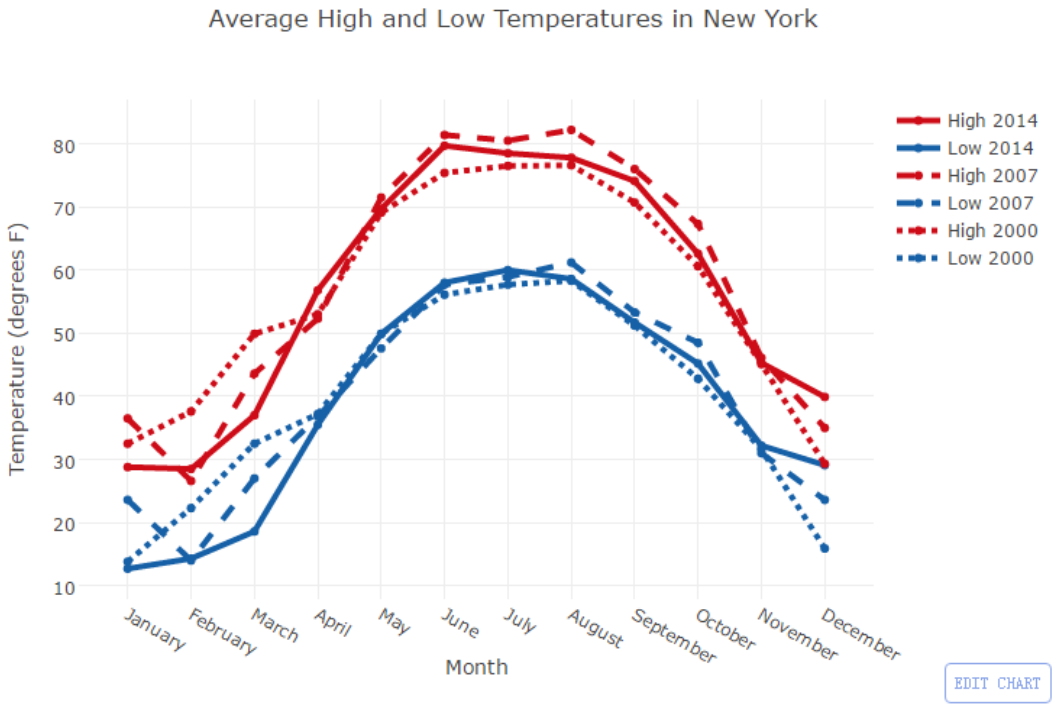
In [2]:

```
import plotly.plotly as py
import plotly.graph_objs as go

# Add data
month = ['January', 'February', 'March', 'April', 'May', 'June', 'July',
         'August', 'September', 'October', 'November', 'December']
high_2000 = [32.5, 37.6, 49.9, 53.0, 69.1, 75.4, 76.5, 76.6, 70.7, 60.6, 45.1, 29.3]
low_2000 = [13.8, 22.3, 32.5, 37.2, 49.9, 56.1, 57.7, 58.3, 51.2, 42.8, 31.6, 15.9]
high_2007 = [36.5, 26.6, 43.6, 52.3, 71.5, 81.4, 80.5, 82.2, 76.0, 67.3, 46.1, 35.0]
low_2007 = [23.6, 14.0, 27.0, 36.8, 47.6, 57.7, 58.9, 61.2, 53.3, 48.5, 31.0, 23.6]
high_2014 = [28.8, 28.5, 37.0, 56.8, 69.7, 79.7, 78.5, 77.8, 74.1, 62.6, 45.3, 39.9]
low_2014 = [12.7, 14.3, 18.6, 35.5, 49.9, 58.0, 60.0, 58.6, 51.7, 45.2, 32.2, 29.1]

# Create and style traces
trace0 = go.Scatter(
    x = month,
    y = high_2014,
    name = 'High 2014',
    line = dict(
        color = ('rgb(205, 12, 24)'),
        width = 4)
)
trace1 = go.Scatter(
    x = month,
    y = low_2014,
    name = 'Low 2014',
    line = dict(
        color = ('rgb(22, 96, 167)'),
```

```
        width = 4,)
)
trace2 = go.Scatter(
    x = month,
    y = high_2007,
    name = 'High 2007',
    line = dict(
        color = ('rgb(205, 12, 24)'),
        width = 4,
        dash = 'dash') # dash options include 'dash', 'dot', and 'dashdot'
)
trace3 = go.Scatter(
    x = month,
    y = low_2007,
    name = 'Low 2007',
    line = dict(
        color = ('rgb(22, 96, 167)'),
        width = 4,
        dash = 'dash')
)
trace4 = go.Scatter(
    x = month,
    y = high_2000,
    name = 'High 2000',
    line = dict(
        color = ('rgb(205, 12, 24)'),
        width = 4,
        dash = 'dot')
)
trace5 = go.Scatter(
    x = month,
    y = low_2000,
    name = 'Low 2000',
    line = dict(
        color = ('rgb(22, 96, 167)'),
        width = 4,
        dash = 'dot')
)
data = [trace0, trace1, trace2, trace3, trace4, trace5]
```

```
# Edit the layout
layout = dict(title = 'Average High and Low Temperatures in New York',
              xaxis = dict(title = 'Temperature (degrees F)'),
              yaxis = dict(title = 'Month'),
              )

# Plot and embed in ipython notebook!
fig = dict(data=data, layout=layout)
py.iplot(fig, filename='styled-line')
```

Out[2]:



连接数据缺口

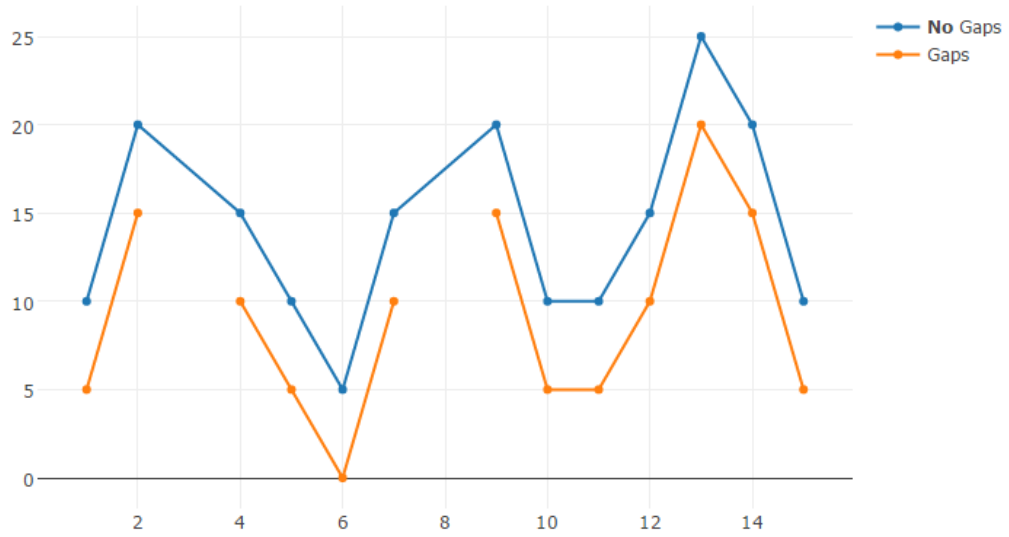
Connect Data Gaps

In [2]:

```
import plotly.plotly as py
import plotly.graph_objs as go

trace1 = go.Scatter(
   x=[1, 2, 3, 4, 5,
```

```
        6, 7, 8, 9, 10,
        11, 12, 13, 14, 15],
    y=[10, 20, None, 15, 10,
        5, 15, None, 20, 10,
        10, 15, 25, 20, 10],
    name = '<b>No</b> Gaps', # Style name/legend entry with html tags
    connectgaps=True
)
trace2 = go.Scatter(
    x=[1, 2, 3, 4, 5,
        6, 7, 8, 9, 10,
        11, 12, 13, 14, 15],
    y=[5, 15, None, 10, 5,
        0, 10, None, 15, 5,
        5, 10, 20, 15, 5],
    name = 'Gaps',
)

data = [trace1, trace2]

fig = dict(data=data)
py.iplot(fig, filename='simple-connectgaps')
```

Out[2]:



修改折线图

Interpolation with Line Plots

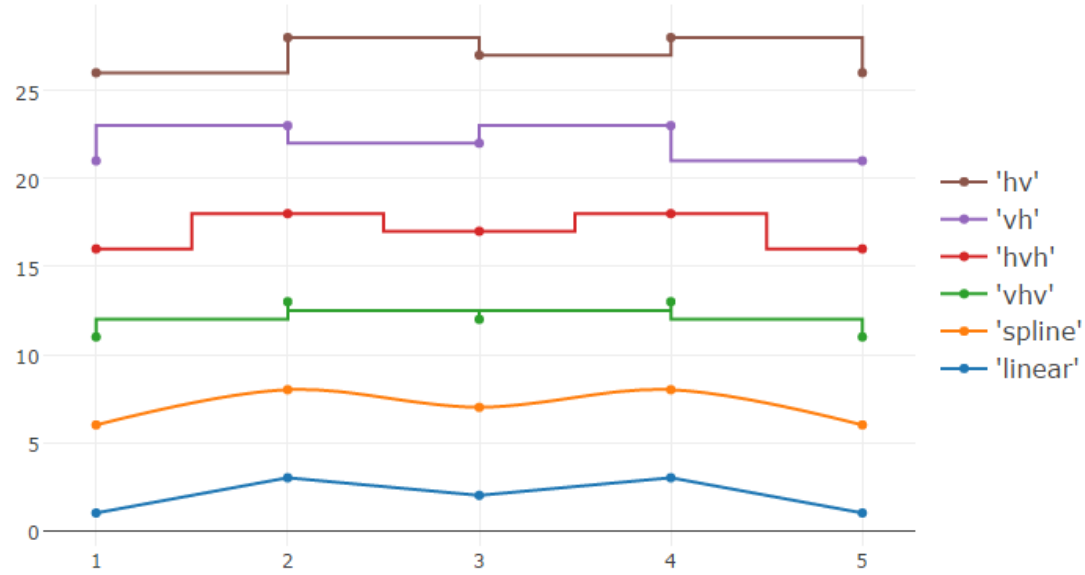
In [2]:

```
import plotly.plotly as py
import plotly.graph_objs as go

trace1 = go.Scatter(
    x=[1, 2, 3, 4, 5],
    y=[1, 3, 2, 3, 1],
    mode='lines+markers',
    name="'linear'",
    hoverinfo='name',
    line=dict(
        shape='linear'
    )
)
trace2 = go.Scatter(
    x=[1, 2, 3, 4, 5],
    y=[6, 8, 7, 8, 6],
    mode='lines+markers',
    name="'spline'",
    text=["tweak line smoothness<br>with 'smoothing' in line object"],
    hoverinfo='text+name',
    line=dict(
        shape='spline'
    )
)
trace3 = go.Scatter(
    x=[1, 2, 3, 4, 5],
    y=[11, 13, 12, 13, 11],
    mode='lines+markers',
    name="'vhv'",
    hoverinfo='name',
    line=dict(
        shape='vhv'
    )
)
```

```
trace4 = go.Scatter(
    x=[1, 2, 3, 4, 5],
    y=[16, 18, 17, 18, 16],
    mode='lines+markers',
    name="'hvh'",
    hoverinfo='name',
    line=dict(
        shape='hvh'
    )
)
trace5 = go.Scatter(
    x=[1, 2, 3, 4, 5],
    y=[21, 23, 22, 23, 21],
    mode='lines+markers',
    name="'vh'",
    hoverinfo='name',
    line=dict(
        shape='vh'
    )
)
trace6 = go.Scatter(
    x=[1, 2, 3, 4, 5],
    y=[26, 28, 27, 28, 26],
    mode='lines+markers',
    name="'hv'",
    hoverinfo='name',
    line=dict(
        shape='hv'
    )
)
data = [trace1, trace2, trace3, trace4, trace5, trace6]
layout = dict(
    legend=dict(
        y=0.5,
        traceorder='reversed',
        font=dict(
            size=16
        )
    )
)
```

```
fig = dict(data=data, layout=layout)
py.iplot(fig, filename='line-shapes')
```

Out[2]:



给线增加标注

Label Lines with Annotations

In [2]:

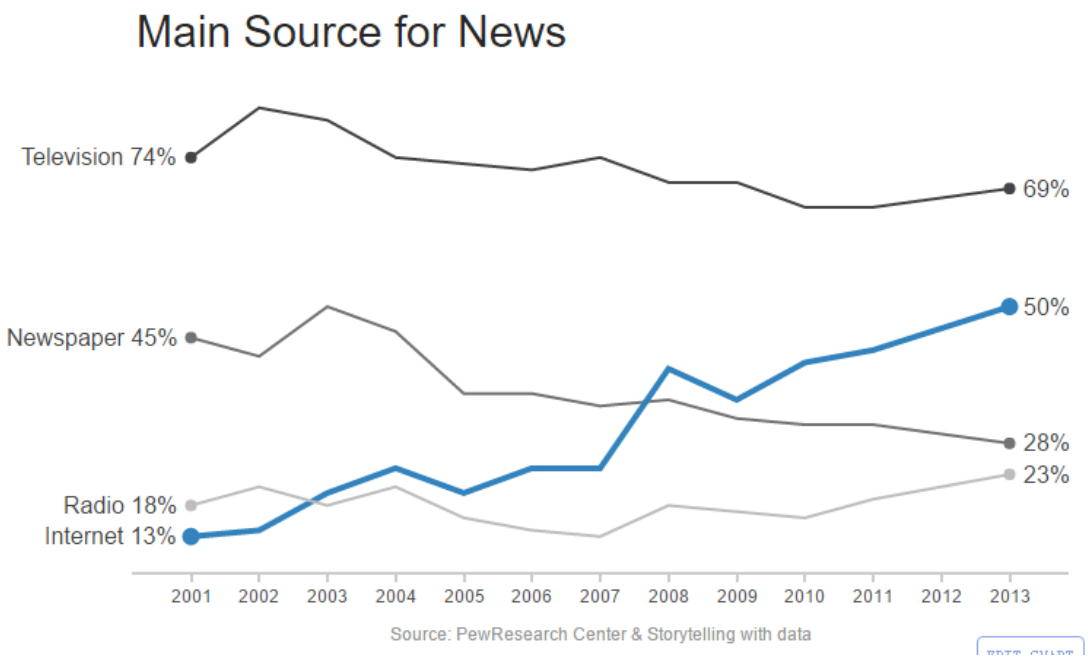
```
import plotly.plotly as py
import plotly.graph_objs as go

title = 'Main Source for News'

labels = ['Television', 'Newspaper', 'Internet', 'Radio']

colors = ['rgba(67,67,67,1)', 'rgba(115,115,115,1)', 'rgba(49,130,189, 1)',
'rgba(189,189,189,1)']

mode_size = [8, 8, 12, 8]

line_size = [2, 2, 4, 2]

x_data = [
```

```
    [2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2013],
    [2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2013],
    [2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2013],
    [2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2013],
]

y_data = [
    [74, 82, 80, 74, 73, 72, 74, 70, 70, 66, 66, 69],
    [45, 42, 50, 46, 36, 36, 34, 35, 32, 31, 31, 28],
    [13, 14, 20, 24, 20, 24, 24, 40, 35, 41, 43, 50],
    [18, 21, 18, 21, 16, 14, 13, 18, 17, 16, 19, 23],
]

traces = []

for i in range(0, 4):
    traces.append(go.Scatter(
        x=x_data[i],
        y=y_data[i],
        mode='lines',
        line=dict(color=colors[i], width=line_size[i]),
        connectgaps=True,
    ))

    traces.append(go.Scatter(
        x=[x_data[i][0], x_data[i][11]],
        y=[y_data[i][0], y_data[i][11]],
        mode='markers',
        marker=dict(color=colors[i], size=mode_size[i])
    ))

layout = go.Layout(
    xaxis=dict(
        showline=True,
        showgrid=False,
        showticklabels=True,
        linecolor='rgb(204, 204, 204)',
        linewidth=2,
        autotick=False,
        ticks='outside',
```

```
            tickcolor='rgb(204, 204, 204)',
            tickwidth=2,
            ticklen=5,
            tickfont=dict(
                family='Arial',
                size=12,
                color='rgb(82, 82, 82)',
            ),
        ),
        yaxis=dict(
            showgrid=False,
            zeroline=False,
            showline=False,
            showticklabels=False,
        ),
        autosize=False,
        margin=dict(
            autoexpand=False,
            l=100,
            r=20,
            t=110,
        ),
        showlegend=False,
    )

    annotations = []

    # Adding labels
    for y_trace, label, color in zip(y_data, labels, colors):
        # labeling the left_side of the plot
        annotations.append(dict(xref='paper', x=0.05, y=y_trace[0],
                                xanchor='right', yanchor='middle',
                                text=label + ' {}%'.format(y_trace[0]),
                                font=dict(family='Arial',
                                          size=16,
                                          color=colors,),
                                showarrow=False))
        # labeling the right_side of the plot
        annotations.append(dict(xref='paper', x=0.95, y=y_trace[11],
                                xanchor='left', yanchor='middle',
```

```
                                text='{}%'.format(y_trace[11]),
                                font=dict(family='Arial',
                                          size=16,
                                          color=colors,),
                                showarrow=False))
# Title
annotations.append(dict(xref='paper', yref='paper', x=0.0, y=1.05,
                                xanchor='left', yanchor='bottom',
                                text='Main Source for News',
                                font=dict(family='Arial',
                                          size=30,
                                          color='rgb(37,37,37)'),
                                showarrow=False))
# Source
annotations.append(dict(xref='paper', yref='paper', x=0.5, y=-0.1,
                                xanchor='center', yanchor='top',
                                text='Source: PewResearch Center & ' +
                                    'Storytelling with data',
                                font=dict(family='Arial',
                                          size=12,
                                          color='rgb(150,150,150)'),
                                showarrow=False))

layout['annotations'] = annotations

fig = go.Figure(data=traces, layout=layout)
py.iplot(fig, filename='news-source')
```

Out[2]:



填充线

Filled Lines

In [2]:

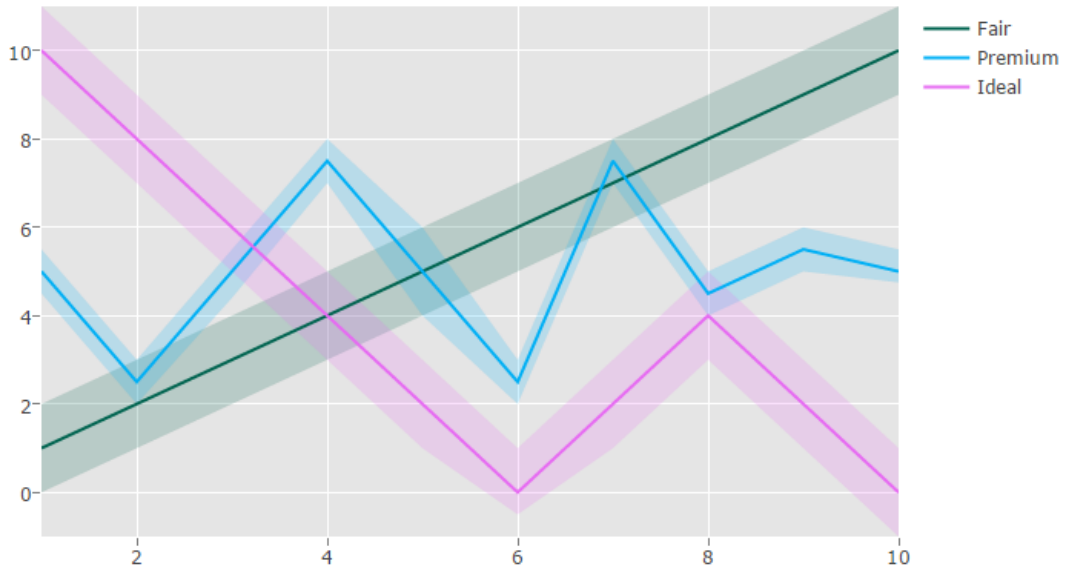
```
import plotly.plotly as py
from plotly.graph_objs import *

x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
x_rev = x[::-1]

# Line 1
y1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
y1_upper = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
y1_lower = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
y1_lower = y1_lower[::-1]

# Line 2
y2 = [5, 2.5, 5, 7.5, 5, 2.5, 7.5, 4.5, 5.5, 5]
y2_upper = [5.5, 3, 5.5, 8, 6, 3, 8, 5, 6, 5.5]
y2_lower = [4.5, 2, 4.4, 7, 4, 2, 7, 4, 5, 4.75]
y2_lower = y2_lower[::-1]
```

```
# Line 3
y3 = [10, 8, 6, 4, 2, 0, 2, 4, 2, 0]
y3_upper = [11, 9, 7, 5, 3, 1, 3, 5, 3, 1]
y3_lower = [9, 7, 5, 3, 1, -.5, 1, 3, 1, -1]
y3_lower = y3_lower[::-1]

trace1 = Scatter(
    x=x+x_rev,
    y=y1_upper+y1_lower,
    fill='tozerox',
    fillcolor='rgba(0,100,80,0.2)',
    line=Line(color='transparent'),
    showlegend=False,
    name='Fair',
)
trace2 = Scatter(
    x=x+x_rev,
    y=y2_upper+y2_lower,
    fill='tozerox',
    fillcolor='rgba(0,176,246,0.2)',
    line=Line(color='transparent'),
    name='Premium',
    showlegend=False,
)
trace3 = Scatter(
    x=x+x_rev,
    y=y3_upper+y3_lower,
    fill='tozerox',
    fillcolor='rgba(231,107,243,0.2)',
    line=Line(color='transparent'),
    showlegend=False,
    name='Fair',
)
trace4 = Scatter(
    x=x,
    y=y1,
    line=Line(color='rgb(0,100,80)'),
    mode='lines',
    name='Fair',
)
```

```
trace5 = Scatter(
    x=x,
    y=y2,
    line=Line(color='rgb(0,176,246)'),
    mode='lines',
    name='Premium',
)
trace6 = Scatter(
    x=x,
    y=y3,
    line=Line(color='rgb(231,107,243)'),
    mode='lines',
    name='Ideal',
)

data = Data([trace1, trace2, trace3, trace4, trace5, trace6])

layout = Layout(
    paper_bgcolor='rgb(255,255,255)',
    plot_bgcolor='rgb(229,229,229)',
    xaxis=XAxis(
        gridcolor='rgb(255,255,255)',
        range=[1,10],
        showgrid=True,
        showline=False,
        showticklabels=True,
        tickcolor='rgb(127,127,127)',
        ticks='outside',
        zeroline=False
    ),
    yaxis=YAxis(
        gridcolor='rgb(255,255,255)',
        showgrid=True,
        showline=False,
        showticklabels=True,
        tickcolor='rgb(127,127,127)',
        ticks='outside',
        zeroline=False
    ),
)
```

```
fig = Figure(data=data, layout=layout)
py.iplot(fig, filename= 'shaded_lines')
```

Out[2]:



柱状图

Bar Chart

How to make Bar Charts in Python with Plotly

如何使用 Plotly 在 python 中使用柱状图

基本柱状图绘制

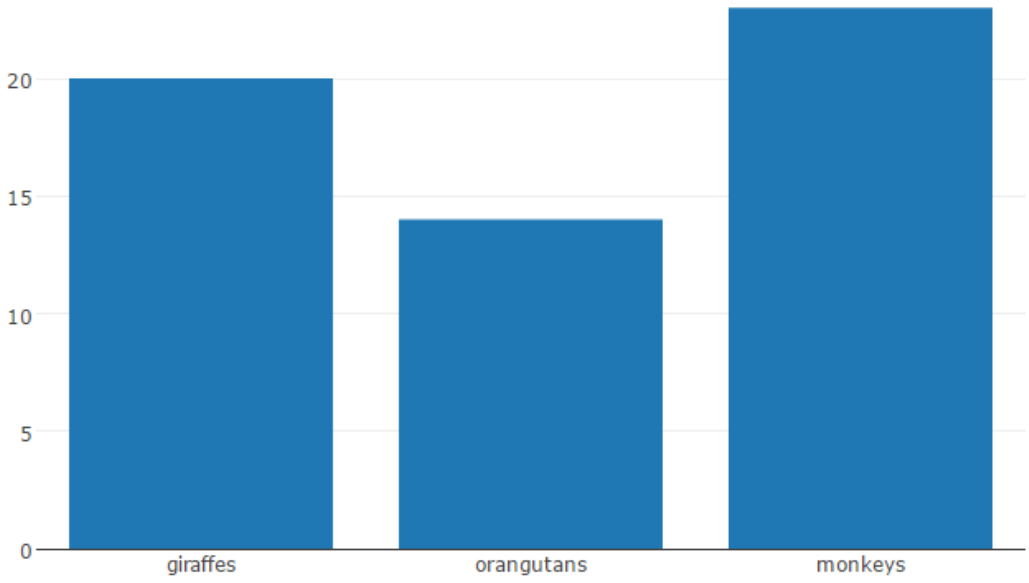
Basic Bar Chart

In [2]:

```
import plotly.plotly as py
import plotly.graph_objs as go
```

```
data = [go.Bar(
            x=['giraffes', 'orangutans', 'monkeys'],
            y=[20, 14, 23]
    )]

py.iplot(data, filename='basic-bar')
```

Out[2]:



柱状簇绘制

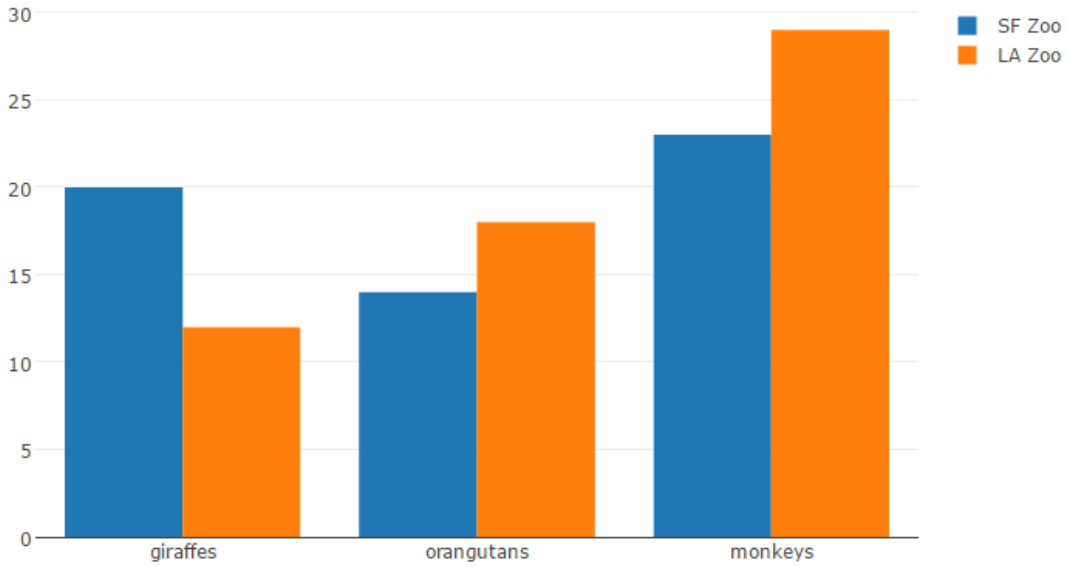
Grouped Bar Chart

In [2]:

```
import plotly.plotly as py
import plotly.graph_objs as go

trace1 = go.Bar(
    x=['giraffes', 'orangutans', 'monkeys'],
    y=[20, 14, 23],
    name='SF Zoo'
)
trace2 = go.Bar(
    x=['giraffes', 'orangutans', 'monkeys'],
```

```
    y=[12, 18, 29],
    name='LA Zoo'
)

data = [trace1, trace2]
layout = go.Layout(
    barmode='group'
)

fig = go.Figure(data=data, layout=layout)
py.iplot(fig, filename='grouped-bar')
```

Out[2]:



层叠柱状图绘制

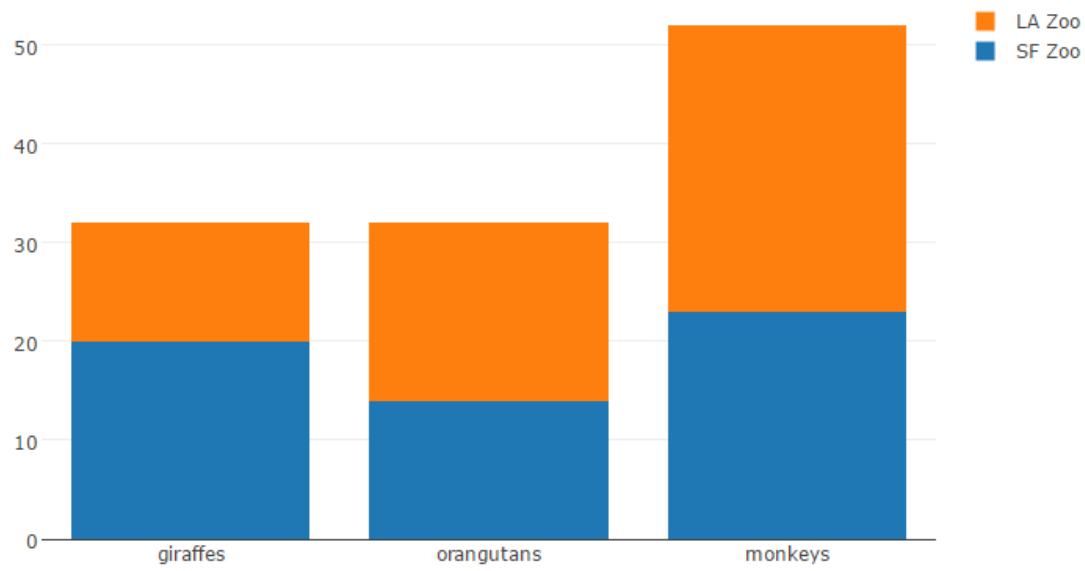
Stacked Bar Chart

In [2]:

```
import plotly.plotly as py
import plotly.graph_objs as go

trace1 = go.Bar(
    x=['giraffes', 'orangutans', 'monkeys'],
    y=[20, 14, 23],
    name='SF Zoo'
```

```
)
trace2 = go.Bar(
    x=['giraffes', 'orangutans', 'monkeys'],
    y=[12, 18, 29],
    name='LA Zoo'
)

data = [trace1, trace2]
layout = go.Layout(
    barmode='stack'
)

fig = go.Figure(data=data, layout=layout)
py.iplot(fig, filename='stacked-bar')
```

Out[2]:



带有悬浮文字的柱状图

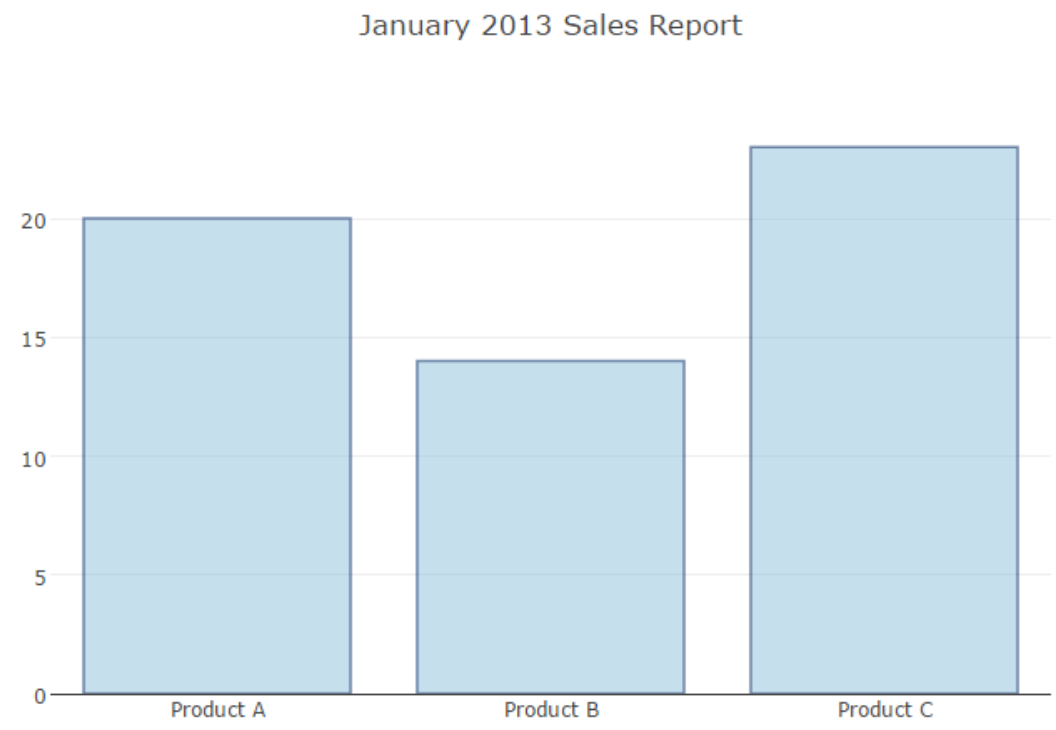
Bar Chart with Hover Text

In [2]:

```
import plotly.plotly as py
import plotly.graph_objs as go

trace0 = go.Bar(
```

```
    x=['Product A', 'Product B', 'Product C'],
    y=[20, 14, 23],
    text=['27% market share', '24% market share', '19% market share'],
    marker=dict(
        color='rgb(158,202,225)',
        line=dict(
            color='rgb(8,48,107)',
            width=1.5,
        )
    ),
    opacity=0.6
)

data = [trace0]
layout = go.Layout(
    title='January 2013 Sales Report',
)

fig = go.Figure(data=data, layout=layout)
py.iplot(fig, filename='text-hover-bar')
```

Out[2]:



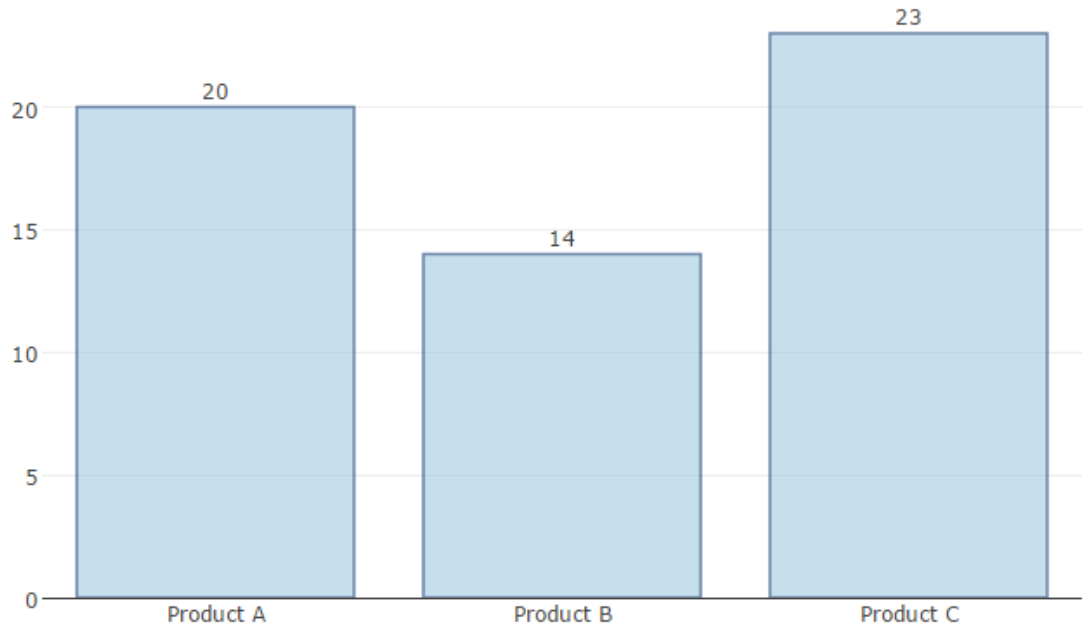
直接标记的柱状图

Styled Bar Chart with Direct Labels

In [2]:

```
import plotly.plotly as py
import plotly.graph_objs as go

x = ['Product A', 'Product B', 'Product C']
y = [20, 14, 23]

data = [go.Bar(
            x=x,y=y,
            marker=dict(
                color='rgb(158,202,225)',
                line=dict(
                    color='rgb(8,48,107)',
                    width=1.5),
            ),
            opacity=0.6
        )]

layout = go.Layout(
    annotations=[
        dict(x=xi,y=yi,
             text=str(yi),
             xanchor='center',
             yanchor='bottom',
             showarrow=False,
        ) for xi, yi in zip(x, y)]
)

fig = go.Figure(data=data, layout=layout)
py.iplot(fig, filename='bar-direct-labels')
```

Out[2]:



旋转柱状图标记

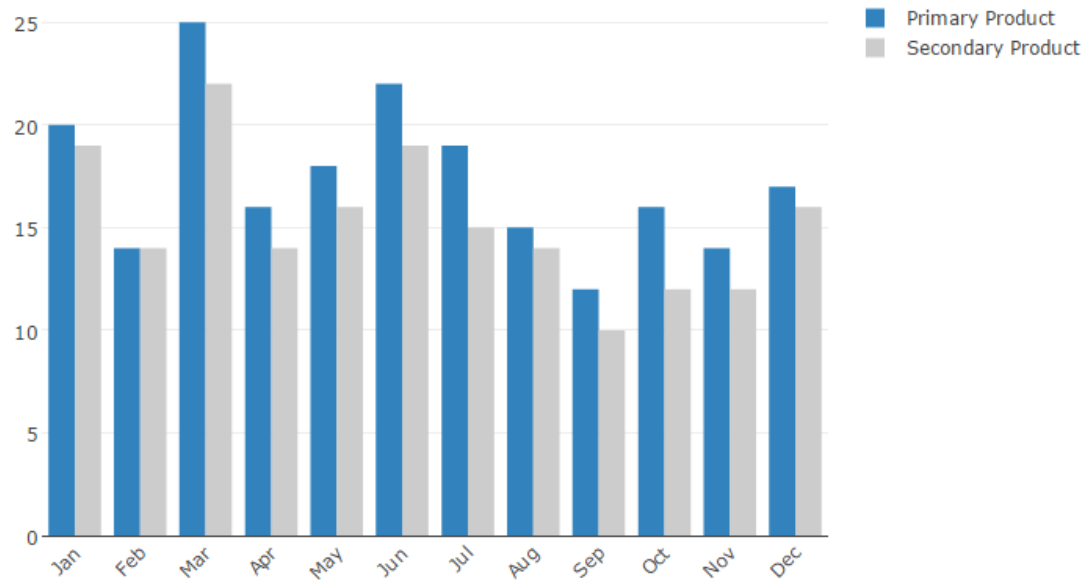
Rotated Bar Chart Labels

In [2]:

```
import plotly.plotly as py
import plotly.graph_objs as go

trace0 = go.Bar(
    x=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
       'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'],
    y=[20, 14, 25, 16, 18, 22, 19, 15, 12, 16, 14, 17],
    name='Primary Product',
    marker=dict(
        color='rgb(49,130,189)'
    )
)
trace1 = go.Bar(
    x=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
       'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'],
    y=[19, 14, 22, 14, 16, 19, 15, 14, 10, 12, 12, 16],
    name='Secondary Product',
    marker=dict(
```

```
        color='rgb(204,204,204)',
    )
)

data = [trace0, trace1]
layout = go.Layout(
    xaxis=dict(tickangle=-45),
    barmode='group',
)

fig = go.Figure(data=data, layout=layout)
py.iplot(fig, filename='angled-text-bar')
```

Out[2]:



指定某一柱体颜色

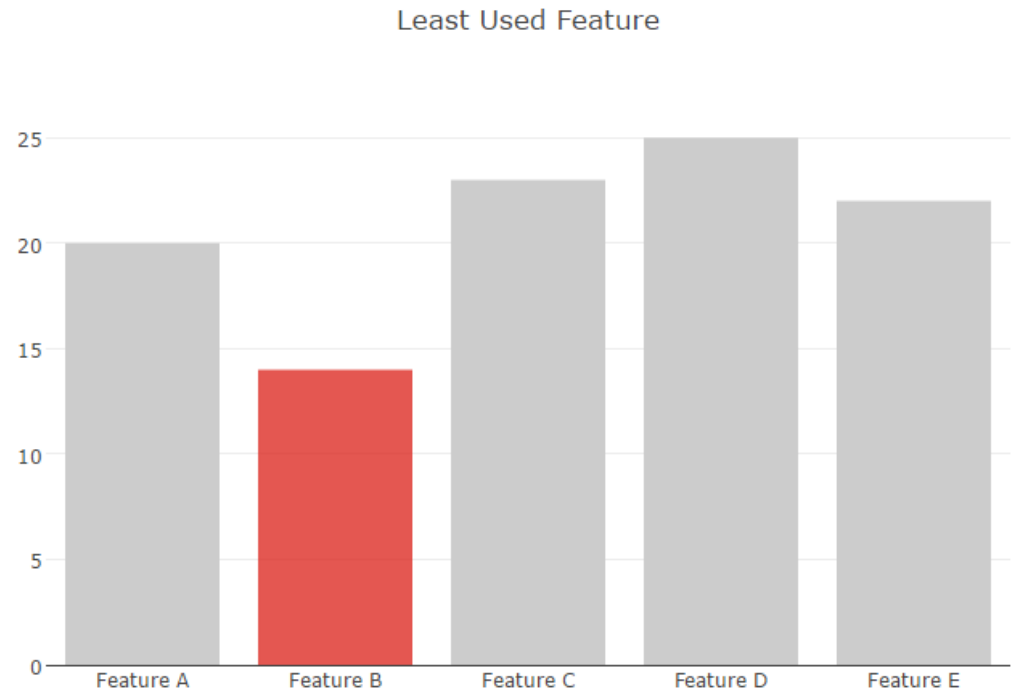
Customizing Individual Bar Colors

In [2]:

```
import plotly.plotly as py
import plotly.graph_objs as go

trace0 = go.Bar(
    x=['Feature A', 'Feature B', 'Feature C',
       'Feature D', 'Feature E'],
    y=[20, 14, 23, 25, 22],
```

```
    marker=dict(
        color=['rgba(204,204,204,1)', 'rgba(222,45,38,0.8)',
               'rgba(204,204,204,1)', 'rgba(204,204,204,1)',
               'rgba(204,204,204,1)']),
)

data = [trace0]
layout = go.Layout(
    title='Least Used Feature',
)

fig = go.Figure(data=data, layout=layout)
py.iplot(fig, filename='color-bar')
```

Out[2]:



设置图的颜色和格式

Colored and Styled Bar Chart

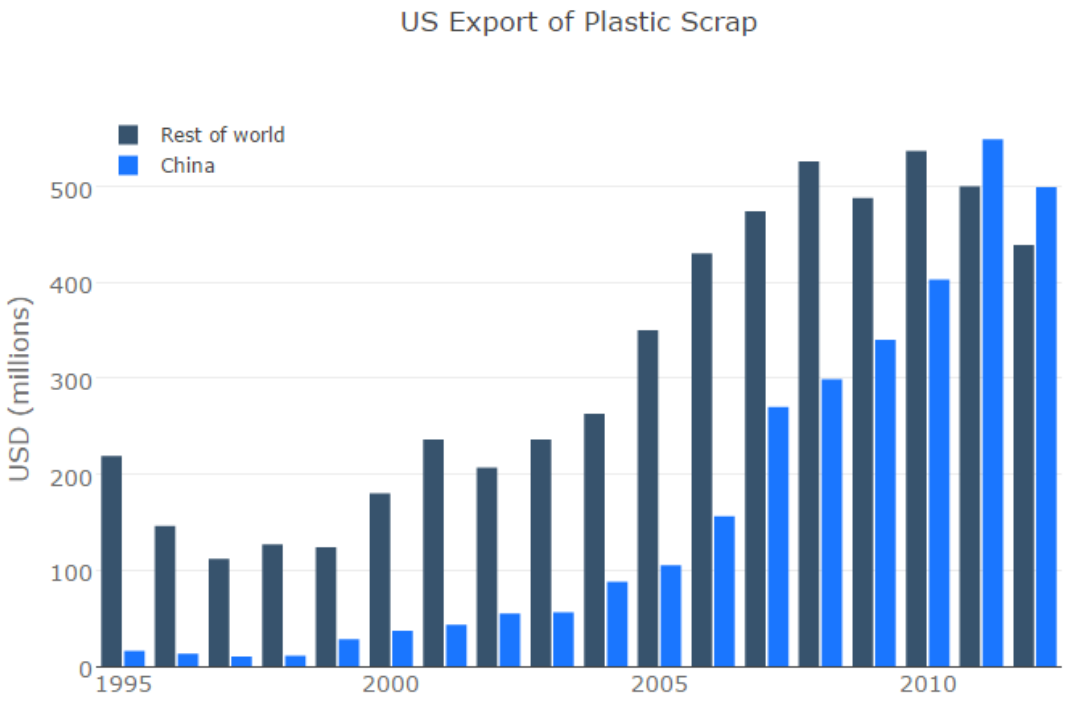
In [2]:

```
import plotly.plotly as py
import plotly.graph_objs as go

trace1 = go.Bar(
```

```
    x=[1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003,
       2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012],
    y=[219, 146, 112, 127, 124, 180, 236, 207, 236, 263,
       350, 430, 474, 526, 488, 537, 500, 439],
    name='Rest of world',
    marker=dict(
        color='rgb(55, 83, 109)'
    )
)
trace2 = go.Bar(
    x=[1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003,
       2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012],
    y=[16, 13, 10, 11, 28, 37, 43, 55, 56, 88, 105, 156, 270,
       299, 340, 403, 549, 499],
    name='China',
    marker=dict(
        color='rgb(26, 118, 255)'
    )
)
data = [trace1, trace2]
layout = go.Layout(
    title='US Export of Plastic Scrap',
    xaxis=dict(
        tickfont=dict(
            size=14,
            color='rgb(107, 107, 107)'
        )
    ),
    yaxis=dict(
        title='USD (millions)',
        titlefont=dict(
            size=16,
            color='rgb(107, 107, 107)'
        ),
        tickfont=dict(
            size=14,
            color='rgb(107, 107, 107)'
        )
    ),
    legend=dict(
```

```
        x=0,
        y=1.0,
        bgcolor='rgba(255, 255, 255, 0)',
        bordercolor='rgba(255, 255, 255, 0)'
    ),
    barmode='group',
    bargap=0.15,
    bargroupgap=0.1
)

fig = go.Figure(data=data, layout=layout)
py.iplot(fig, filename='style-bar')
```

Out[2]:



瀑布式柱状图

Waterfall Bar Chart

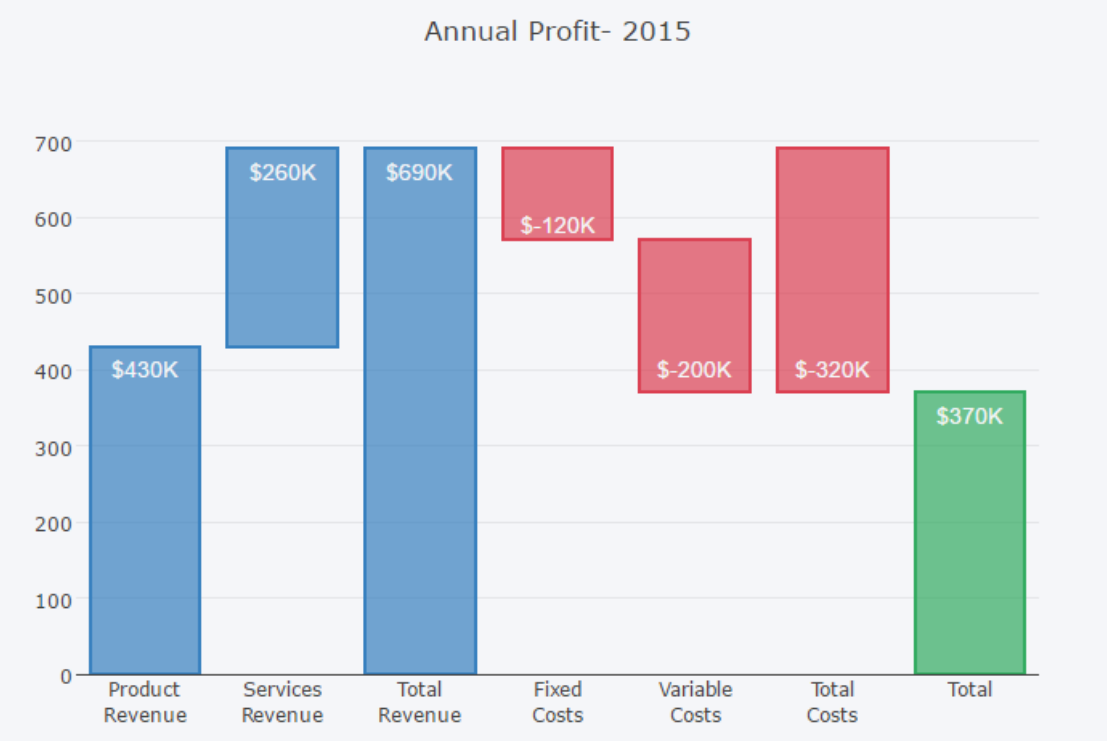
In [2]:

```
import plotly.plotly as py
import plotly.graph_objs as go

x_data = ['Product<br>Revenue', 'Services<br>Revenue',
          'Total<br>Revenue', 'Fixed<br>Costs',
```

```
        'Variable<br>Costs', 'Total<br>Costs', 'Total']
y_data = [400, 660, 660, 590, 400, 400, 340]
text = ['$430K', '$260K', '$690K', '$-120K', '$-200K', '$-320K', '$370K']

# Base
trace0 = go.Bar(
    x=x_data,
    y=[0, 430, 0, 570, 370, 370, 0],
    marker=dict(
        color='rgba(1,1,1, 0.0)',
    )
)
# Revenue
trace1 = go.Bar(
    x=x_data,
    y=[430, 260, 690, 0, 0, 0, 0],
    marker=dict(
        color='rgba(55, 128, 191, 0.7)',
        line=dict(
            color='rgba(55, 128, 191, 1.0)',
            width=2,
        )
    )
)
# Costs
trace2 = go.Bar(
    x=x_data,
    y=[0, 0, 0, 120, 200, 320, 0],
    marker=dict(
        color='rgba(219, 64, 82, 0.7)',
        line=dict(
            color='rgba(219, 64, 82, 1.0)',
            width=2,
        )
    )
)
# Profit
trace3 = go.Bar(
    x=x_data,
    y=[0, 0, 0, 0, 0, 0, 370],
```

```
        marker=dict(
            color='rgba(50, 171, 96, 0.7)',
            line=dict(
                color='rgba(50, 171, 96, 1.0)',
                width=2,
            )
        )
    )
    data = [trace0, trace1, trace2, trace3]
    layout = go.Layout(
        title='Annual Profit- 2015',
        barmode='stack',
        paper_bgcolor='rgba(245, 246, 249, 1)',
        plot_bgcolor='rgba(245, 246, 249, 1)',
        showlegend=False
    )

    annotations = []

    for i in range(0, 7):
        annotations.append(dict(x=x_data[i], y=y_data[i], text=text[i],
                                font=dict(family='Arial', size=14,
                                color='rgba(245, 246, 249, 1)'),
                                showarrow=False,))
        layout['annotations'] = annotations

    fig = go.Figure(data=data, layout=layout)
    py.iplot(fig, filename='waterfall-bar-profit')
```

Out[2]:



有依存关系的柱状图

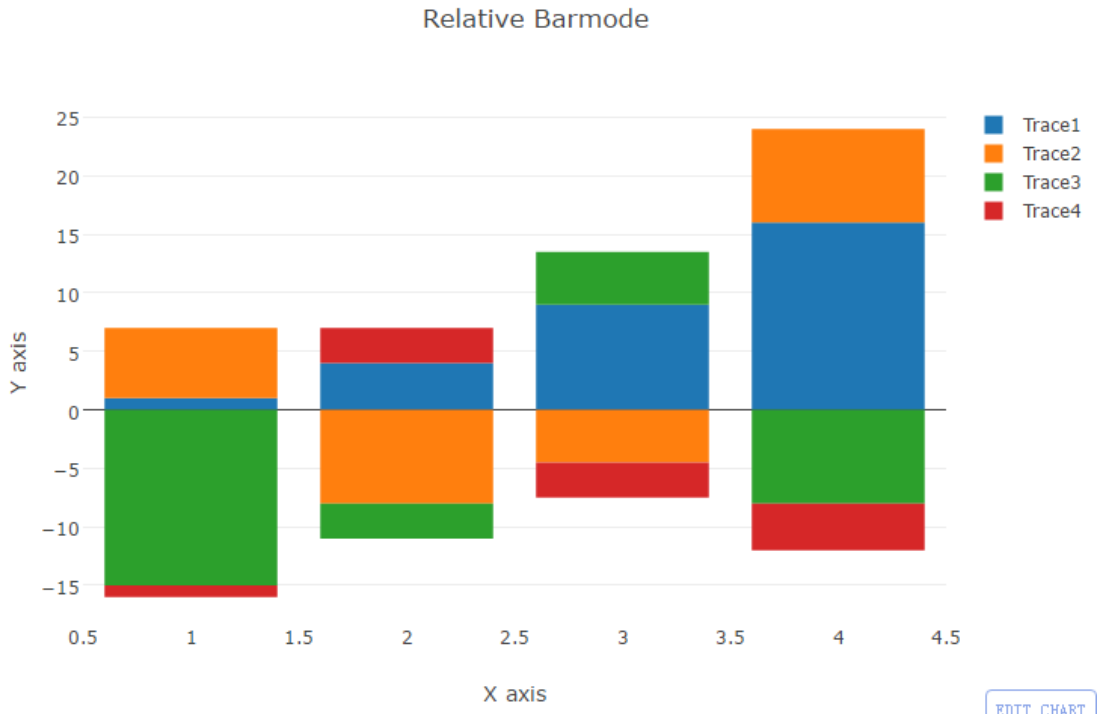
Bar Chart with Relative Barmode

In [2]:

```
import plotly.plotly as py
import plotly.graph_objs as go

x = [1, 2, 3, 4]

trace1 = {
  'x': x,
  'y': [1, 4, 9, 16],
  'name': 'Trace1',
  'type': 'bar'
};
trace2 = {
  'x': x,
  'y': [6, -8, -4.5, 8],
  'name': 'Trace2',
  'type': 'bar'
};
trace3 = {
```

```
  'x': x,
  'y': [-15, -3, 4.5, -8],
  'name': 'Trace3',
  'type': 'bar'
 }

trace4 = {
  'x': x,
  'y': [-1, 3, -3, -4],
  'name': 'Trace4',
  'type': 'bar'
 }

data = [trace1, trace2, trace3, trace4];
layout = {
  'xaxis': {'title': 'X axis'},
  'yaxis': {'title': 'Y axis'},
  'barmode': 'relative',
  'title': 'Relative Barmode'
};
py.iplot({'data': data, 'layout': layout}, filename='barmode-relative')
```

Out[2]:



横向柱状图

Horizontal Bar Charts

See examples of horizontal bar charts here

查看上述资料获取相关信息这里

水平条形图

Horizontal Bar Charts in Python

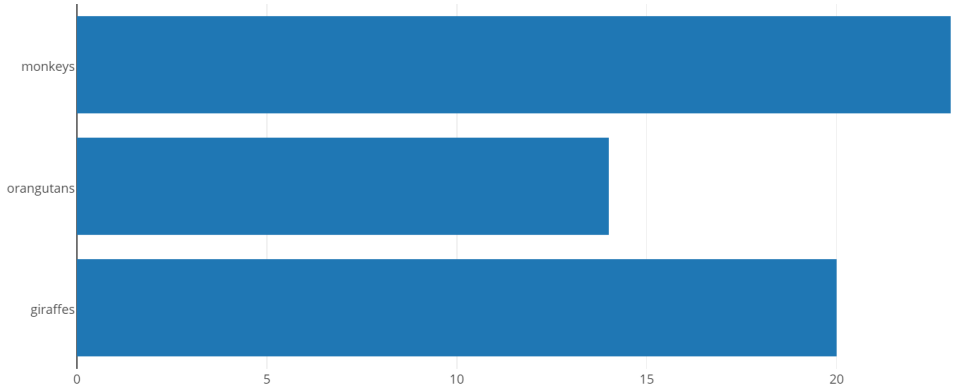
常用水平条形图

Basic Horizontal Bar Chart

In[1]:

```
import plotly.plotly as py
import plotly.graph_objs as go

data = [go.Bar(
            x=[20, 14, 23],
            y=['giraffes', 'orangutans', 'monkeys'],
            orientation = 'h'
)]

py.iplot(data, filename='horizontal-bar')
```

Out[1]



彩色水平条形图

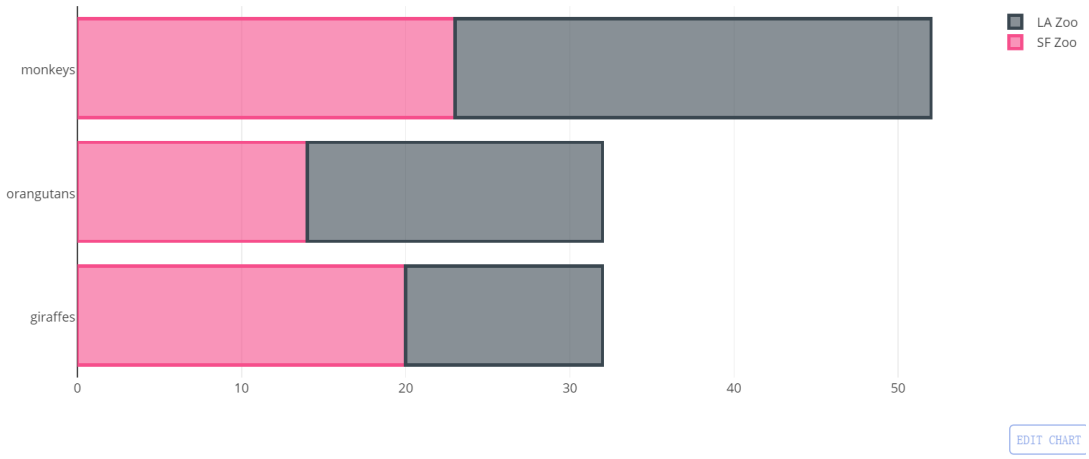
Colored Horizontal Bar Chart

In[2]:

```
iimport plotly.plotly as py
import plotly.graph_objs as go

trace1 = go.Bar(
    y=['giraffes', 'orangutans', 'monkeys'],
    x=[20, 14, 23],
    name='SF Zoo',
    orientation = 'h',
    marker = dict(
        color = 'rgba(246, 78, 139, 0.6)',
        line = dict(
            color = 'rgba(246, 78, 139, 1.0)',
            width = 3)
    )
)
trace2 = go.Bar(
    y=['giraffes', 'orangutans', 'monkeys'],
    x=[12, 18, 29],
```

```
    name='LA Zoo',
    orientation = 'h',
    marker = dict(
        color = 'rgba(58, 71, 80, 0.6)',
        line = dict(
            color = 'rgba(58, 71, 80, 1.0)',
            width = 3)
    )
)

data = [trace1, trace2]
layout = go.Layout(
    barmode='stack'
)

fig = go.Figure(data=data, layout=layout)
py.iplot(fig, filename='marker-h-bar')
```

Out[2]:



柱状图调色板

Color Palette for Bar Chart

In[3]:

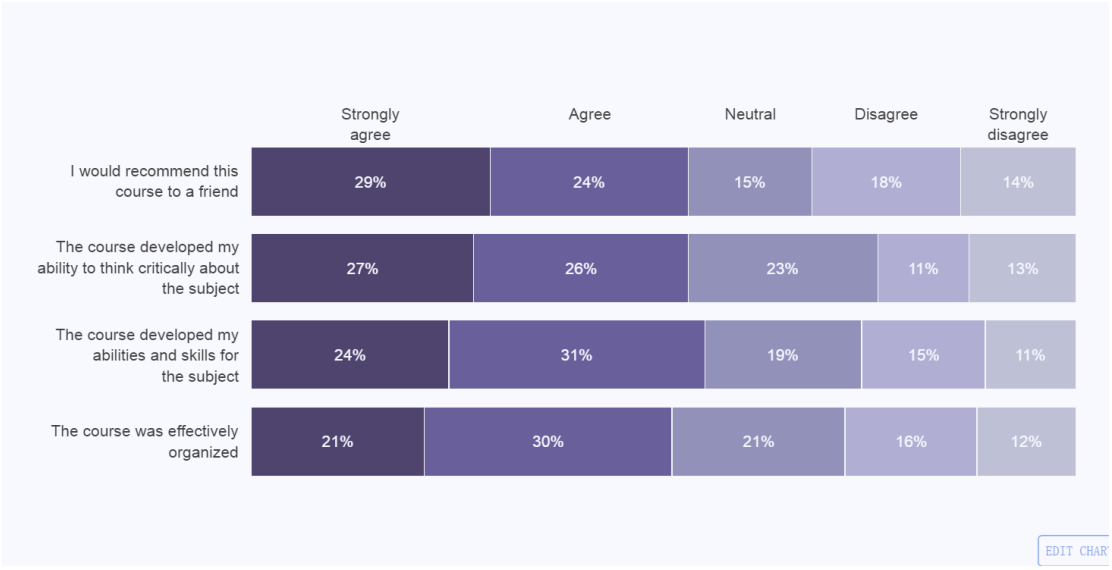
```
import plotly.plotly as py
import plotly.graph_objs as go
```

```
top_labels = ['Strongly<br>agree', 'Agree', 'Neutral', 'Disagree',
              'Strongly<br>disagree']

colors = ['rgba(38, 24, 74, 0.8)', 'rgba(71, 58, 131, 0.8)',
          'rgba(122, 120, 168, 0.8)', 'rgba(164, 163, 204, 0.85)',
          'rgba(190, 192, 213, 1)']

x_data = [[21, 30, 21, 16, 12],
          [24, 31, 19, 15, 11],
          [27, 26, 23, 11, 13],
          [29, 24, 15, 18, 14]]

y_data = ['The course was effectively<br>organized',
          'The course developed my<br>abilities and skills ' +
          'for<br>the subject', 'The course developed ' +
          'my<br>ability to think critically about<br>the subject',
          'I would recommend this<br>course to a friend']


traces = []

for i in range(0, len(x_data[0])):
    for xd, yd in zip(x_data, y_data):
        traces.append(go.Bar(
            x=xd[i],
            y=yd,
            orientation='h',
            marker=dict(
                color=colors[i],
                line=dict(
                        color='rgb(248, 248, 249)',
                        width=1)
            )
        ))

layout = go.Layout(
    xaxis=dict(
        showgrid=False,
        showline=False,
```

```
        showticklabels=False,
        zeroline=False,
        domain=[0.15, 1]
    ),
    yaxis=dict(
        showgrid=False,
        showline=False,
        showticklabels=False,
        zeroline=False,
    ),
    barmode='stack',
    paper_bgcolor='rgb(248, 248, 255)',
    plot_bgcolor='rgb(248, 248, 255)',
    margin=dict(
        l=120,
        r=10,
        t=140,
        b=80
    ),
    showlegend=False,
)

annotations = []

for yd, xd in zip(y_data, x_data):
    # labeling the y-axis
    annotations.append(dict(xref='paper', yref='y',
                            x=0.14, y=yd,
                            xanchor='right',
                            text=str(yd),
                            font=dict(family='Arial', size=14,
                                      color='rgb(67, 67, 67)'),
                            showarrow=False, align='right'))
    # labeling the first percentage of each bar (x_axis)
    annotations.append(dict(xref='x', yref='y',
                            x=xd[0] / 2, y=yd,
                            text=str(xd[0]) + '%',
                            font=dict(family='Arial', size=14,
                                      color='rgb(248, 248, 255)'),
                            showarrow=False))
```

```
    # labeling the first Likert scale (on the top)
    if yd == y_data[-1]:
        annotations.append(dict(xref='x', yref='paper',
                                x=xd[0] / 2, y=1.1,
                                text=top_labels[0],
                                font=dict(family='Arial', size=14,
                                          color='rgb(67, 67, 67)'),
                                showarrow=False))
    space = xd[0]
    for i in range(1, len(xd)):
            # labeling the rest of percentages for each bar (x_axis)
            annotations.append(dict(xref='x', yref='y',
                                    x=space + (xd[i]/2), y=yd,
                                    text=str(xd[i]) + '%',
                                    font=dict(family='Arial', size=14,
                                              color='rgb(248, 248, 255)'),
                                    showarrow=False))
            # labeling the Likert scale
            if yd == y_data[-1]:
                annotations.append(dict(xref='x', yref='paper',
                                        x=space + (xd[i]/2), y=1.1,
                                        text=top_labels[i],
                                        font=dict(family='Arial', size=14,
                                                  color='rgb(67, 67, 67)'),
                                        showarrow=False))
            space += xd[i]

layout['annotations'] = annotations

fig = go.Figure(data=traces, layout=layout)
py.iplot(fig, filename='bar-colorscale')
Copy to Clipboard!!
```

Out[3]:



附有折线的柱状图

Bar Chart with Line Plot

In[4]:

```
import plotly.plotly as py
import plotly.graph_objs as go
from plotly import tools

import numpy as np

y_saving = [1.3586, 2.2623000000000002, 4.9821999999999997, 6.5096999999999996,
          7.4812000000000003, 7.5133000000000001, 15.2148, 17.520499999999998
          ]
y_net_worth = [93453.919999999998, 81666.570000000007, 69889.619999999995,
            78381.529999999999, 141395.29999999999, 92969.020000000004,
            66090.179999999993, 122379.3]
x_saving = ['Japan', 'United Kingdom', 'Canada', 'Netherlands',
          'United States', 'Belgium', 'Sweden', 'Switzerland']
x_net_worth = ['Japan', 'United Kingdom', 'Canada', 'Netherlands',
            'United States', 'Belgium', 'Sweden', 'Switzerland'
            ]
trace0 = go.Bar(
    x=y_saving,
    y=x_saving,
    marker=dict(
```

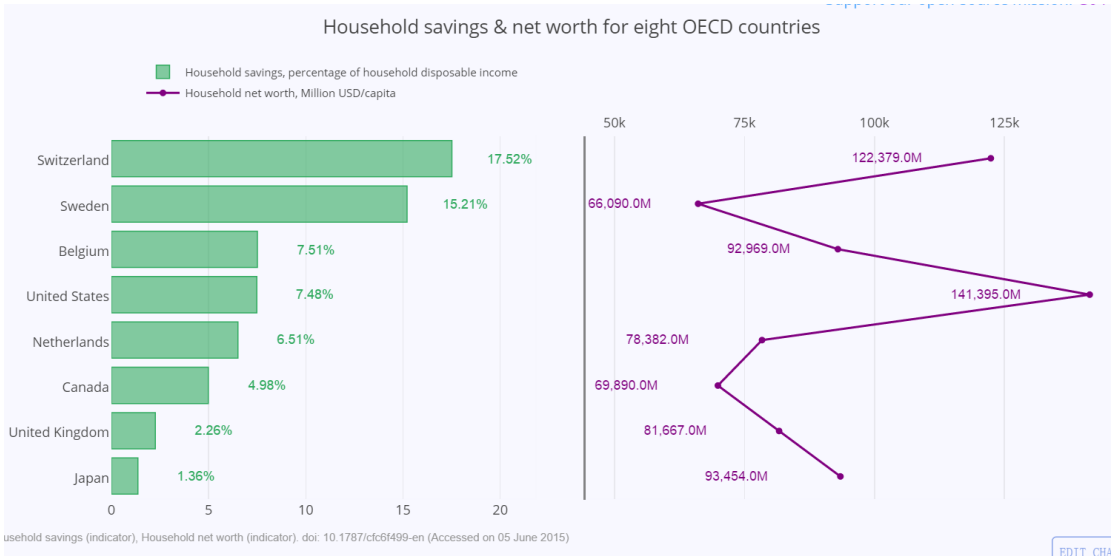
```
        color='rgba(50, 171, 96, 0.6)',
        line=dict(
            color='rgba(50, 171, 96, 1.0)',
            width=1),
    ),
    name='Household savings, percentage of household disposable income',
    orientation='h',
)
trace1 = go.Scatter(
    x=y_net_worth,
    y=x_net_worth,
    mode='lines+markers',
    line=dict(
        color='rgb(128, 0, 128)'),
    name='Household net worth, Million USD/capita',
)
layout = dict(
    title='Household savings & net worth for eight OECD countries',
    yaxis1=dict(
        showgrid=False,
        showline=False,
        showticklabels=True,
        domain=[0, 0.85],
    ),
    yaxis2=dict(
        showgrid=False,
        showline=True,
        showticklabels=False,
        linecolor='rgba(102, 102, 102, 0.8)',
        linewidth=2,
        domain=[0, 0.85],
    ),
    xaxis1=dict(
        zeroline=False,
        showline=False,
        showticklabels=True,
        showgrid=True,
        domain=[0, 0.42],
    ),
    xaxis2=dict(
```

```
        zeroline=False,
        showline=False,
        showticklabels=True,
        showgrid=True,
        domain=[0.47, 1],
        side='top',
        dtick=25000,
    ),
    legend=dict(
        x=0.029,
        y=1.038,
        font=dict(
            size=10,
        ),
    ),
    margin=dict(
        l=100,
        r=20,
        t=70,
        b=70,
    ),
    paper_bgcolor='rgb(248, 248, 255)',
    plot_bgcolor='rgb(248, 248, 255)',
)

annotations = []

y_s = np.round(y_saving, decimals=2)
y_nw = np.rint(y_net_worth)

# Adding labels
for ydn, yd, xd in zip(y_nw, y_s, x_saving):
    # labeling the scatter savings
    annotations.append(dict(xref='x2', yref='y2',
                            y=xd, x=ydn - 20000,
                            text='{:,}'.format(ydn) + 'M',
                            font=dict(family='Arial', size=12,
                                      color='rgb(128, 0, 128)'),
                            showarrow=False))
    # labeling the bar net worth
```

```
    annotations.append(dict(xref='x1', yref='y1',
                            y=xd, x=yd + 3,
                            text=str(yd) + '%',
                            font=dict(family='Arial', size=12,
                                      color='rgb(50, 171, 96)'),
                            showarrow=False))
# Source
annotations.append(dict(xref='paper', yref='paper',
                        x=-0.2, y=-0.109,
                        text='OECD "' +
                             '(2015), Household savings (indicator), ' +
                             'Household net worth (indicator). doi: ' +
                             '10.1787/cfc6f499-en (Accessed on 05 June 2015)',
                        font=dict(family='Arial', size=10,
                                  color='rgb(150,150,150)'),
                        showarrow=False))

layout['annotations'] = annotations

# Creating two subplots
fig = tools.make_subplots(rows=1, cols=2, specs=[[{}, {}]], shared_xaxes=True,
                          shared_yaxes=False, vertical_spacing=0.001)

fig.append_trace(trace0, 1, 1)
fig.append_trace(trace1, 1, 2)

fig['layout'].update(layout)
py.iplot(fig, filename='oecd-networth-saving-bar-line')
```

```
This is the format of your plot grid:
[ (1,1) x1,y1 ]  [ (1,2) x2,y2 ]

Copy to Clipboard!!
```

Out[4]:



参考

Reference

See more examples of bar charts and styling options https://plot.ly/python/bar-charts/

更多柱状图和式样选项的例子请查看 https://plot.ly/python/bar-charts/

See https://plot.ly/python/reference/#bar for more information and chart attribute options!

查看 https://plot.ly/python/reference/#bar 来获取更多的信息和图表属性选项。

甘特图(横道图)

Gantt Charts in Python

版本检查

Version Check

Note: Gantt Charts are available in version 1.12.2+

提示:甘特图只能在 1.12.2 以上版本使用

Run pip install plotly --upgrade to update your Plotly version

运行 pip install plotly --upgrade 来更新你的 plotly 版本

In[1]:

```
import plotly
```

```
plotly.__version__
```

Out[1]:

```
'1.12.2'
```

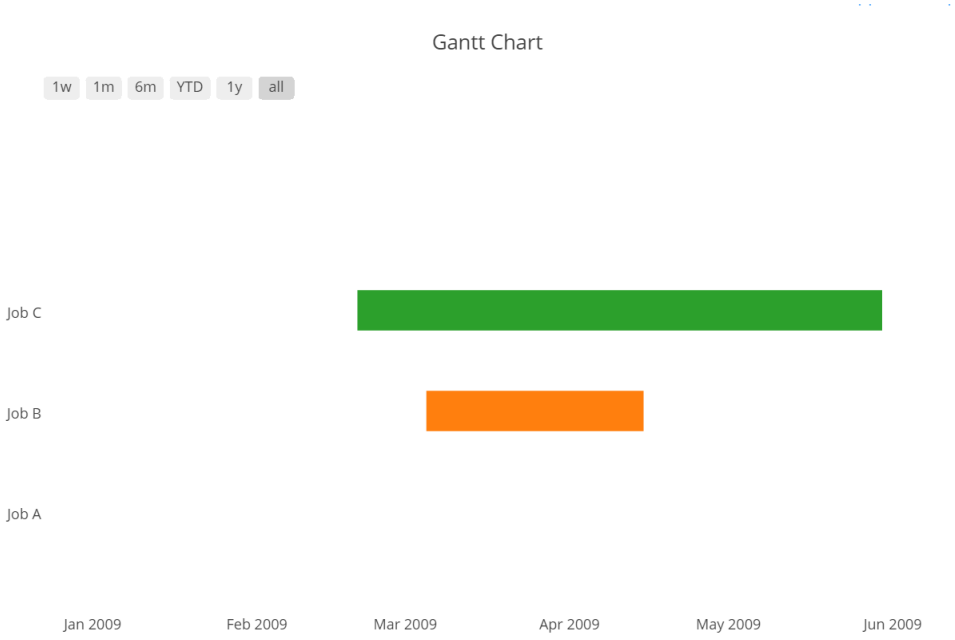
简单甘特图

Simple Gantt Chart

In[2]:

```
import plotly.plotly as py
from plotly.tools import FigureFactory as FF

df = [dict(Task="Job A", Start='2009-01-01', Finish='2009-02-30'),
      dict(Task="Job B", Start='2009-03-05', Finish='2009-04-15'),
      dict(Task="Job C", Start='2009-02-20', Finish='2009-05-30')]

fig = FF.create_gantt(df)
py.iplot(fig, filename='Simple Gantt Chart', world_readable=True)
```

Out[2]:



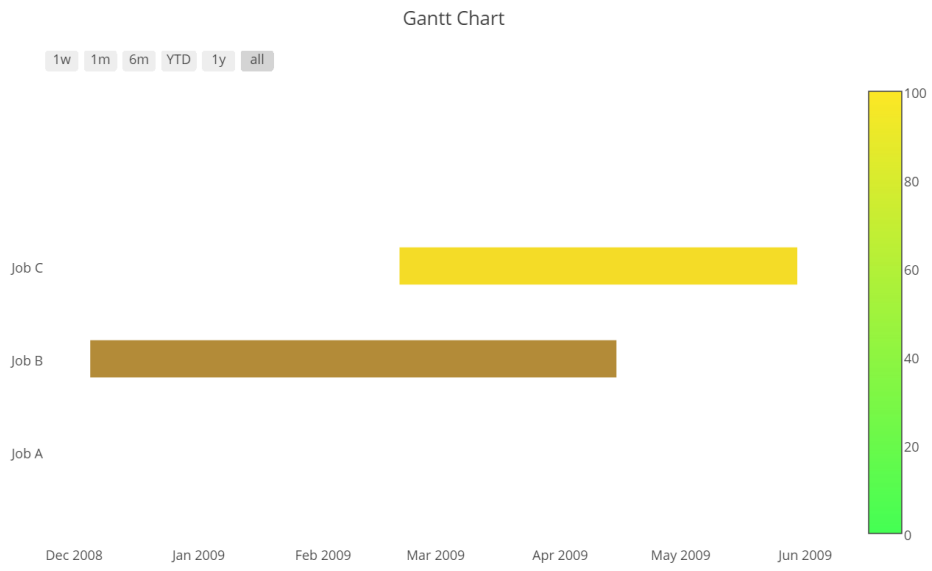
使用数值型变量做索引

Index by Numeric Variable

In[3]:

```
import plotly.plotly as py
from plotly.tools import FigureFactory as FF

df = [dict(Task="Job A", Start='2009-01-01', Finish='2009-02-30', Complete=10),
      dict(Task="Job B", Start='2008-12-05', Finish='2009-04-15', Complete=60),
      dict(Task="Job C", Start='2009-02-20', Finish='2009-05-30', Complete=95)]

fig = FF.create_gantt(df, colors='Viridis', index_col='Complete', show_colorbar=True)
py.iplot(fig, filename='Numeric Variable', world_readable=True)
```

Out[3]:



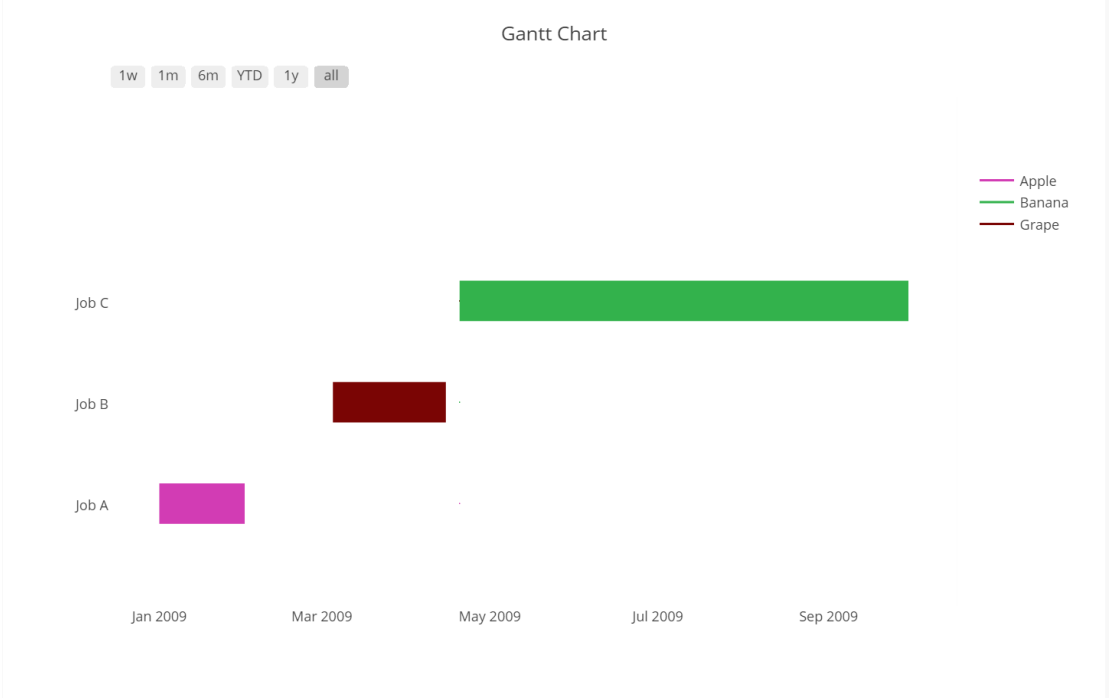
使用字符型变量做索引

Index by String Variable

In[4]:

```
import plotly.plotly as py
from plotly.tools import FigureFactory as FF

df = [dict(Task="Job A", Start='2009-01-01', Finish='2009-02-01', Resource='Apple'),
      dict(Task="Job B", Start='2009-03-05', Finish='2009-04-15', Resource='Grape'),
      dict(Task="Job C", Start='2009-04-20', Finish='2009-09-30', Resource='Banana')]

colors = ['#7a0504', (0.2, 0.7, 0.3), 'rgb(210, 60, 180)']

fig = FF.create_gantt(df, colors=colors, index_col='Resource', reverse_colors=True,
show_colorbar=True)
py.iplot(fig, filename='String Variable', world_readable=True)
```

Out[4]:



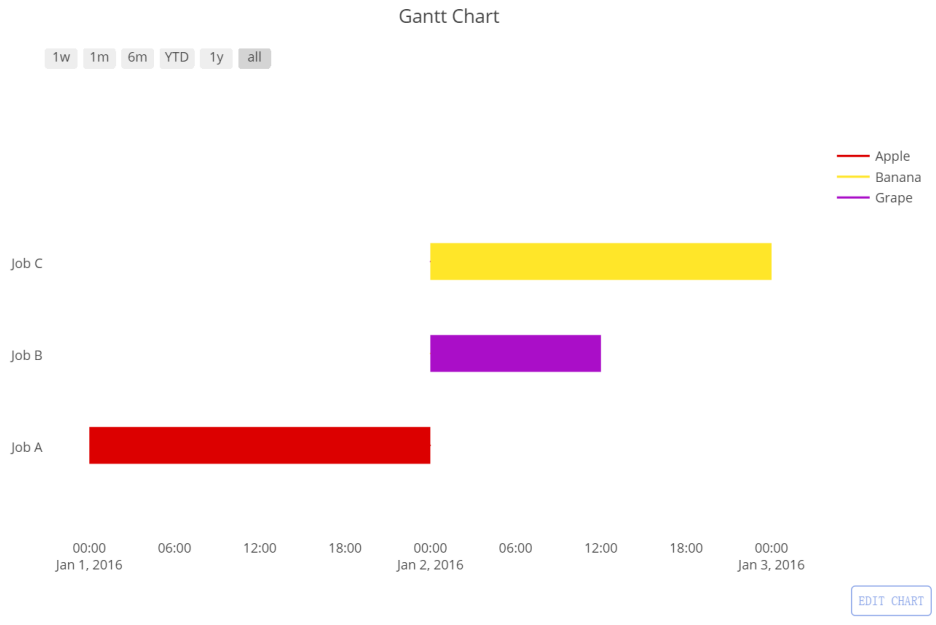
使用颜色字典

Use a Dictionary for Colors

In[5]:

```
import plotly.plotly as py
from plotly.tools import FigureFactory as FF

df = [dict(Task="Job A", Start='2016-01-01', Finish='2016-01-02', Resource='Apple'),
    dict(Task="Job B", Start='2016-01-02', Finish='2016-01-02 12:00:00',
Resource='Grape'),
    dict(Task="Job C", Start='2016-01-02', Finish='2016-01-03', Resource='Banana')]

colors = dict(Apple = 'rgb(220, 0, 0)',
            Grape = 'rgb(170, 14, 200)',
            Banana = (1, 0.9, 0.16))

fig = FF.create_gantt(df, colors=colors, index_col='Resource', show_colorbar=True)
py.iplot(fig, filename='Dictioanry Colors', world_readable=True)
```

Out[5]:



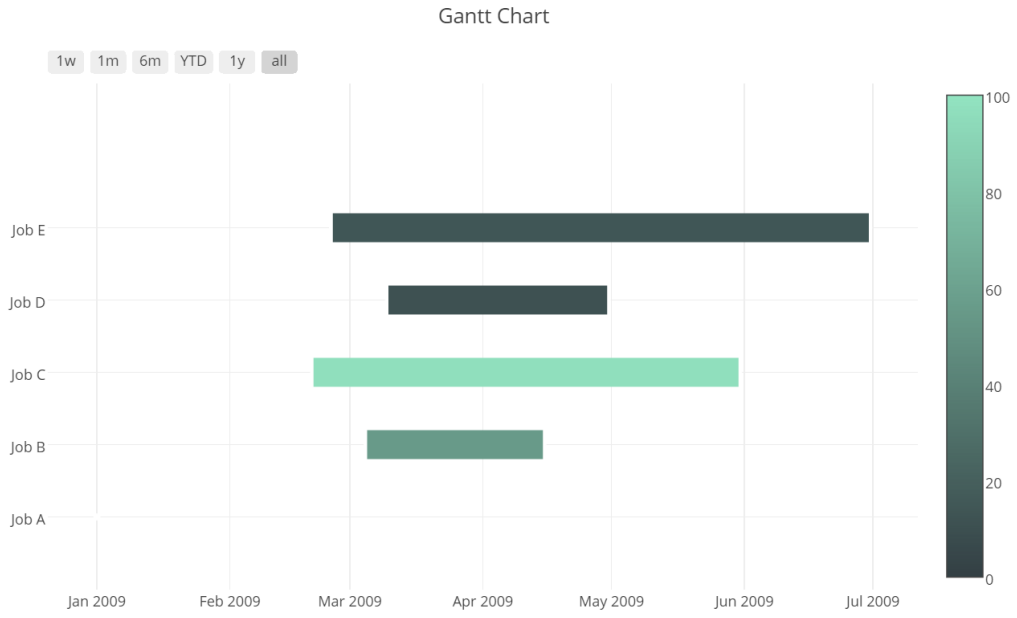
使用 pandas 数据框

Use a Pandas Dataframe

In[6]:

```
import plotly.plotly as py
from plotly.tools import FigureFactory as FF

import pandas as pd

df =
pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/gantt_example.csv')

fig = FF.create_gantt(df, colors=['#333F44', '#93e4c1'], index_col='Complete',
show_colorbar=True,
                      bar_width=0.2, showgrid_x=True, showgrid_y=True)
py.iplot(fig, filename='Use a Pandas Dataframe', world_readable=True)
```

Out[6]:



参考

Reference

For info on Plotly Range Slider and Selector, see: https://plot.ly/python/reference/#layout-xaxis-rangeselector.

关于 Plotly 的范围滑块和选择器,请见 https://plot.ly/python/reference/#layout-xaxis-rangeselector.

In[7]:

```
help(FF.create_gantt)
```

```
Help on function create_gantt in module plotly.tools:

create_gantt(df, colors=None, index_col=None, show_colorbar=False,
reverse_colors=False, title='Gantt Chart', bar_width=0.2, showgrid_x=False,
showgrid_y=False, height=600, width=900, tasks=None, task_names=None,
data=None)
    Returns figure for a gantt chart

    :param (array|list) df: input data for gantt chart. Must be either a
        a dataframe or a list. If dataframe, the columns must include
        'Task', 'Start' and 'Finish'. Other columns can be included and
        used for indexing. If a list, its elements must be dictionaries
        with the same required column headers: 'Task', 'Start' and
        'Finish'.
    :param (str|list|dict|tuple) colors: either a plotly scale name, an
        rgb or hex color, a color tuple or a list of colors. An rgb color
        is of the form 'rgb(x, y, z)' where x, y, z belong to the interval
        [0, 255] and a color tuple is a tuple of the form (a, b, c) where
        a, b and c belong to [0, 1]. If colors is a list, it must
        contain the valid color types aforementioned as its members.
        If a dictionary, all values of the indexing column must be keys in
        colors.
    :param (str|float) index_col: the column header (if df is a data
        frame) that will function as the indexing column. If df is a list,
        index_col must be one of the keys in all the items of df.
    :param (bool) show_colorbar: determines if colorbar will be visible.
        Only applies if values in the index column are numeric.
    :param (bool) reverse_colors: reverses the order of selected colors
    :param (str) title: the title of the chart
    :param (float) bar_width: the width of the horizontal bars in the plot
    :param (bool) showgrid_x: show/hide the x-axis grid
    :param (bool) showgrid_y: show/hide the y-axis grid
    :param (float) height: the height of the chart
    :param (float) width: the width of the chart
```

Example 1: Simple Gantt Chart
```
import plotly.plotly as py
from plotly.tools import FigureFactory as FF

# Make data for chart
df = [dict(Task="Job A", Start='2009-01-01', Finish='2009-02-30'),
      dict(Task="Job B", Start='2009-03-05', Finish='2009-04-15'),
      dict(Task="Job C", Start='2009-02-20', Finish='2009-05-30')]

# Create a figure
fig = FF.create_gantt(df)

# Plot the data
py.iplot(fig, filename='Simple Gantt Chart', world_readable=True)
```

Example 2: Index by Column with Numerical Entries
```
import plotly.plotly as py
from plotly.tools import FigureFactory as FF

# Make data for chart
df = [dict(Task="Job A", Start='2009-01-01',
           Finish='2009-02-30', Complete=10),
      dict(Task="Job B", Start='2009-03-05',
           Finish='2009-04-15', Complete=60),
      dict(Task="Job C", Start='2009-02-20',
           Finish='2009-05-30', Complete=95)]

# Create a figure with Plotly colorscale
fig = FF.create_gantt(df, colors='Blues', index_col='Complete',
                      show_colorbar=True, bar_width=0.5,
                      showgrid_x=True, showgrid_y=True)

# Plot the data
py.iplot(fig, filename='Numerical Entries', world_readable=True)
```

Example 3: Index by Column with String Entries

```
import plotly.plotly as py
from plotly.tools import FigureFactory as FF

# Make data for chart
df = [dict(Task="Job A", Start='2009-01-01',
           Finish='2009-02-30', Resource='Apple'),
      dict(Task="Job B", Start='2009-03-05',
           Finish='2009-04-15', Resource='Grape'),
      dict(Task="Job C", Start='2009-02-20',
           Finish='2009-05-30', Resource='Banana')]

# Create a figure with Plotly colorscale
fig = FF.create_gantt(df, colors=['rgb(200, 50, 25)',
                                  (1, 0, 1),
                                  '#6c4774'],
                      index_col='Resource',
                      reverse_colors=True,
                      show_colorbar=True)

# Plot the data
py.iplot(fig, filename='String Entries', world_readable=True)
```

Example 4: Use a dictionary for colors

```
import plotly.plotly as py
from plotly.tools import FigureFactory as FF

# Make data for chart
df = [dict(Task="Job A", Start='2009-01-01',
           Finish='2009-02-30', Resource='Apple'),
      dict(Task="Job B", Start='2009-03-05',
           Finish='2009-04-15', Resource='Grape'),
      dict(Task="Job C", Start='2009-02-20',
           Finish='2009-05-30', Resource='Banana')]

# Make a dictionary of colors
colors = {'Apple': 'rgb(255, 0, 0)',
          'Grape': 'rgb(170, 14, 200)',
```

```
        'Banana': (1, 1, 0.2)}

# Create a figure with Plotly colorscale
fig = FF.create_gantt(df, colors=colors,
                      index_col='Resource',
                      show_colorbar=True)

# Plot the data
py.iplot(fig, filename='dictioanry colors', world_readable=True)
```

Example 5: Use a pandas dataframe

```
import plotly.plotly as py
from plotly.tools import FigureFactory as FF

import pandas as pd

# Make data as a dataframe
df = pd.DataFrame([['Run', '2010-01-01', '2011-02-02', 10],
                   ['Fast', '2011-01-01', '2012-06-05', 55],
                   ['Eat', '2012-01-05', '2013-07-05', 94]],
                  columns=['Task', 'Start', 'Finish', 'Complete'])

# Create a figure with Plotly colorscale
fig = FF.create_gantt(df, colors='Blues', index_col='Complete',
                      show_colorbar=True, bar_width=0.5,
                      showgrid_x=True, showgrid_y=True)

# Plot the data
py.iplot(fig, filename='data with dataframe', world_readable=True)
```

堆积图

Filled Area Plots

Filled Area Plots In Python

How to make filled area plots in Python with Plotly.

Python 语言下，如果用 Plotly 实现填充面积图的绘制。

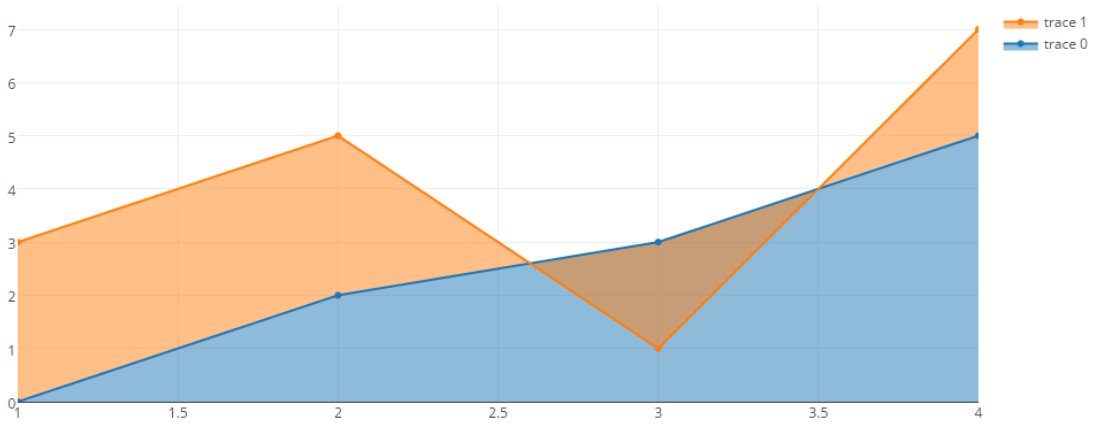
基本堆积图

Basic Overlaid Area Chart

In [1]:

```
import plotly.plotly as py
import plotly.graph_objs as go

trace1 = go.Scatter(
    x=[1, 2, 3, 4],
    y=[0, 2, 3, 5],
    fill='tozeroy'
)
trace2 = go.Scatter(
    x=[1, 2, 3, 4],
    y=[3, 5, 1, 7],
    fill='tonexty'
)

data = [trace1, trace2]
py.iplot(data, filename='basic-area')
```

Out [1]:



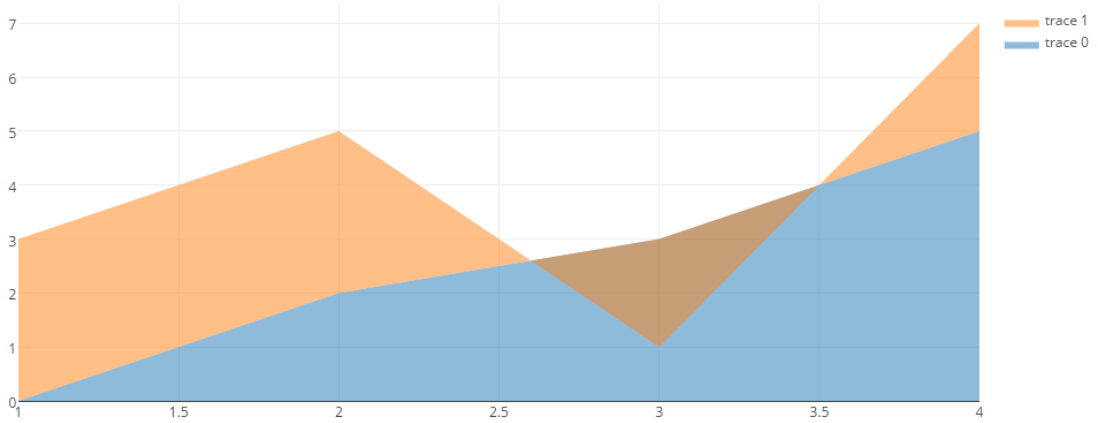
无轮廓线遮盖面积图

Overlaid Area Chart Without Boundary Lines

In [2]:

```
import plotly.plotly as py
import plotly.graph_objs as go

trace1 = go.Scatter(
    x=[1, 2, 3, 4],
    y=[0, 2, 3, 5],
    fill='tozeroy',
    mode= 'none'
)
trace2 = go.Scatter(
    x=[1, 2, 3, 4],
    y=[3, 5, 1, 7],
    fill='tonexty',
    mode= 'none'
)

data = [trace1, trace2]
py.iplot(data, filename='basic-area-no-bound')
```

Out [2]:



内部填充式面积图

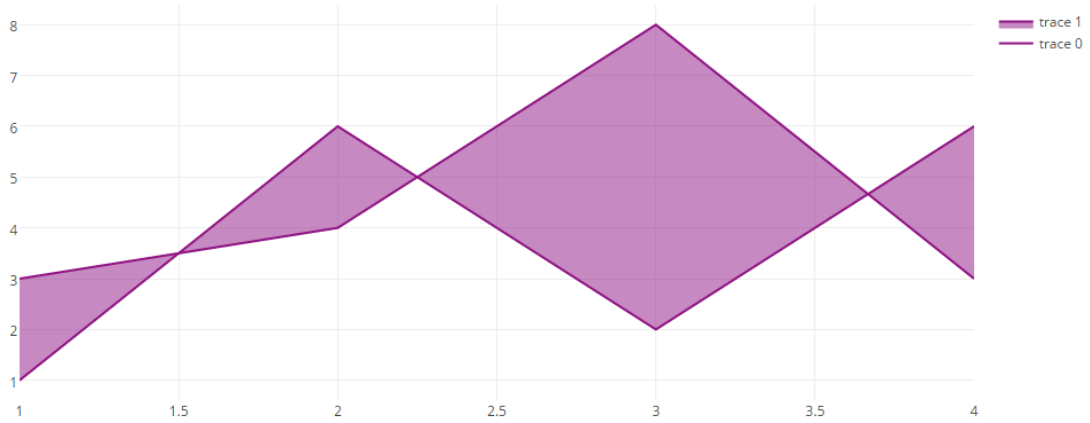
Interior Filling for Area Chart

In [3]:

```
import plotly.plotly as py
import plotly.graph_objs as go

trace0 = go.Scatter(
    x=[1, 2, 3, 4],
    y=[3, 4, 8, 3],
    fill= None,
    mode='lines',
    line=dict(
        color='rgb(143, 19, 131)',
    )
)
trace1 = go.Scatter(
    x=[1, 2, 3, 4],
    y=[1, 6, 2, 6],
    fill='tonexty',
    mode='lines',
    line=dict(
        color='rgb(143, 19, 131)',
    )
)
```

```
data = [trace0, trace1]
py.iplot(data, filename='filling-interior-area')
```

Out [3]:



累积值堆积面积图

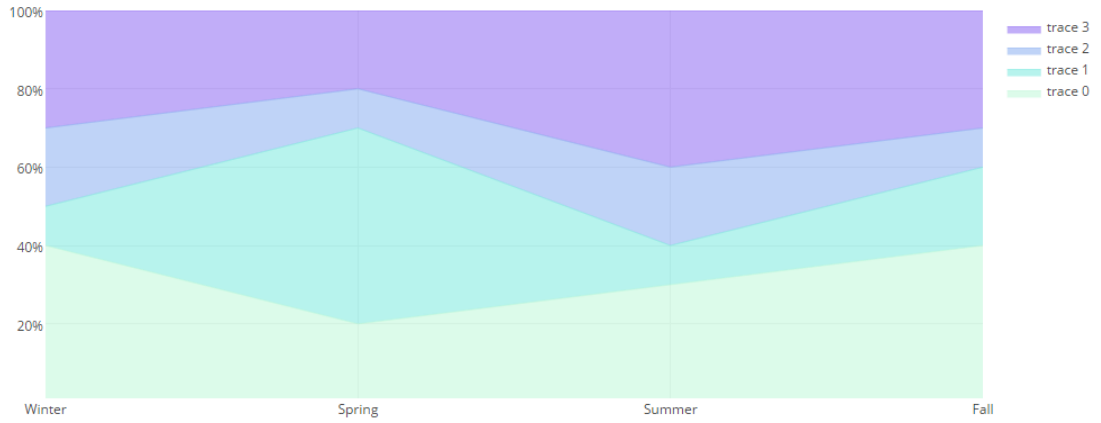
Stacked Area Chart with Cumulative Values

In [4]:

```
import plotly.plotly as py
import plotly.graph_objs as go

trace0 = go.Scatter(
    x=['Winter', 'Spring', 'Summer', 'Fall'],
    y=['40', '20', '30', '40'],
    mode='lines',
    line=dict(width=0.5,
              color='rgb(184, 247, 212)'),
    fill='tonexty'
)
trace1 = go.Scatter(
    x=['Winter', 'Spring', 'Summer', 'Fall'],
    y=['50', '70', '40', '60'],
    mode='lines',
    line=dict(width=0.5,
```

```
            color='rgb(111, 231, 219)'),
        fill='tonexty'
    )
    trace2 = go.Scatter(
        x=['Winter', 'Spring', 'Summer', 'Fall'],
        y=['70', '80', '60', '70'],
        mode='lines',
        line=dict(width=0.5,
            color='rgb(127, 166, 238)'),
        fill='tonexty'
    )
    trace3 = go.Scatter(
        x=['Winter', 'Spring', 'Summer', 'Fall'],
        y=['100', '100', '100', '100'],
        mode='lines',
        line=dict(width=0.5,
            color='rgb(131, 90, 241)'),
        fill='tonexty'
    )
    data = [trace0, trace1, trace2, trace3]
    layout = go.Layout(
        showlegend=True,
        xaxis=dict(
            type='category',
        ),
        yaxis=dict(
            type='linear',
            range=[1, 100],
            dtick=20,
            ticksuffix='%'
        )
    )
    fig = go.Figure(data=data, layout=layout)
    py.iplot(fig, filename='stacked-area-plot')
```

Out [4]:



原始值堆积面积图

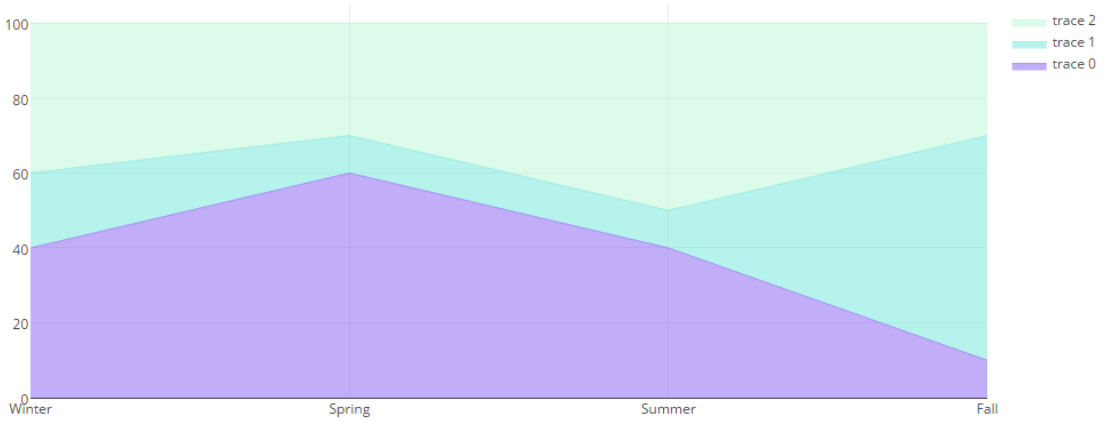
Stacked Area Chart with Original Values

In [5]:

```
import plotly.plotly as py
import plotly.graph_objs as go

# Add original data
x=['Winter', 'Spring', 'Summer', 'Fall']

y0_org=[40, 60, 40, 10]
y1_org=[20, 10, 10, 60]
y2_org=[40, 30, 50, 30]

# Add data to create cumulative stacked values
y0_stck=y0_org
y1_stck=[y0+y1 for y0, y1 in zip(y0_org, y1_org)]
y2_stck=[y0+y1+y2 for y0, y1, y2 in zip(y0_org, y1_org, y2_org)]

# Make original values strings and add % for hover text
y0_txt=[str(y0)+'%' for y0 in y0_org]
y1_txt=[str(y1)+'%' for y1 in y1_org]
y2_txt=[str(y2)+'%' for y2 in y2_org]

trace0 = go.Scatter(
    x=x,
```

```
    y=y0_stck,
    text=y0_txt,
    hoverinfo='x+text',
    mode='lines',
    line=dict(width=0.5,
              color='rgb(131, 90, 241)'),
    fill='tonexty'
)
trace1 = go.Scatter(
    x=x,
    y=y1_stck,
    text=y1_txt,
    hoverinfo='x+text',
    mode='lines',
    line=dict(width=0.5,
              color='rgb(111, 231, 219)'),
    fill='tonexty'
)
trace2 = go.Scatter(
    x=x,
    y=y2_stck,
    text=y2_txt,
    hoverinfo='x+text',
    mode='lines',
    line=dict(width=0.5,
              color='rgb(184, 247, 212)'),
    fill='tonexty'
)
data = [trace0, trace1, trace2]

fig = go.Figure(data=data)
py.iplot(fig, filename='stacked-area-plot-hover')
```

Out [5]:



参考

Reference

See https://plot.ly/python/reference/#scatter-line and https://plot.ly/python/reference/#scatter-fill for more information and attribute options!

访问 https://plot.ly/python/reference/#scatter-line 和 https://plot.ly/python/reference/#scatter-fill 可获得更多资讯信息和属性选项

饼图扇形图

Pie Charts

Python 中的饼图

Pie Charts in Python

How to make Pie Charts.

如何制作饼图

Version Check

Note: Pie Charts are available in version 1.9.12+

Run pip install plotly --upgrade to update your Plotly version

版本检查：

说明： 1.9.12 版本后新增饼图功能

执行 **pip install plotly --upgrade** 可升级您的 Plotly 版本

In [1]:

```
import plotly
plotly.__version__
```

Out [1]:

```
'1.11.0'
```

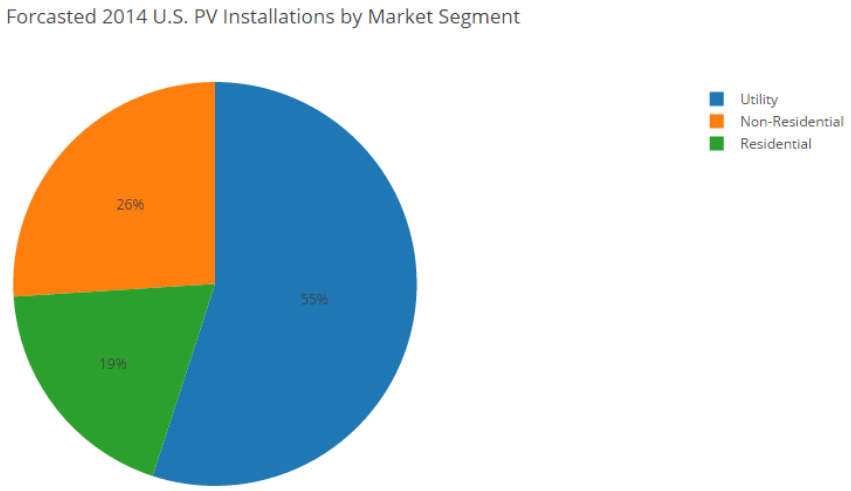
基本饼图

Basic Pie Chart

In [2]:

```
import plotly.plotly as py
import plotly.graph_objs as go

fig = {
    'data': [{'labels': ['Residential', 'Non-Residential', 'Utility'],
              'values': [19, 26, 55],
              'type': 'pie'}],
    'layout': {'title': 'Forcasted 2014 U.S. PV Installations by Market Segment'}
     }

py.iplot(fig)
```

Out [2]:



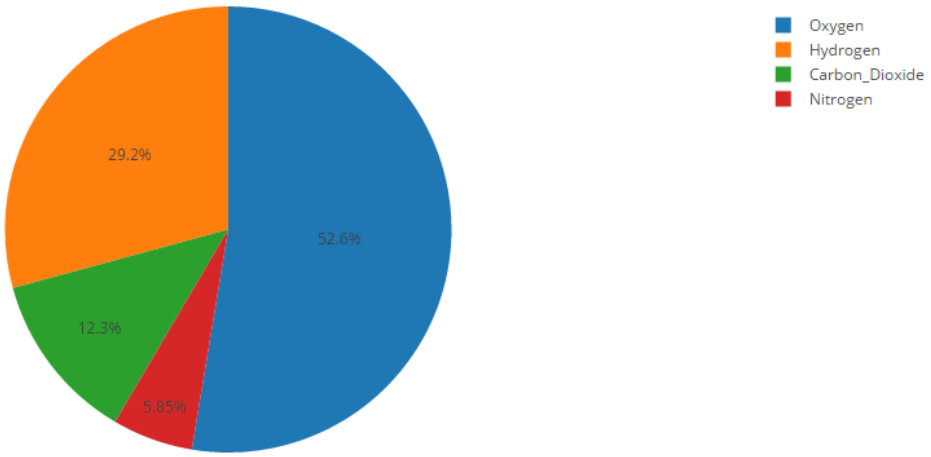
使用饼对戏的饼图

Pie Chart using Pie object

In [3]:

```
import plotly.plotly as py
import plotly.graph_objs as go

labels=['Oxygen','Hydrogen','Carbon_Dioxide','Nitrogen']
values=[4500,2500,1053,500]

trace=go.Pie(labels=labels,values=values)

py.iplot([trace])
```

Out [3]:



圆环图

Donut Chart

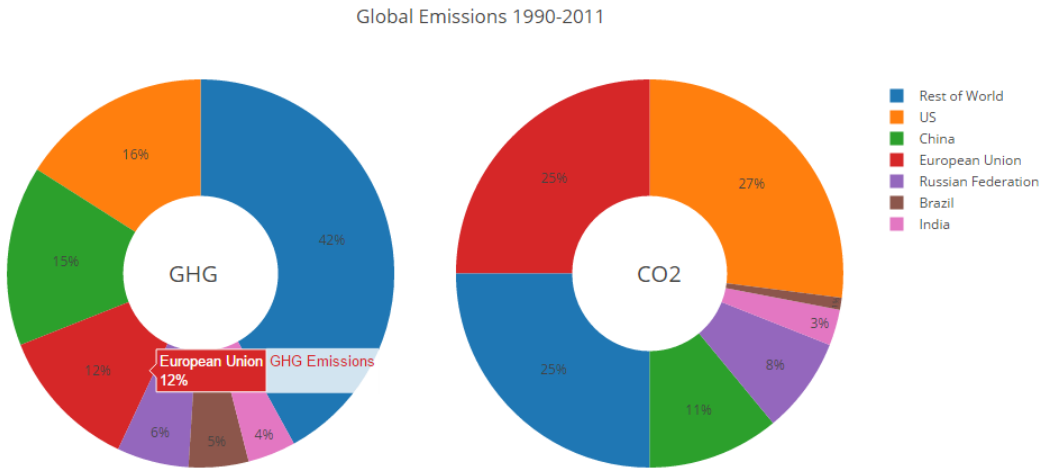
In [4]:

```
import plotly.plotly as py
import plotly.graph_objs as go

fig = {
  "data": [
    {
      "values": [16, 15, 12, 6, 5, 4, 42],
      "labels": [
        "US",
        "China",
        "European Union",
        "Russian Federation",
        "Brazil",
        "India",
        "Rest of World"
      ],
      "domain": {"x": [0, .48]},
      "name": "GHG Emissions",
      "hoverinfo":"label+percent+name",
```

```
      "hole": .4,
      "type": "pie"
    },
    {
      "values": [27, 11, 25, 8, 1, 3, 25],
      "labels": [
        "US",
        "China",
        "European Union",
        "Russian Federation",
        "Brazil",
        "India",
        "Rest of World"
      ],
      "text":"CO2",
      "textposition":"inside",
      "domain": {"x": [.52, 1]},
      "name": "CO2 Emissions",
      "hoverinfo":"label+percent+name",
      "hole": .4,
      "type": "pie"
    }],
  "layout": {
      "title":"Global Emissions 1990-2011",
      "annotations": [
          {
              "font": {
                  "size": 20
              },
              "showarrow": False,
              "text": "GHG",
              "x": 0.20,
              "y": 0.5
          },
          {
              "font": {
                  "size": 20
              },
              "showarrow": False,
              "text": "CO2",
```

```
                "x": 0.8,
                "y": 0.5
            }
        ]
    }
}
py.iplot(fig)
```

Out [4]:



饼图子图集

Pie Chart Subplots

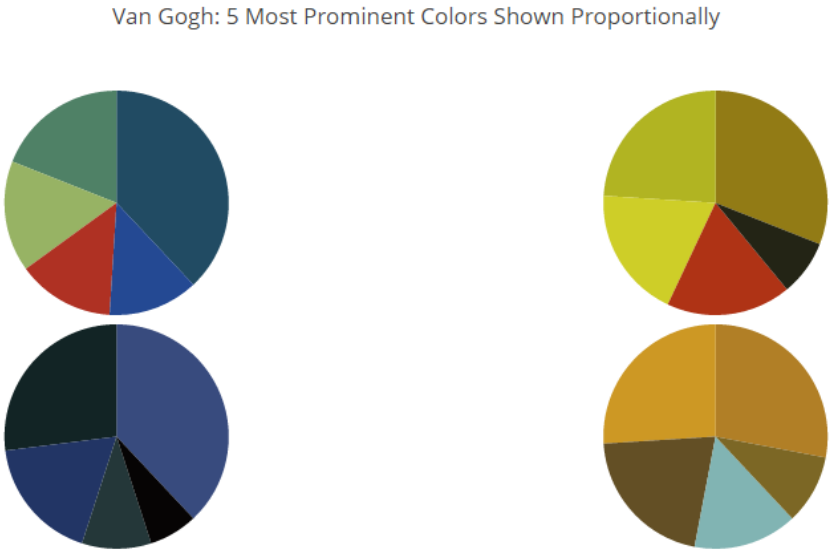
In [5]:

```
import plotly.plotly as py
from plotly.graph_objs import *

fig = {
    'data': [
        {
            'labels': ['1st', '2nd', '3rd', '4th', '5th'],
            'values': [38, 27, 18, 10, 7],
            'type': 'pie',
            'name': 'Starry Night',
            'marker': {'colors': ['rgb(56, 75, 126)',
```

```
                        'rgb(18, 36, 37)',
                        'rgb(34, 53, 101)',
                        'rgb(36, 55, 57)',
                        'rgb(6, 4, 4)']},
    'domain': {'x': [0, .48],
               'y': [0, .49]},
    'hoverinfo':'label+percent+name',
    'textinfo':'none'
},
{
    'labels': ['1st', '2nd', '3rd', '4th', '5th'],
    'values': [28, 26, 21, 15, 10],
    'marker': {'colors': ['rgb(177, 127, 38)',
                        'rgb(205, 152, 36)',
                        'rgb(99, 79, 37)',
                        'rgb(129, 180, 179)',
                        'rgb(124, 103, 37)']},
    'type': 'pie',
    'name': 'Sunflowers',
    'domain': {'x': [.52, 1],
               'y': [0, .49]},
    'hoverinfo':'label+percent+name',
    'textinfo':'none'

},
{
    'labels': ['1st', '2nd', '3rd', '4th', '5th'],
    'values': [38, 19, 16, 14, 13],
    'marker': {'colors': ['rgb(33, 75, 99)',
                        'rgb(79, 129, 102)',
                        'rgb(151, 179, 100)',
                        'rgb(175, 49, 35)',
                        'rgb(36, 73, 147)']},
    'type': 'pie',
    'name': 'Irises',
    'domain': {'x': [0, .48],
               'y': [.51, 1]},
    'hoverinfo':'label+percent+name',
    'textinfo':'none'
},
```

```
        {
            'labels': ['1st', '2nd', '3rd', '4th', '5th'],
            'values': [31, 24, 19, 18, 8],
            'marker': {'colors': ['rgb(146, 123, 21)',
                                  'rgb(177, 180, 34)',
                                  'rgb(206, 206, 40)',
                                  'rgb(175, 51, 21)',
                                  'rgb(35, 36, 21)']},
            'type': 'pie',
            'name':'The Night Café',
            'domain': {'x': [.52, 1],
                       'y': [.51, 1]},
            'hoverinfo':'label+percent+name',
            'textinfo':'none'
        }
    ],
    'layout': {'title': 'Van Gogh: 5 Most Prominent Colors Shown Proportionally',
               'showlegend': False}
}

py.iplot(fig)
```

Out [5]:



参考

Reference

See https://plot.ly/python/reference/#pie

for more information and attribute options!

访问 **https://plot.ly/python/reference/#pie**

可获得更多资讯信息和属性选项

的时间序列

Time Series in Python

How to plot date and time in python.

如何在 python 中描述日期和时间

Plotly 入门

New to Plotly?

Plotly's Python library is free and open source! Get started by downloading the client and reading the primer.

You can set up Plotly to work in online or offline mode, or in jupyter notebooks.

We also have a quick-reference cheatsheet (new!) to help you get started!

Plotly 是一个免费并开源的 python 库，可以通过 Get started 下载客户端程序并阅读概要。

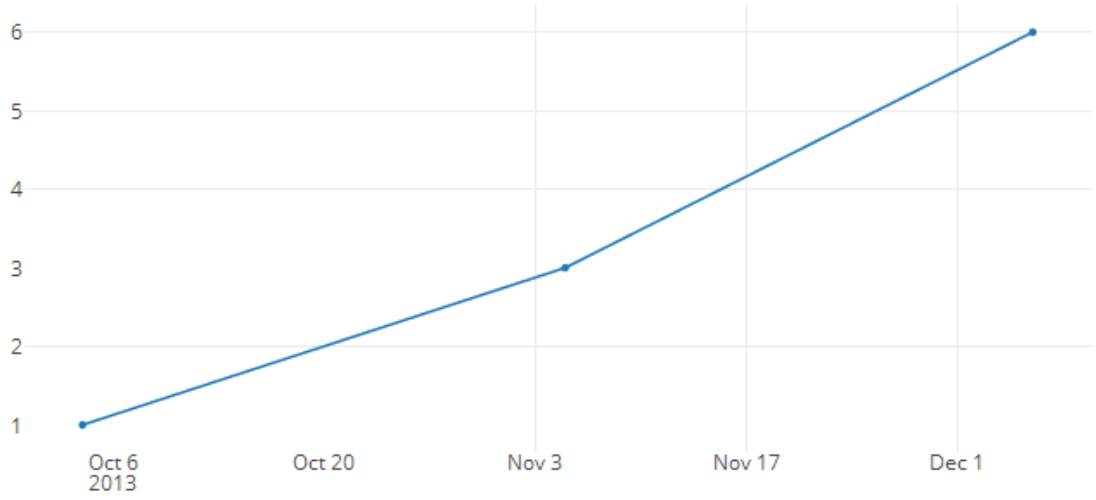
你可以安装 plotly，并且以“在线”和“离线”两种方式运行，或者在著名的 jupyter notebooks. 上运行。

我们也同样有一个快速参考文档 cheatsheet 来帮助你入门 plotly。以下是 plotly 的绘图能力简介。

时间序列绘图

Time Series Plot with datetime Objects

```
import plotly.plotly as py
import plotly.graph_objs as go

from datetime import datetime
x = [datetime(year=2013, month=10, day=04),
     datetime(year=2013, month=11, day=05),
     datetime(year=2013, month=12, day=06)]

data = [go.Scatter(x=x,y=[1, 3, 6])]
py.iplot(data)
```

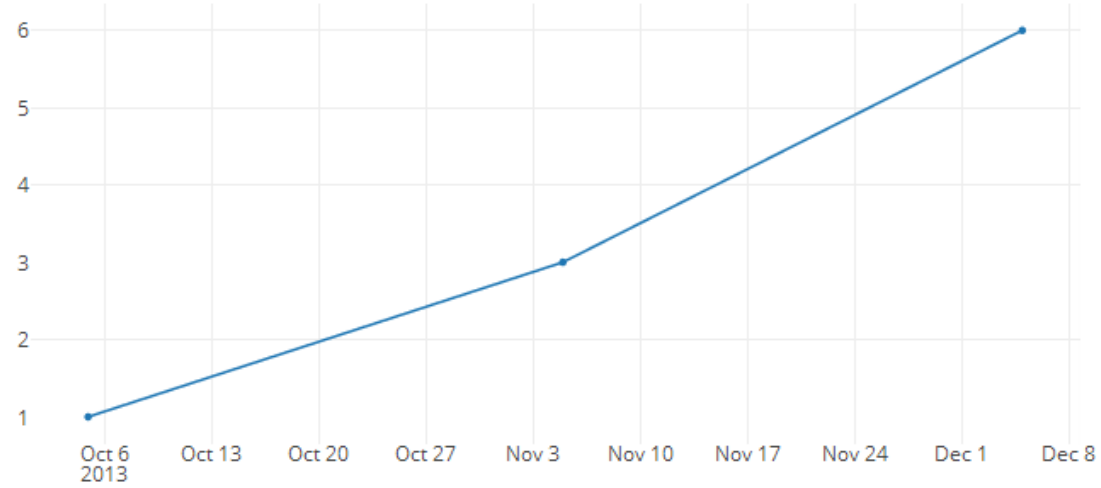


日期字符串的使用方式

Date Strings

```
import plotly.plotly as py
import plotly.graph_objs as go

data = [go.Scatter(
        x=['2013-10-04 22:23:00', '2013-11-04 22:23:00', '2013-12-04 22:23:00'],
```

```
        y=[1, 3, 6])]
py.iplot(data)
```

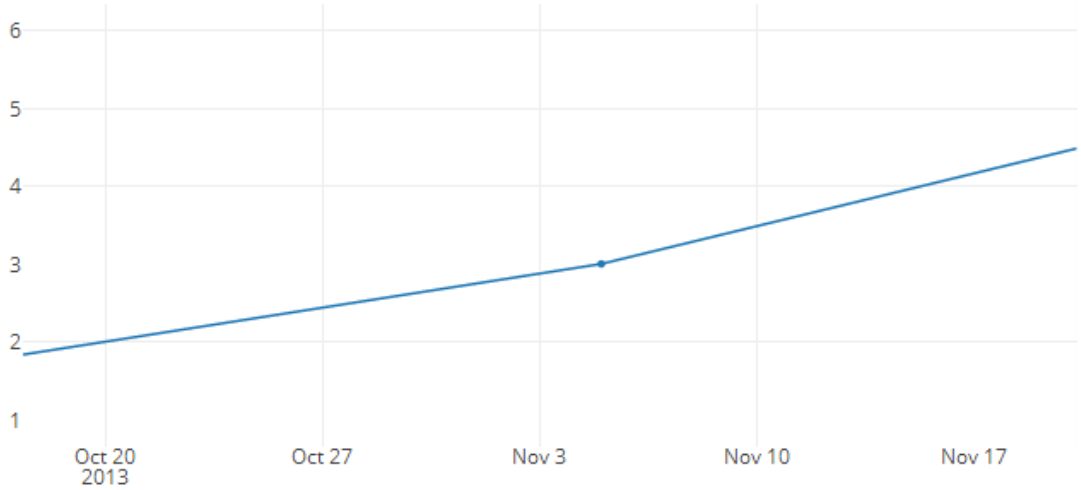


Time Series Plot with Custom Date Range

通过自由约束日期范围进行时间序列绘图

```
import plotly.plotly as py
import plotly.graph_objs as go

import datetime

def to_unix_time(dt):
    epoch =  datetime.datetime.utcfromtimestamp(0)
    return (dt - epoch).total_seconds() * 1000

x = [datetime.datetime(year=2013, month=10, day=04),
    datetime.datetime(year=2013, month=11, day=05),
    datetime.datetime(year=2013, month=12, day=06)]
data = [go.Scatter(
          x=x,
          y=[1, 3, 6])]

layout = go.Layout(xaxis = dict(
              range = [to_unix_time(datetime.datetime(2013, 10, 17)),
```

```
                    to_unix_time(datetime.datetime(2013, 11, 20))]
    ))

fig = go.Figure(data = data, layout = layout)
py.iplot(fig)
```



参考

Reference

See https://plot.ly/python/reference/#layout-xaxis-rangeslider and https://plot.ly/python/reference/#layout-xaxis-rangeselector for more information and chart attribute options!

如想进一步了解，可以去参阅 https://plot.ly/python/reference/#layout-xaxis-rangeslider 和 https://plot.ly/python/reference/#layout-xaxis-rangeselector 所链接的内容来获得更多信息和画图属性特征。

滑动选择控件

Range Slider and Selector in Python

Plotly 入门

New to Plotly?

Plotly's Python library is free and open source! Get started by downloading the client and reading the primer.
You can set up Plotly to work in online or offline mode, or in jupyter notebooks.
We also have a quick-reference cheatsheet (new!) to help you get started!

Plotly 是一个免费并开源的 python 库，可以通过 Get started 下载客户端程序并阅读概要。
你可以安装 plotly，并且以“在线”和“离线”两种方式运行，或者在著名的 jupyter notebooks. 上运行。
我们也同样有一个快速参考文档 cheatsheet 来帮助你入门 plotly。以下是 plotly 的绘图能力简介。

版本检查

Version Check

Note: range sliders and range selectors are available in version **1.9.7+**
Run pip install plotly --upgrade to update your Plotly version
注：在 1.9.7 版本开始才能使用范围滑块和选择功能。

```
import plotly
plotly.__version__
```

'1.12.9'

范围滑块和范围选择的基本使用示范

Basic Range Slider and Range Selectors

```
import plotly.plotly as py
import plotly.graph_objs as go

from datetime import datetime
import pandas.io.data as web

df = web.DataReader("aapl", 'yahoo',
                    datetime(2007, 10, 1),
                    datetime(2009, 4, 1))

trace = go.Scatter(x=df.index,
                   y=df.High)

data = [trace]
layout = dict(
    title='Time series with range slider and selectors',
    xaxis=dict(
        rangeselector=dict(
            buttons=list([
                dict(count=1,
                     label='1m',
                     step='month',
                     stepmode='backward'),
                dict(count=6,
                     label='6m',
                     step='month',
                     stepmode='backward'),
                dict(count=1,
                    label='YTD',
                    step='year',
                    stepmode='todate'),
                dict(count=1,
                    label='1y',
                    step='year',
                    stepmode='backward'),
                dict(step='all')
```

```
                ])
            ),
            rangeslider=dict(),
            type='date'
        )
    )
    fig = dict(data=data, layout=layout)
    py.iplot(fig)
```

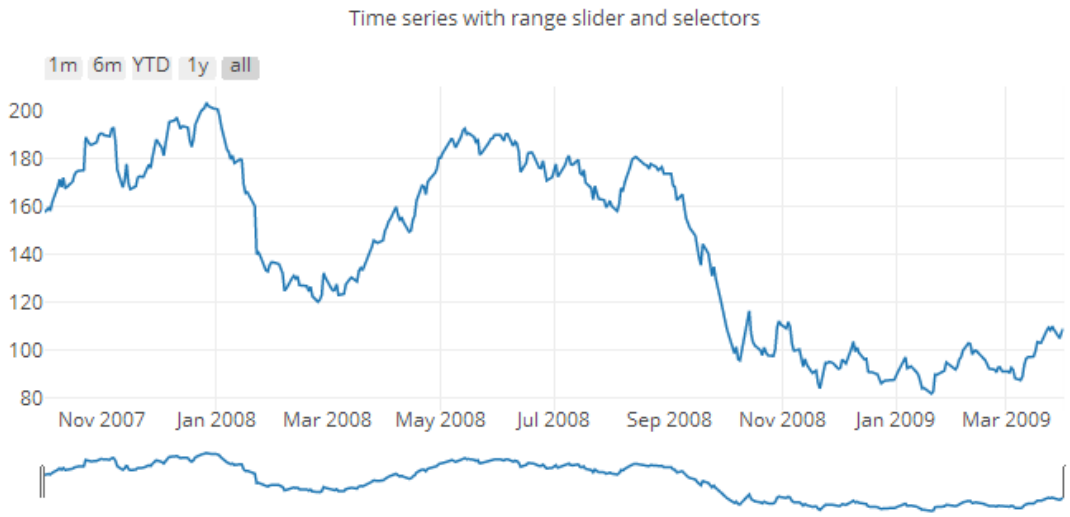
/Users/brandendunbar/Desktop/test/venv/lib/python2.7/site-packages/pandas/io/data.py:35: FutureWarning:

The pandas.io.data module is moved to a separate package (pandas-datareader) and will be removed from pandas in a future version.

After installing the pandas-datareader package (https://github.com/pydata/pandas-datareader), you can change the import ``from pandas.io import data, wb`` to ``from pandas_datareader import data, wb``.

需要说明的是 pandas.io.data 模块将被移入 pandas-datareader 这个单独模块中，并且在未来的 pandas 版本中删除。

在安装好 pandas-datareader 模块后 (参考链接 https://github.com/pydata/pandas-datareader), 你需要改变模块的导入方式，从``import from pandas.io import data, wb``改到 ``import from pandas_datareader import data, wb``，即可实现原有的功能。



参考阅读

Reference

See https://plot.ly/python/reference/#layout-xaxis-rangeselector
and https://plot.ly/python/reference/#layout-xaxis-rangeslider
for more information and attribute options!

参考阅读

**你可以查阅 https://plot.ly/python/reference/#layout-xaxis-rangeselector
和 https://plot.ly/python/reference/#layout-xaxis-rangeslider 获取更多的信息和属性选项。**

计量图表

Gauge Charts in Python

How to make guage meter charts in Python with Plotly.
如何使用 Plotly 在 Python 里绘制仪表计量图表。

刚开始接触 Plotly？

New to Plotly?

Plotly's Python library is free and open source! Get started by downloading the client and reading the primer.
You can set up Plotly to work in online or offline mode, or in jupyter notebooks.
We also have a quick-reference cheatsheet (new!) to help you get started!!

Plotly 的 Python 库是免费并且开源的。Get started by downloading the client and reading the primer。

你可以设置 Plotly 以 online 或者 offline 的方式运行，也可以在 jupyter notebooks 中运行。

我们同样也有一个快速参考的 cheatsheet (new!)来帮助新手学习。

计量图表概述

Gauge Chart Outline

We will use donut charts with custom colors to create a semi-circular gauge meter, such that lower half of the chart is invisible(same color as background).

我们将使用 donut（圆环）图表并自定义颜色来创建一个 semi-circular（半圆形的）计量图表，这样下半个图表是看不见的（与背景颜色相同）。

This semi-circular meter will be overlapped on a base donut chart to create the analog range of the meter. We will have to rotate the base chart to align the range marks in the edges of meter's section, because Plotly places them at the center of a pie section.

这个semi-circular（半圆形的）仪表重叠在donut（圆环）图表之上，为仪表增加了analog range（模拟量程）。我们必须旋转基本图表与仪表的距离标记的边缘对其，因为在默认情况下Plotly将他们放置在饼图的中心。

基本图表（旋转）

Base Chart (rotated)

To make a gauge meter with 5 equally sized sections, we will create 6 sections in the base chart. So that center(position of label) aligns with the edges of each section.

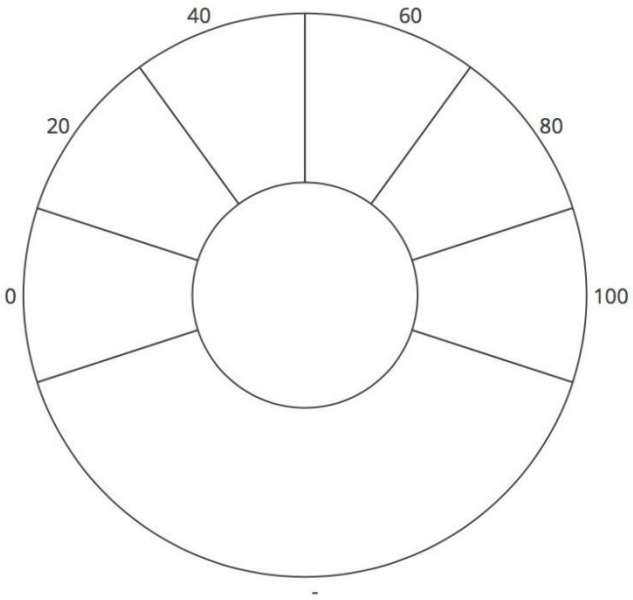
为了使gauge meter（计量仪表）分成大小相等的5部分，我们会创建一个6部分的基本图表。这样中心（标签的位置）与每个部分的边缘对齐。

In [1]:

```
import plotly.plotly as py
import plotly.graph_objs as go

base_chart = {
    "values": [40, 10, 10, 10, 10, 10, 10],
    "labels": ["-", "0", "20", "40", "60", "80", "100"],
    "domain": {"x": [0, .48]},
    "marker": {
        "colors": [
            'rgb(255, 255, 255)',
            'rgb(255, 255, 255)',
            'rgb(255, 255, 255)',
            'rgb(255, 255, 255)',
            'rgb(255, 255, 255)',
            'rgb(255, 255, 255)',
            'rgb(255, 255, 255)'
        ],
```

```
        "line": {
            "width": 1
        }
    },
    "name": "Gauge",
    "hole": .4,
    "type": "pie",
    "direction": "clockwise",
    "rotation": 108,
    "showlegend": False,
    "hoverinfo": "none",
    "textinfo": "label",
    "textposition": "outside"
}
```

Outline of the generated base chart will look like the one below.
Based chart（基本图表）生成的轮廓看起来像下面这样。



仪表图

Meter Chart

Now we will superimpose our semi-circular meter on top of this.
现在，我们要在这之上叠加semi-circular（半圆形的）图表。

For that, we will also use 6 sections, but one of them will be invisible to form the lower half (colored same as the background).

为此，我们还将使用6个部分，但其中一个是不可见的，形成下半个部分的图表（颜色和背景一样）。

In [2]:

```
meter_chart = {
    "values": [50, 10, 10, 10, 10, 10],
    "labels": ["Log Level", "Debug", "Info", "Warn", "Error", "Fatal"],
    "marker": {
        'colors': [
            'rgb(255, 255, 255)',
            'rgb(232,226,202)',
            'rgb(226,210,172)',
            'rgb(223,189,139)',
            'rgb(223,162,103)',
            'rgb(226,126,64)'
        ]
    },
    "domain": {"x": [0, 0.48]},
    "name": "Gauge",
    "hole": .3,
    "type": "pie",
    "direction": "clockwise",
    "rotation": 90,
    "showlegend": False,
    "textinfo": "label",
    "textposition": "inside",
    "hoverinfo": "none"
```

```
}
```

You can see that the first section's value is equal to the sum of other sections.

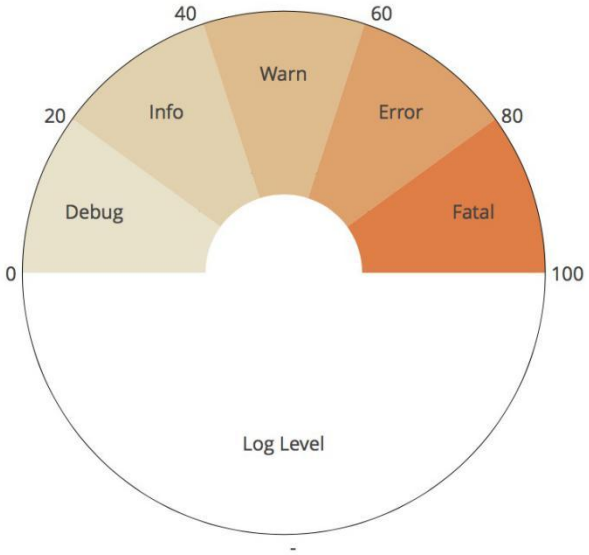
你可以看到第一部分的值等于其他所有部分的总和。

We are using rotation and direction parameters to start the sections from 3 o'clock [rotation=90] instead of the default value of 12 o'clock [rotation=0].

我们使用rotation和direction参数从3点钟[rotation=90]位置开始为仪表部分而非默认的12点钟位置[rotation=0]。

After imposing on the base chart, it'll look like this.

在基本图表上叠加之后，它将如下所示。



刻度

DIAL

Now we need a dial to show the current position in the meter at a particular time.

现在我们需要使用dial（刻度）表示在特定时间仪表的当前位置。

Plotly's path shape can be used for this. A nice explanation of SVG path is available here by Mozilla.

We can use a filled triangle to create our Dial.

Plotly的路径形状可被用与此。SVG路径的一个很好的解释在此，由Mozilla所提供。我们可以使用一个filled triangle（填充的三角）形来创建我们的Dial（刻度）。

```
...
'shapes': [
    {
        'type': 'path',
        'path': 'M 0.235 0.5 L 0.24 0.62 L 0.245 0.5 Z',
        'fillcolor': 'rgba(44, 160, 101, 0.5)',
        'line': {
            'width': 0.5
        },
        'xref': 'paper',
        'yref': 'paper'
    }
]
...
```

For the filled-triangle, the first point (0.235, 0.5) is left to the center of meter (0.24, 0.5), the second point (0.24 0.62) is representing the current position on the semi-circle and the third point (0.245, 0.5) is just right to the center.

对于filled-triangle（填充的三角），第一个点（0.235，0.5）在仪表中心的左侧，第二个点（0.24，0.62）代表指正在当前semi-circle（半圆）所在的位置，第三个点（0.245，0.5）正好在中心的右侧。

M represents the 'Move' command that moves cursor to a particular point, L is the 'Line To' command and Z represents the 'Close Path' command. This way, this path string makes a triangle with those three points.

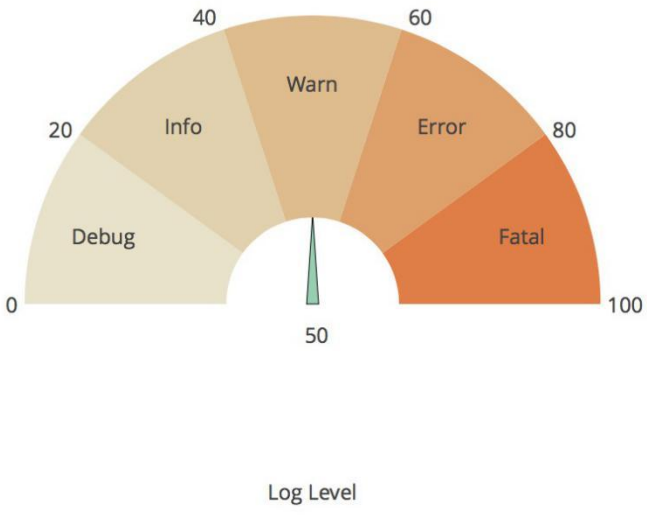
M代表'Move'（移动）指令，将指针移动到某个特定的点上，L是’ Line To’ 指令，z代表'Close Path'（关闭路径）指令。如此，这个路径字符串通过三个点创建了一个三角形。

In [3]:

```
layout = {
    'xaxis': {
        'showticklabels': False,
        'autotick': False,
        'showgrid': False,
        'zeroline': False,
    },
    'yaxis': {
        'showticklabels': False,
        'autotick': False,
        'showgrid': False,
        'zeroline': False,
    },
    'shapes': [
        {
            'type': 'path',
            'path': 'M 0.235 0.5 L 0.24 0.65 L 0.245 0.5 Z',
            'fillcolor': 'rgba(44, 160, 101, 0.5)',
            'line': {
                'width': 0.5
            },
            'xref': 'paper',
            'yref': 'paper'
        }
    ],
    'annotations': [
```

```
        {
            'xref': 'paper',
            'yref': 'paper',
            'x': 0.23,
            'y': 0.45,
            'text': '50',
            'showarrow': False
        }
    ]
}

# we don't want the boundary now
base_chart['marker']['line']['width'] = 0

fig = {"data": [base_chart, meter_chart],
       "layout": layout}
py.iplot(fig, filename='gauge-meter-chart')
```

Out [3]:



参考

Reference

See https://plot.ly/python/reference/ for more information and chart attribute options!

查看 https://plot.ly/python/reference/，获取更多的信息与图表属性的选项！

表格

Tables in Python

How to make guage meter charts in Python with Plotly.

如何使用 Plotly 在 Python 里绘制仪表计量图表。

刚开始接触 Plotly？

New to Plotly?

Plotly's Python library is free and open source! Get started by downloading the client and reading the primer.

You can set up Plotly to work in online or offline mode, or in jupyter notebooks.

We also have a quick-reference cheatsheet (new!) to help you get started!!

Plotly 的 Python 库是免费并且开源的。Get started by downloading the client and reading the primer。

你可以设置 Plotly 以 online 或者 offline 的方式运行，也可以在 jupyter notebooks 中运行。

我们同样也有一个快速参考的 cheatsheet (new!)来帮助新手学习。

版本检查

Version Check

Note: Tables are available in version 1.9.2+
注意：1.9.2版本及以上支持表格

Run pip install plotly --upgrade to update your Plotly version

运行pip install plotly --upgrade to更新你的Plotly版本。

In [1]:
```
import plotly
plotly.__version__
```

Out [1]:
```
'1.12.9'
```

简单表格

Simple Table

In [2]:
```
import plotly.plotly as py
from plotly.tools import FigureFactory as FF

data_matrix = [['Country', 'Year', 'Population'],
               ['United States', 2000, 282200000],
               ['Canada', 2000, 27790000],
               ['United States', 2005, 295500000],
```

```
            ['Canada', 2005, 32310000],
            ['United States', 2010, 309000000],
            ['Canada', 2010, 34000000]]

table = FF.create_table(data_matrix)
py.iplot(table, filename='simple_table'
```

Out [2]:

Country	Year	Population
United States	2000	282200000
Canada	2000	27790000
United States	2005	295500000
Canada	2005	32310000
United States	2010	309000000
Canada	2010	34000000

EDIT CHART

添加链接

Add Links

In [3]:

```
import plotly.plotly as py
from plotly.tools import FigureFactory as FF

data_matrix = [['User', 'Language', 'Chart Type', '# of Views'],
            ['<a href="https://plot.ly/~empet/folder/home">empet</a>',
             '<a href="https://plot.ly/python/">Python</a>',
             '<a href="https://plot.ly/~empet/8614/">Network Graph</a>',
             298],
            ['<a href="https://plot.ly/~Grondo/folder/home">Grondo</a>',
             '<a href="https://plot.ly/matlab/">Matlab</a>',
             '<a href="https://plot.ly/~Grondo/42/">Subplots</a>',
             356],
            ['<a href="https://plot.ly/~Dreamshot/folder/home">Dreamshot</a>',
             '<a href="https://help.plot.ly/tutorials/">Web App</a>',
```

```
                '<a href="https://plot.ly/~Dreamshot/6575/_2014-us-city-
populations/">Bubble Map</a>',
                262],
               ['<a
href="https://plot.ly/~FiveThirtyEight/folder/home">FiveThirtyEight</a>',
                '<a href="https://help.plot.ly/tutorials/">Web App</a>',
                '<a href="https://plot.ly/~FiveThirtyEight/30/">Scatter</a>',
                692],
               ['<a href="https://plot.ly/~cpsievert/folder/home">cpsievert</a>',
                '<a href="https://plot.ly/r/">R</a>',
                '<a href="https://plot.ly/~cpsievert/1130/">Surface</a>',
                302]]

    table = FF.create_table(data_matrix)
    py.iplot(table, filename='linked_table')
```

Out [3]:

User	Language	Chart Type	# of Views
empet	Python	Network Graph	298
Grondo	Matlab	Subplots	356
Dreamshot	Web App	Bubble Map	262
FiveThirtyEight	Web App	Scatter	692
cpsievert	R	Surface	302

EDIT CHART

使用 LaTex

Use LaTeX

In [4]:

```
import plotly.plotly as py
from plotly.tools import FigureFactory as FF
```

```
data_matrix = [['Name', 'Equation'],
               ['Pythagorean Theorem', '$a^{2}+b^{2}=c^{2}$'],
               ['Euler\'s Formula', '$F-E+V=2$'],
               ['The Origin of Complex Numbers', '$i^{2}=-1$'],
               ['Einstein\'s Theory of Relativity', '$E=m c^{2}$']]

table = FF.create_table(data_matrix)
py.iplot(table, filename='latex_table')
```

Out [4]:

Name	Equation
Pythagorean Theorem	$a^{2}+b^{2}=c^{2}$
Euler's Formula	$F-E+V=2$
The Origin of Complex Numbers	$i^{2}=-1$
Einstein's Theory of Relativity	$E=m c^{2}$

EDIT CHART

使用 Pandas 的 Dataframe

Use a Panda's Dataframe

In [5]:

```
import plotly.plotly as py
from plotly.tools import FigureFactory as FF

import pandas as pd

df = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/gapminderDataFiveYear.csv')
df_sample = df[100:120]

table = FF.create_table(df_sample)
```

```
py.iplot(table, filename='pandas_table')
```

Out [5]:

country	year	pop	continent	lifeExp	gdpPercap
Bangladesh	1972	70759295.0	Asia	45.252	630.2336265
Bangladesh	1977	80428306.0	Asia	46.923	659.8772322
Bangladesh	1982	93074406.0	Asia	50.009	676.9818656
Bangladesh	1987	103764241.0	Asia	52.819	751.9794035
Bangladesh	1992	113704579.0	Asia	56.018	837.8101643
Bangladesh	1997	123315288.0	Asia	59.412	972.7700352
Bangladesh	2002	135656790.0	Asia	62.013	1136.39043
Bangladesh	2007	150448339.0	Asia	64.062	1391.253792
Belgium	1952	8730405.0	Europe	68.0	8343.105127
Belgium	1957	8989111.0	Europe	69.24	9714.960623
Belgium	1962	9218400.0	Europe	70.25	10991.20676
Belgium	1967	9556500.0	Europe	70.94	13149.04119
Belgium	1972	9709100.0	Europe	71.44	16672.14356
Belgium	1977	9821800.0	Europe	72.8	19117.97448
Belgium	1982	9856303.0	Europe	73.93	20979.84589
Belgium	1987	9870200.0	Europe	75.35	22525.56308
Belgium	1992	10045622.0	Europe	76.46	25575.57069
Belgium	1997	10199787.0	Europe	77.53	27561.19663
Belgium	2002	10311970.0	Europe	78.32	30485.88375
Belgium	2007	10392226.0	Europe	79.441	33692.60508

EDIT CHART

包含和索引列

Include and Index Column

In [7]:

```
import plotly.plotly as py
from plotly.tools import FigureFactory as FF

from datetime import date
import pandas.io.data as web

di = web.DataReader("aapl", 'yahoo', date(2009, 1, 1), date(2009, 3, 1))
```

```
# Converting timestamp to date
di["Date1"] = di.index.date
di.set_index("Date1", drop=True, inplace=True)

table = FF.create_table(di, index=True, index_title='Date')
py.iplot(table, filename='index_table_pd')
```

Out [7]:

Date	Open	High	Low	Close	Volume	Adj Close
2009-01-02	85.880003	91.040001	85.16	90.750001	186503800.0	11.869133
2009-01-05	93.170003	96.179998	92.709999	94.580002	295402100.0	12.370056
2009-01-06	95.95	97.170001	92.389998	93.02	322327600.0	12.166024
2009-01-07	91.809999	92.500001	90.260003	91.01	188262200.0	11.903138
2009-01-08	90.43	93.150002	90.039998	92.699999	168375200.0	12.124172
2009-01-09	93.209997	93.380001	90.14	90.579997	136711400.0	11.846898
2009-01-12	90.460001	90.99	87.550001	88.66	154429100.0	11.595783
2009-01-13	88.239997	89.739998	86.35	87.709998	199599400.0	11.471533
2009-01-14	86.239998	87.250001	84.720003	85.329997	255416000.0	11.160254
2009-01-15	80.570002	84.120003	80.050003	83.379999	457908500.0	10.905215
2009-01-16	84.3	84.380002	80.399998	82.330002	261906400.0	10.767887
2009-01-20	81.929999	82.000001	78.200001	78.200001	229978700.0	10.227726
2009-01-21	79.390001	82.880001	79.309999	82.83	272317500.0	10.833281
2009-01-22	88.039999	89.999997	85.820002	88.36	352382100.0	11.556546
2009-01-23	86.819998	89.870001	86.499997	88.36	190942500.0	11.556546
2009-01-26	88.859998	90.969999	88.299998	89.640002	173059600.0	11.723957
2009-01-27	90.190001	91.549999	89.739998	90.73	154509600.0	11.866517
2009-01-28	92.119999	94.999998	91.499998	94.2	215351500.0	12.320356
2009-01-29	93.090001	94.340003	92.600003	92.999999	148182300.0	12.163408
2009-01-30	92.600003	93.62	90.009997	90.13	162869700.0	11.788043

2009-01-30	92.600003	93.62	90.009997	90.13	162869700.0	11.788043
2009-02-02	89.100003	92.000003	88.899999	91.509998	139561800.0	11.968532
2009-02-03	91.920001	93.380001	90.279997	92.979999	149827300.0	12.160793
2009-02-04	93.219997	96.25	93.100001	93.549998	202105400.0	12.235342
2009-02-05	92.77	97.250003	92.619997	96.459998	187311600.0	12.615939
2009-02-06	97.019998	99.999999	96.999997	99.719999	171802400.0	13.042313
2009-02-09	99.999999	103.000001	99.500001	102.510003	178752700.0	13.407216
2009-02-10	101.330003	102.510003	97.059999	97.830003	212265200.0	12.795121
2009-02-11	96.370003	98.31	95.770002	96.82	168743400.0	12.663024
2009-02-12	95.829997	99.75	95.829997	99.270002	204297100.0	12.983458
2009-02-13	98.990003	99.939998	98.120003	99.16	152244400.0	12.969071
2009-02-17	96.870001	97.039998	94.280002	94.530001	169559600.0	12.363516
2009-02-18	95.049999	95.849998	92.719999	94.369997	171194800.0	12.34259
2009-02-19	93.370001	94.250001	90.11	90.639998	230701100.0	11.854746
2009-02-20	89.399997	92.399999	89.000001	91.199998	187579000.0	11.927987
2009-02-23	91.650002	92.000003	86.509997	86.950001	196745500.0	11.372133
2009-02-24	87.449999	90.889997	87.000002	90.250003	201776400.0	11.803738
2009-02-25	89.860001	92.919997	89.25	91.159997	208263300.0	11.922756
2009-02-26	92.000003	92.919997	88.96	89.189999	157467100.0	11.665101
2009-02-27	87.930003	91.3	87.669997	89.310001	176664600.0	11.680796

EDIT CHART

自定义表格颜色

Custom Table Colors

A custom colorscale should be a list[list]:
[[0, 'Header_Color'],[.5, 'Odd_Row_Color'],[1, 'Even_Row_Color']]

自定义colorscale应该是list[list]:
[[0, 'Header_Color'],[.5, 'Odd_Row_Color'],[1, 'Even_Row_Color']]

In [8]:

```
import plotly.plotly as py
from plotly.tools import FigureFactory as FF

import pandas as pd
```

```
df = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/gapminderDataFiveYear.csv')
df_sample = df[400:410]

colorscale = [[0, '#4d004c'],[.5, '#f2e5ff'],[1, '#ffffff']]

table = FF.create_table(df_sample, colorscale=colorscale)
py.iplot(table, filename='color_table')
```

Out [8]:

country	year	pop	continent	lifeExp	gdpPercap
Czech Republic	1972	9862158.0	Europe	70.29	13108.4536
Czech Republic	1977	10161915.0	Europe	70.71	14800.16062
Czech Republic	1982	10303704.0	Europe	70.96	15377.22855
Czech Republic	1987	10311597.0	Europe	71.58	16310.4434
Czech Republic	1992	10315702.0	Europe	72.4	14297.02122
Czech Republic	1997	10300707.0	Europe	74.01	16048.51424
Czech Republic	2002	10256295.0	Europe	75.51	17596.21022
Czech Republic	2007	10228744.0	Europe	76.486	22833.30851
Denmark	1952	4334000.0	Europe	70.78	9692.385245
Denmark	1957	4487831.0	Europe	71.81	11099.65935

自定义字体颜色

Custom Font Colors

In [9]:

```
import plotly.plotly as py
from plotly.tools import FigureFactory as FF

text = [['Team', 'Rank'], ['A', 1], ['B', 2], ['C', 3], ['D', 4], ['E', 5], ['F', 6]]

colorscale = [[0, '#272D31'],[.5, '#ffffff'],[1, '#ffffff']]
font=['#FCFCFC', '#00EE00', '#008B00', '#004F00', '#660000', '#CD0000', '#FF3030']

table = FF.create_table(text, colorscale=colorscale, font_colors=font)
```

```
table.layout.width=250
py.iplot(table, filename='font_table')
```

Out [9]:

Team	Rank
A	1
B	2
C	3
D	4
E	5
F	6

EDIT CHART

更改字体大小

Change Font Size

In [10]:

```
import plotly.plotly as py
from plotly.tools import FigureFactory as FF

data_matrix = [['Country', 'Year', 'Population'],
               ['United States', 2000, 282200000],
               ['Canada', 2000, 27790000],
               ['United States', 2005, 295500000],
               ['Canada', 2005, 32310000],
               ['United States', 2010, 309000000],
               ['Canada', 2010, 34000000]]

table = FF.create_table(data_matrix, index=True)

# Make text size larger
for i in range(len(table.layout.annotations)):
    table.layout.annotations[i].font.size = 20
```

```
py.iplot(table, filename='index_table')
```

Out [10]:

Country	Year	Population
United States	2000	282200000
Canada	2000	27790000
United States	2005	295500000
Canada	2005	32310000
United States	2010	309000000
Canada	2010	34000000

EDIT CHART

Tables with Graphs

表格与图

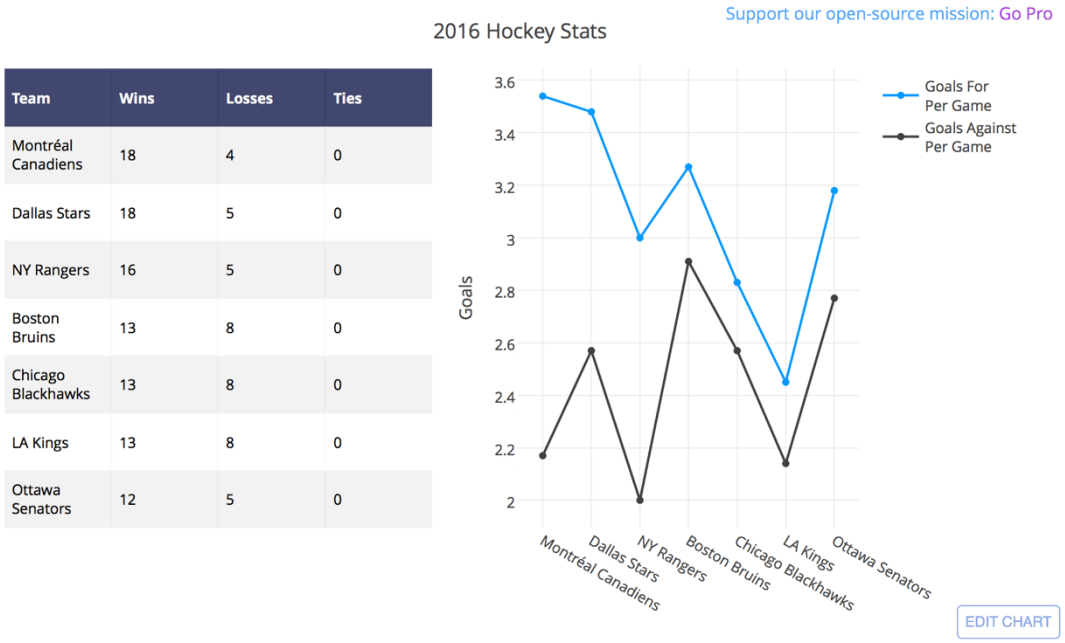
Tables with Graphs

In [11]:

```
import plotly.plotly as py
import plotly.graph_objs as go
from plotly.tools import FigureFactory as FF

# Add table data
table_data = [['Team', 'Wins', 'Losses', 'Ties'],
              ['Montréal<br>Canadiens', 18, 4, 0],
              ['Dallas Stars', 18, 5, 0],
              ['NY Rangers', 16, 5, 0],
              ['Boston<br>Bruins', 13, 8, 0],
              ['Chicago<br>Blackhawks', 13, 8, 0],
              ['LA Kings', 13, 8, 0],
              ['Ottawa<br>Senators', 12, 5, 0]]
# Initialize a figure with FF.create_table(table_data)
figure = FF.create_table(table_data, height_constant=60)

# Add graph data
teams = ['Montréal Canadiens', 'Dallas Stars', 'NY Rangers',
         'Boston Bruins', 'Chicago Blackhawks', 'LA Kings', 'Ottawa Senators']
```

```
GFPG = [3.54, 3.48, 3.0, 3.27, 2.83, 2.45, 3.18]
GAPG = [2.17, 2.57, 2.0, 2.91, 2.57, 2.14, 2.77]
# Make traces for graph
trace1 = go.Scatter(x=teams, y=GFPG,
                    marker=dict(color='#0099ff'),
                    name='Goals For<br>Per Game',
                    xaxis='x2', yaxis='y2')
trace2 = go.Scatter(x=teams, y=GAPG,
                    marker=dict(color='#404040'),
                    name='Goals Against<br>Per Game',
                    xaxis='x2', yaxis='y2')

# Add trace data to figure
figure['data'].extend(go.Data([trace1, trace2]))

# Edit layout for subplots
figure.layout.xaxis.update({'domain': [0, .5]})
figure.layout.xaxis2.update({'domain': [0.6, 1.]})
# The graph's yaxis MUST BE anchored to the graph's xaxis
figure.layout.yaxis2.update({'anchor': 'x2'})
figure.layout.yaxis2.update({'title': 'Goals'})
# Update the margins to add a title and see graph x-labels.
figure.layout.margin.update({'t':50, 'b':100})
figure.layout.update({'title': '2016 Hockey Stats'})

# Plot!
py.iplot(figure, filename='subplot_table')
```

Out [11]:

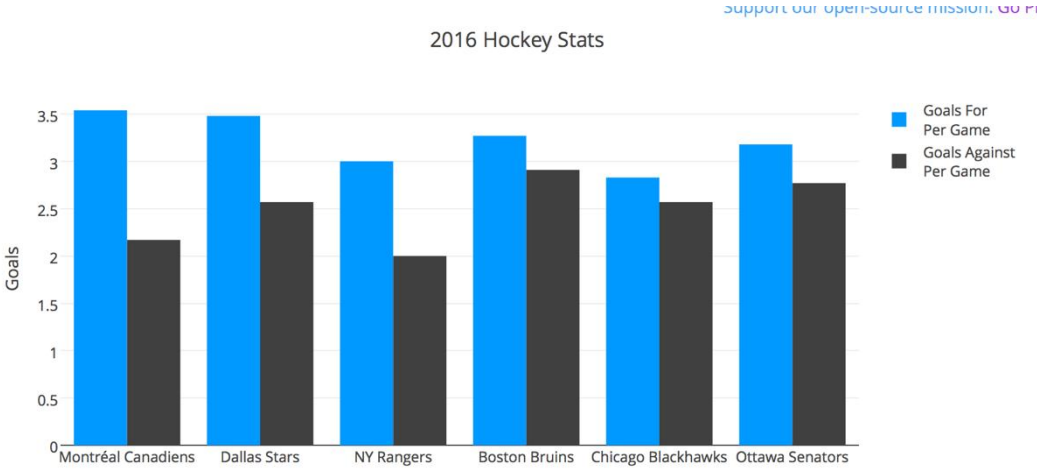


In [12]:

```
import plotly.plotly as py
import plotly.graph_objs as go
from plotly.tools import FigureFactory as FF

# Add table data
table_data = [['Team', 'Wins', 'Losses', 'Ties'],
              ['Montréal<br>Canadiens', 18, 4, 0],
              ['Dallas Stars', 18, 5, 0],
              ['NY Rangers', 16, 5, 0],
              ['Boston<br>Bruins', 13, 8, 0],
              ['Chicago<br>Blackhawks', 13, 8, 0],
              ['Ottawa<br>Senators', 12, 5, 0]]
# Initialize a figure with FF.create_table(table_data)
figure = FF.create_table(table_data, height_constant=60)

# Add graph data
teams = ['Montréal Canadiens', 'Dallas Stars', 'NY Rangers',
         'Boston Bruins', 'Chicago Blackhawks', 'Ottawa Senators']
GFPG = [3.54, 3.48, 3.0, 3.27, 2.83, 3.18]
GAPG = [2.17, 2.57, 2.0, 2.91, 2.57, 2.77]
# Make traces for graph
```

```
trace1 = go.Bar(x=teams, y=GFPG, xaxis='x2', yaxis='y2',
                marker=dict(color='#0099ff'),
                name='Goals For<br>Per Game')
trace2 = go.Bar(x=teams, y=GAPG, xaxis='x2', yaxis='y2',
                marker=dict(color='#404040'),
                name='Goals Against<br>Per Game')

# Add trace data to figure
figure['data'].extend(go.Data([trace1, trace2]))

# Edit layout for subplots
figure.layout.yaxis.update({'domain': [0, .45]})
figure.layout.yaxis2.update({'domain': [.6, 1]})
# The graph's yaxis2 MUST BE anchored to the graph's xaxis2 and vice versa
figure.layout.yaxis2.update({'anchor': 'x2'})
figure.layout.xaxis2.update({'anchor': 'y2'})
figure.layout.yaxis2.update({'title': 'Goals'})
# Update the margins to add a title and see graph x-labels.
figure.layout.margin.update({'t':75, 'l':50})
figure.layout.update({'title': '2016 Hockey Stats'})
# Update the height because adding a graph vertically will interact with
# the plot height calculated for the table
figure.layout.update({'height':800})

# Plot!
py.iplot(figure, filename='subplot_table_vertical')
```

Out [12]:



Team	Wins	Losses	Ties
Montréal Canadiens	18	4	0
Dallas Stars	18	5	0
NY Rangers	16	5	0
Boston Bruins	13	8	0
Chicago Blackhawks	13	8	0
Ottawa Senators	12	5	0

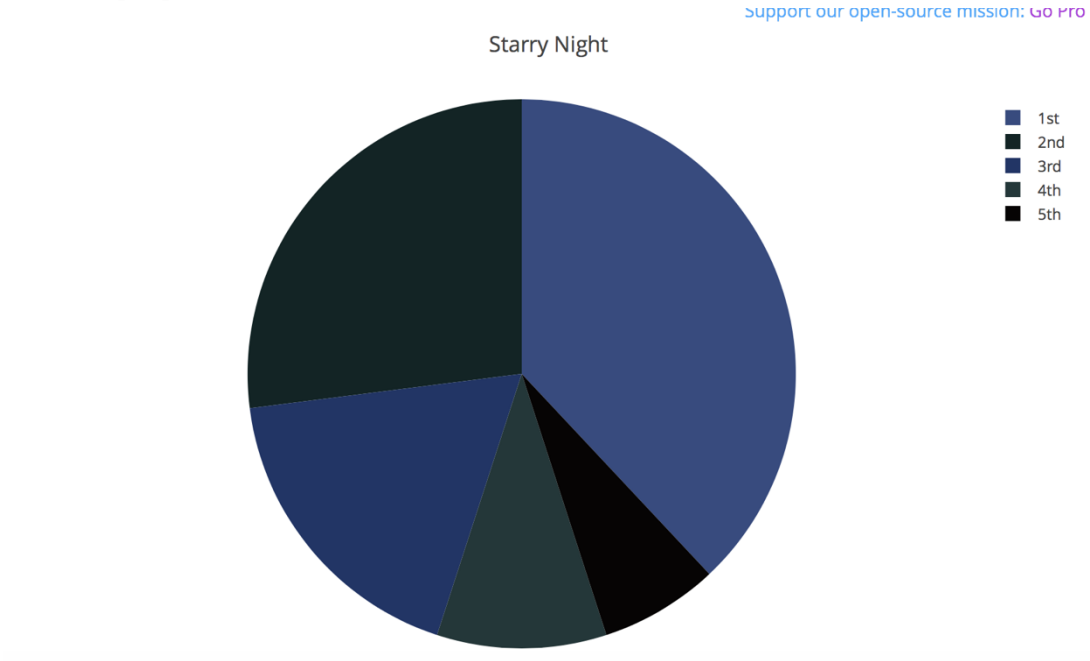
EDIT CHART

In [14]:

```
import plotly.plotly as py
import plotly.graph_objs as go
from plotly.tools import FigureFactory as FF

# Add table data
table_data = [['Prominence', 'Percent', 'RGB Value'],
              [1, '38%', 'rgb(56, 75, 126)'],
              [2, '27%', 'rgb(18, 36, 37)'],
              [3, '18%', 'rgb(34, 53, 101)'],
              [4, '10%', 'rgb(36, 55, 57)'],
              [5, '7%', 'rgb(6, 4, 4)']]
# Initialize a figure with FF.create_table(table_data)
figure = FF.create_table(table_data, height_constant=60)

# Add graph data
trace1 ={'labels': ['1st', '2nd', '3rd', '4th', '5th'],
        'values': [38, 27, 18, 10, 7],
        'type': 'pie',
        'name': 'Starry Night',
        'marker': {'colors': ['rgb(56, 75, 126)',
                              'rgb(18, 36, 37)',
                              'rgb(34, 53, 101)',
                              'rgb(36, 55, 57)',
                              'rgb(6, 4, 4)']},
           'domain': {'x': [0, 1],
                      'y': [.4, 1]},
           'hoverinfo':'label+percent+name',
```

```
        'textinfo':'none'
    }

# Add trace data to figure
figure['data'].extend(go.Data([trace1]))

# Edit layout for subplots
figure.layout.yaxis.update({'domain': [0, .30]})
# The graph's yaxis2 MUST BE anchored to the graph's xaxis2 and vice versa
# Update the margins to add a title and see graph x-labels.
figure.layout.margin.update({'t':75, 'l':50})
figure.layout.update({'title': 'Starry Night'})
# Update the height because adding a graph vertically will interact with
# the plot height calculated for the table
figure.layout.update({'height':800})

# Plot!
py.iplot(figure)
```

Out [14]:



Prominence	Percent	RGB Value
1	38%	rgb(56, 75, 126)
2	27%	rgb(18, 36, 37)
3	18%	rgb(34, 53, 101)
4	10%	rgb(36, 55, 57)
5	7%	rgb(6, 4, 4)

EDIT CHART

参考

Reference

In [15]:

```
help(FF.create_table)
```

```
Help on function create_table in module plotly.tools:

create_table(table_text, colorscale=None, font_colors=None, index=False,
index_title='', annotation_offset=0.45, height_constant=30, hoverinfo='none',
**kwargs)
    BETA function that creates data tables

    :param (pandas.Dataframe | list[list]) text: data for table.
    :param (str|list[list]) colorscale: Colorscale for table where the
        color at value 0 is the header color, .5 is the first table color
        and 1 is the second table color. (Set .5 and 1 to avoid the striped
        table effect). Default=[[0, '#66b2ff'], [.5, '#d9d9d9'],
        [1, '#ffffff']]
    :param (list) font_colors: Color for fonts in table. Can be a single
        color, three colors, or a color for each row in the table.
        Default=['#000000'] (black text for the entire table)
    :param (int) height_constant: Constant multiplied by # of rows to
        create table height. Default=30.
    :param (bool) index: Create (header-colored) index column index from
        Pandas dataframe or list[0] for each list in text. Default=False.
    :param (string) index_title: Title for index column. Default=''.
    :param kwargs: kwargs passed through plotly.graph_objs.Heatmap.
```

These kwargs describe other attributes about the annotated Heatmap trace such as the colorscale. For more information on valid kwargs call help(plotly.graph_objs.Heatmap)

Example 1: Simple Plotly Table

```
import plotly.plotly as py
from plotly.tools import FigureFactory as FF

text = [['Country', 'Year', 'Population'],
        ['US', 2000, 282200000],
        ['Canada', 2000, 27790000],
        ['US', 2010, 309000000],
        ['Canada', 2010, 34000000]]

table = FF.create_table(text)
py.iplot(table)
```

Example 2: Table with Custom Coloring

```
import plotly.plotly as py
from plotly.tools import FigureFactory as FF

text = [['Country', 'Year', 'Population'],
        ['US', 2000, 282200000],
        ['Canada', 2000, 27790000],
        ['US', 2010, 309000000],
        ['Canada', 2010, 34000000]]

table = FF.create_table(text,
                        colorscale=[[0, '#000000'],
                                    [.5, '#80beff'],
                                    [1, '#cce5ff']],
                        font_colors=['#ffffff', '#000000',
                                     '#000000'])
py.iplot(table)
```

Example 3: Simple Plotly Table with Pandas

```
```

```
import plotly.plotly as py
from plotly.tools import FigureFactory as FF

import pandas as pd

df =
pd.read_csv('http://www.stat.ubc.ca/~jenny/notOcto/STAT545A/examples/gapminder/
data/gapminderDataFiveYear.txt', sep='	')
df_p = df[0:25]

table_simple = FF.create_table(df_p)
py.iplot(table_simple)
```

Support our open-source mission: Go Pro

支持我们的开源任务：专业版

线性表图

Linear Gauge Charts
How to make interactive linear-guage charts in Python with Plotly.
如何在 Python 中使用 Plotly 创建交互式线性图表

线性表 shell 脚本

Linear Gauge Chart Shell
Note the following tutorial shows how to create a linear-gauge chart with 4 gauges. It's recommended to use a *width* between 600-1000px and *ticklen* should be *width*/20. These variables are definied in the code below.
注意，接下来的指导是展示如何使用四种规格创建一个线性表。建议使用 *width* 在 600-1000px 之间，*ticklen* 应该为 *width*/20。这些变量都定义在下面的代码中。

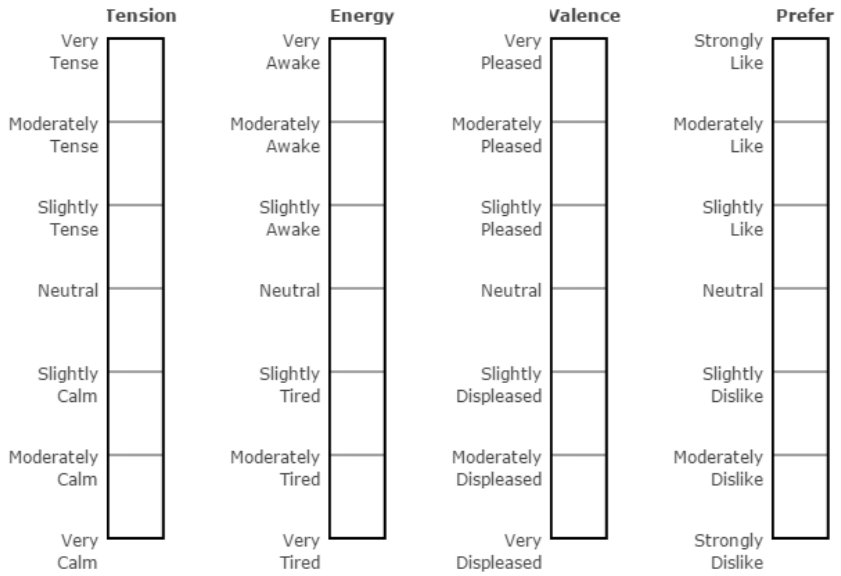
In [1]:

```
from plotly import tools
import plotly.plotly as py
import plotly.graph_objs as go

# Define Titles and Labels for Each Scale
```

```
scales = ['<b>Tension</b>', '<b>Energy</b>',
        '<b>Valence</b>', '<b>Prefer</b>']
scale1 = ['Very <br> Calm ', 'Moderately <br> Calm ',
        'Slightly <br> Calm ', 'Neutral ',
        'Slightly <br> Tense ', 'Moderately <br> Tense ',
        'Very <br> Tense ']
scale2 = ['Very <br> Tired ', 'Moderately <br> Tired ',
        'Slightly <br> Tired ', 'Neutral ',
        'Slightly <br> Awake ', 'Moderately <br> Awake ',
        'Very <br> Awake ']
scale3 = ['Very <br> Displeased ', 'Moderately <br> Displeased ',
        'Slightly <br> Displeased ', 'Neutral ',
        'Slightly <br> Pleased ', 'Moderately <br> Pleased ',
        'Very <br> Pleased ']
scale4 = ['Strongly <br> Dislike ', 'Moderately <br> Dislike ',
        'Slightly <br> Dislike ', 'Neutral ',
        'Slightly <br> Like ', 'Moderately <br> Like ',
        'Strongly <br> Like ']
scale_labels = [scale1, scale2, scale3, scale4]

# Add Scale Titles to the Plot
traces = []
for i in range(len(scales)):
    traces.append(go.Scatter(
        x=[0.6], # Pad the title - a longer scale title would need a higher value
        y=[6.25],
        text=scales[i],
        mode='text',
        hoverinfo='none',
        showlegend=False,
        xaxis='x'+str(i+1),
        yaxis='y'+str(i+1)
    ))

# Create Scales
## Since we have 7 lables, the scale will range from 0-6
shapes = []
for i in range(len(scales)):
    shapes.append({'type': 'rect',
                   'x0': .02, 'x1': 1.02,
                   'y0': 0, 'y1': 6,
                   'xref':'x'+str(i+1), 'yref':'y'+str(i+1)})

x_domains = [[0, .25], [.25, .5], [.5, .75], [.75, 1]] # Split for 4 scales
chart_width = 800

# Define X-Axes
xaxes = []
for i in range(len(scales)):
    xaxes.append({'domain': x_domains[i], 'range':[0, 4],
                  'showgrid': False, 'showline': False,
                  'zeroline': False, 'showticklabels': False})

# Define Y-Axes (and set scale labels)
## ticklen is used to create the segments of the scale,
## for more information see: https://plot.ly/python/reference/#layout-yaxis-ticklen
yaxes = []
for i in range(len(scales)):
    yaxes.append({'anchor':'x'+str(i+1), 'range':[-.5,6.5],
                  'showgrid': False, 'showline': False, 'zeroline': False,
```

```
                'ticks':'inside', 'ticklen': chart_width/20,
                'ticktext':scale_labels[i], 'tickvals':[0., 1., 2., 3., 4., 5., 6.]
                })

# Put all elements of the layout together
layout = {'shapes': shapes,
        'xaxis1': xaxes[0],
        'xaxis2': xaxes[1],
        'xaxis3': xaxes[2],
        'xaxis4': xaxes[3],
        'yaxis1': yaxes[0],
        'yaxis2': yaxes[1],
        'yaxis3': yaxes[2],
        'yaxis4': yaxes[3],
        'autosize': False,
        'width': chart_width,
        'height': 600
}

### ADD RATING DATA HERE ###

fig = dict(data=traces, layout=layout)
py.iplot(fig, filename='linear-gauge-layout')
```

Out[1]:



增加等级数据

Add Rating Data

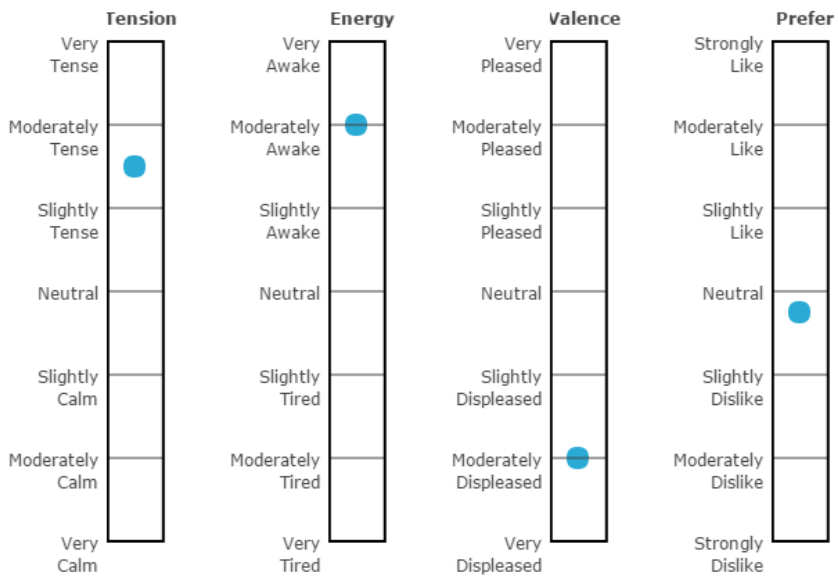
Ratings should be scaled between 0 - 6 to fit the y-values of the scales created above.

等级应该衡量在 0-6 之间，满足上面创建的规模中的 y 值。

In [2]:

```
ratings = [4.5, 5, 1, 2.75]

for i in range(len(ratings)):
    traces.append(go.Scatter(
            x=[0.5], y=ratings[i],
            xaxis='x'+str(i+1), yaxis='y'+str(i+1),
            mode='marker', marker={'size': 16, 'color': '#29ABD6'},
            text=ratings[i], hoverinfo='text', showlegend=False
    ))

fig = dict(data=traces, layout=layout)
py.iplot(fig, filename='linear-gauge')
```

Out[2]:



多图表类型

Multiple Chart Types in Python

How to design figures with multiple chart types in python.

如何在 Python 中设计多图表类型的数据

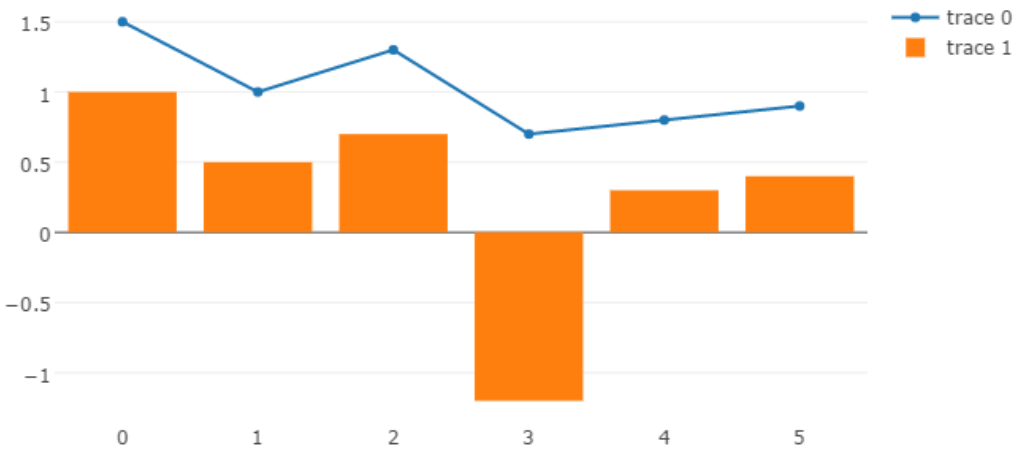
线条图和柱形图

Line Chart and a Bar Chart

In [1]:

```
ratings = [4.5, 5, 1, 2.75]

for i in range(len(ratings)):
    traces.append(go.Scatter(
            x=[0.5], y=ratings[i],
            xaxis='x'+str(i+1), yaxis='y'+str(i+1),
            mode='marker', marker={'size': 16, 'color': '#29ABD6'},
            text=ratings[i], hoverinfo='text', showlegend=False
    ))

fig = dict(data=traces, layout=layout)
py.iplot(fig, filename='linear-gauge')
```

Out[1]:



最速下降法的等高线和散点图

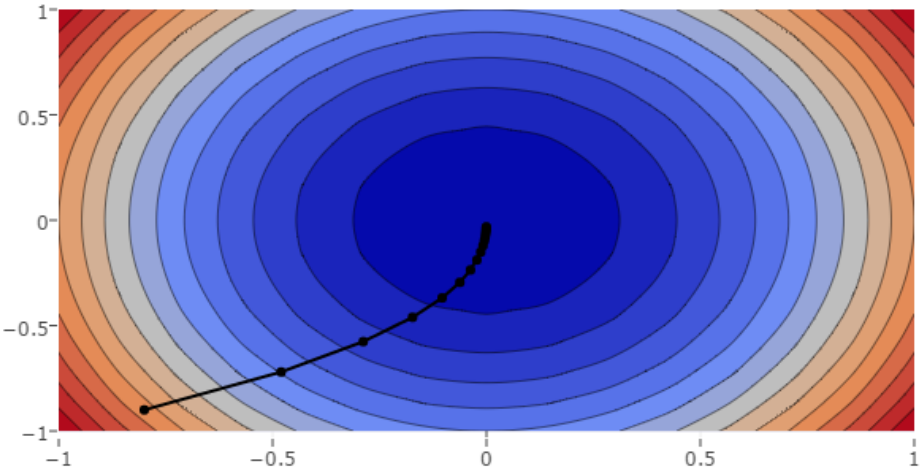
A Contour and Scatter Plot of the Method of Steepest Descent

In [2]:

```
import plotly.plotly as py
import plotly.graph_objs as go

import json
import six.moves.urllib
```

```
        response =
six.moves.urllib.request.urlopen('https://raw.githubusercontent.com/plotly/datasets/master/steepest.json
')
        data = json.load(response)

        trace1 = go.Contour(
            z=data['contour_z'][0],
            y=data['contour_y'][0],
            x=data['contour_x'][0],
            ncontours=30,
            showscale=False
        )
        trace2 = go.Scatter(
            x=data['trace_x'],
            y=data['trace_y'],
            mode='markers+lines',
            name='steepest',
            line=dict(
                color='black'
            )
        )

        data = [trace1, trace2]
        py.iplot(data, filename='contour-scatter')
```

Out[2]:



绘制 SVG 图形

How to make SVG shapes in python. Examples of lines, circle, rectangle, and path.

如何在 Python 绘制 SVG 图形。以下是一些绘制线形图，圆形图，矩形图，和路径图的例子。

刚开始接触 Plotly？

New to Plotly?

Plotly's Python library is free and open source! Get started by downloading the client and reading the primer.

You can set up Plotly to work in online or offline mode, or in jupyter notebooks.

We also have a quick-reference cheatsheet (new!) to help you get started!!

Plotly 的 Python 库是免费并且开源的。Get started by downloading the client and reading the primer。

你可以设置 Plotly 以 online 或者 offline 的方式运行，也可以在 jupyter notebooks 中运行。

我们同样也有一个快速参考的 cheatsheet (new!)来帮助新手学习。

相对于轴的垂直线和水平线

Vertical and Horizontal Lines Positioned Relative to the Axes

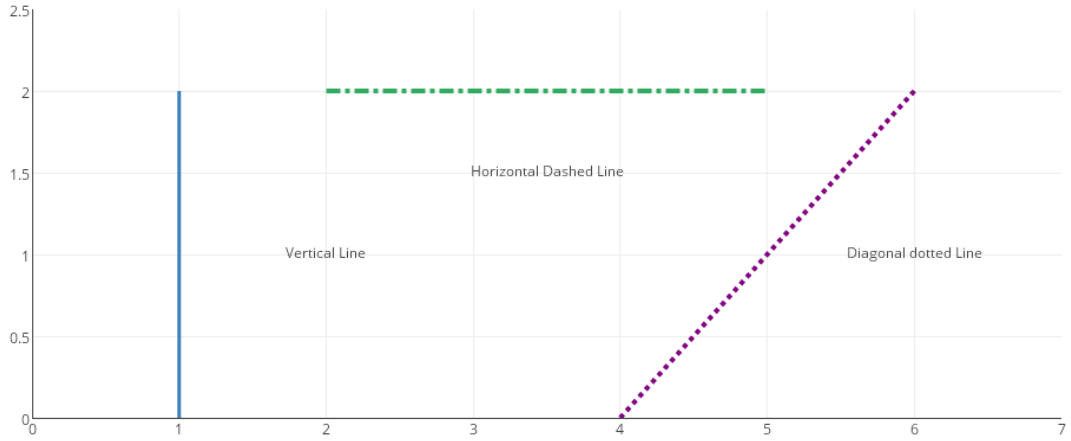
In [1]:

```
import plotly.plotly as py
import plotly.graph_objs as go

trace0 = go.Scatter(
    x=[2, 3.5, 6],
    y=[1, 1.5, 1],
    text=['Vertical Line', 'Horizontal Dashed Line', 'Diagonal dotted Line'],
```

```
    mode='text',
)
data = [trace0]
layout = {
    'xaxis': {
        'range': [0, 7]
    },
    'yaxis': {
        'range': [0, 2.5]
    },
    'shapes': [
        # Line Vertical
        {
            'type': 'line',
            'x0': 1,
            'y0': 0,
            'x1': 1,
            'y1': 2,
            'line': {
                'color': 'rgb(55, 128, 191)',
                'width': 3,
            },
        },
        # Line Horizontal
        {
            'type': 'line',
            'x0': 2,
            'y0': 2,
            'x1': 5,
            'y1': 2,
            'line': {
                'color': 'rgb(50, 171, 96)',
                'width': 4,
                'dash': 'dashdot',
            },
        },
        # Line Diagonal
        {
            'type': 'line',
            'x0': 4,
```

```
            'y0': 0,
            'x1': 6,
            'y1': 2,
            'line': {
                'color': 'rgb(128, 0, 128)',
                'width': 4,
                'dash': 'dot',
            },
        },
    ]
}

fig = {
    'data': data,
    'layout': layout,
}

    py.iplot(fig, filename='shapes-lines')
```

Out[1]:



相对于图和轴的作线

Lines Positioned Relative to the Plot & to the Axes

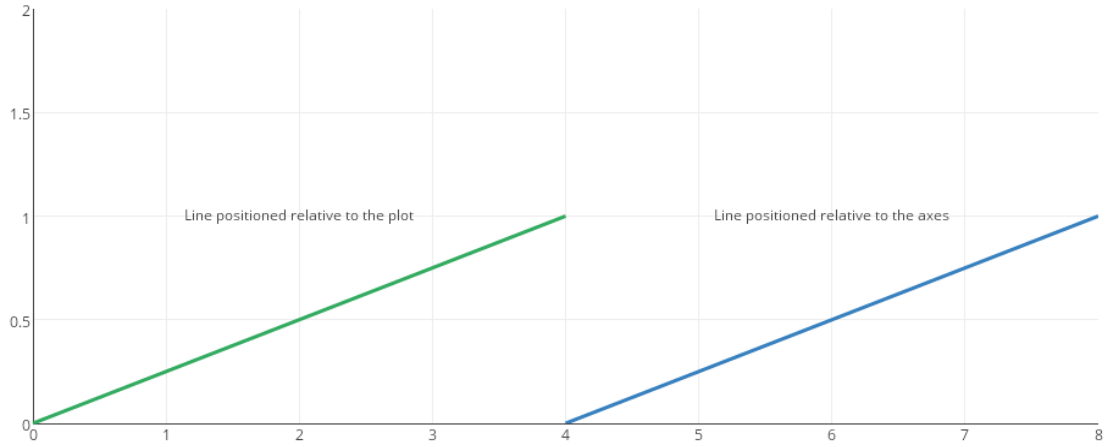
In [2]:

```
import plotly.plotly as py
import plotly.graph_objs as go
```

```
trace0 = go.Scatter(
    x=[2, 6],
    y=[1, 1],
    text=['Line positioned relative to the plot',
          'Line positioned relative to the axes'],
    mode='text',
)
data = [trace0]
layout = {
    'xaxis': {
        'range': [0, 8]
    },
    'yaxis': {
        'range': [0, 2]
    },
    'shapes': [
        # Line reference to the axes
        {
            'type': 'line',
            'xref': 'x',
            'yref': 'y',
            'x0': 4,
            'y0': 0,
            'x1': 8,
            'y1': 1,
            'line': {
                'color': 'rgb(55, 128, 191)',
                'width': 3,
            },
        },
        # Line reference to the plot
        {
            'type': 'line',
            'xref': 'paper',
            'yref': 'paper',
            'x0': 0,
            'y0': 0,
            'x1': 0.5,
            'y1': 0.5,
            'line': {
```

```
            'color': 'rgb(50, 171, 96)',
            'width': 3,
        },
    },
  ]
}
fig = {
  'data': data,
  'layout': layout,
}
py.iplot(fig, filename='shapes-line-ref')
```

Out[2]:



创建图形的切线

Creating Tangent Lines with Shapes

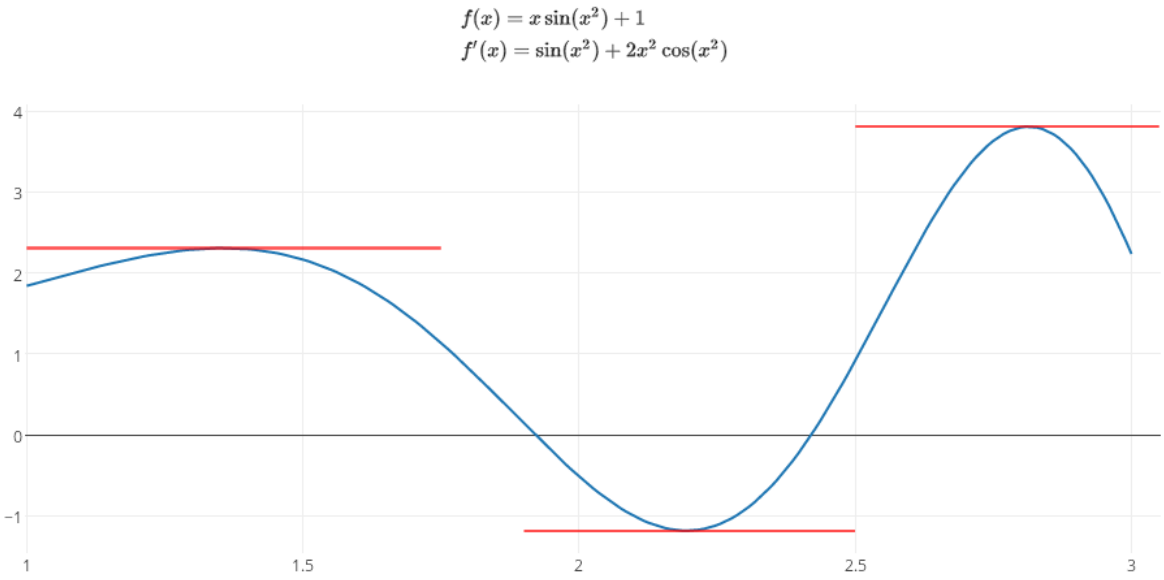
In [3]:

```
import plotly.plotly as py
import plotly.graph_objs as go

import numpy as np

x0 = np.linspace(1, 3, 200)
y0 = x0 * np.sin(np.power(x0, 2)) + 1

trace0 = go.Scatter(
```

```
    x=x0,
    y=y0,
)
data = [trace0]
layout = {
    'title': "$f(x)=x\\sin(x^2)+1\\\\ f\'(x)=\\sin(x^2)+2x^2\\cos(x^2)$",
    'shapes': [
        {
            'type': 'line',
            'x0': 1,
            'y0': 2.30756,
            'x1': 1.75,
            'y1': 2.30756,
            'opacity': 0.7,
            'line': {
                'color': 'red',
                'width': 2.5,
            },
        },
        {
            'type': 'line',
            'x0': 2.5,
            'y0': 3.80796,
            'x1': 3.05,
            'y1': 3.80796,
            'opacity': 0.7,
            'line': {
                'color': 'red',
                'width': 2.5,
            },
        },
        {
            'type': 'line',
            'x0': 1.90,
            'y0': -1.1827,
            'x1': 2.50,
            'y1': -1.1827,
            'opacity': 0.7,
            'line': {
                'color': 'red',
```

```
            'width': 2.5,
          },
        },
    ]
}
fig = {
    'data': data,
    'layout': layout,
}
py.iplot(fig, filename='tangent-line')
```

Out[3]:



相对于轴的矩形

Rectangles Positioned Relative to the Axes

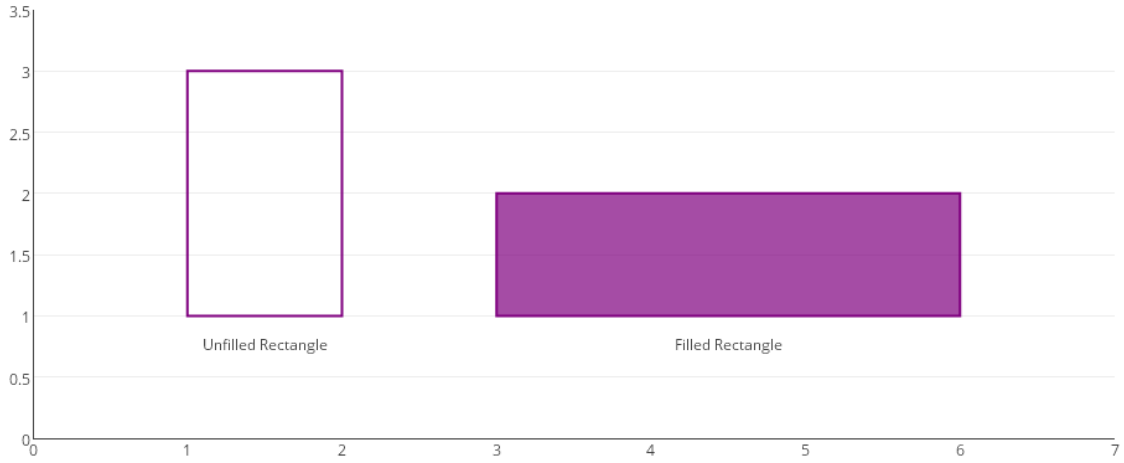
In [4]:

```
import plotly.plotly as py
import plotly.graph_objs as go

trace0 = go.Scatter(
    x=[1.5, 4.5],
    y=[0.75, 0.75],
    text=['Unfilled Rectangle', 'Filled Rectangle'],
    mode='text',
)
```

```
data = [trace0]
layout = {
    'xaxis': {
        'range': [0, 7],
        'showgrid': False,
    },
    'yaxis': {
        'range': [0, 3.5]
    },
    'shapes': [
        # unfilled Rectangle
        {
            'type': 'rect',
            'x0': 1,
            'y0': 1,
            'x1': 2,
            'y1': 3,
            'line': {
                'color': 'rgba(128, 0, 128, 1)',
            },
        },
        # filled Rectangle
        {
            'type': 'rect',
            'x0': 3,
            'y0': 1,
            'x1': 6,
            'y1': 2,
            'line': {
                'color': 'rgba(128, 0, 128, 1)',
                'width': 2,
            },
            'fillcolor': 'rgba(128, 0, 128, 0.7)',
        },
    ]
}
fig = {
    'data': data,
    'layout': layout,
}
```

```
py.iplot(fig, filename='shapes-rectangle')
```

Out[4]:



相对于图和轴线的矩形

Rectangle Positioned Relative to the Plot & to the Axes

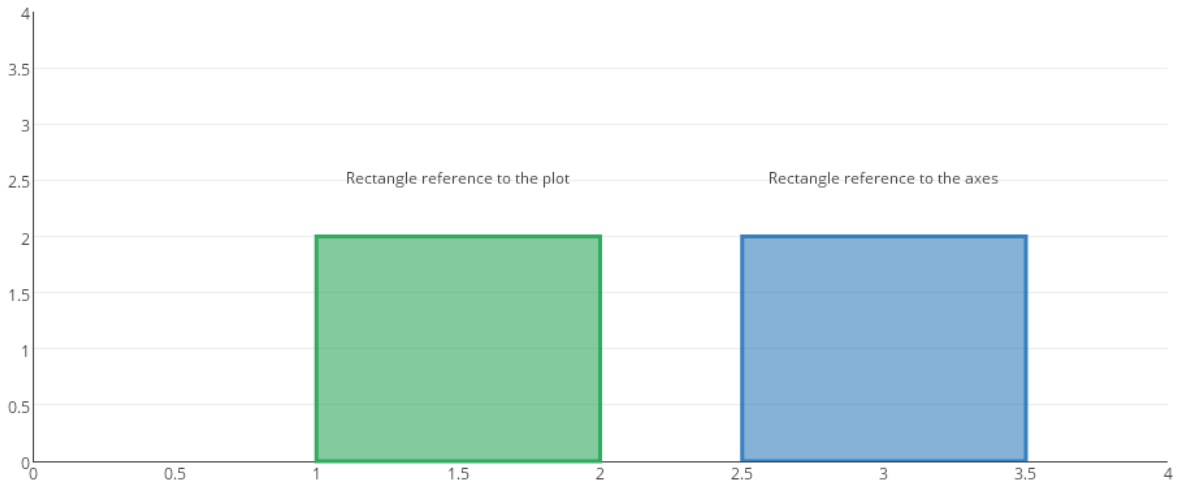
In [5]:

```
import plotly.plotly as py
import plotly.graph_objs as go

trace0 = go.Scatter(
    x=[1.5, 3],
    y=[2.5, 2.5],
    text=['Rectangle reference to the plot',
          'Rectangle reference to the axes'],
    mode='text',
)
data = [trace0]
layout = {
    'xaxis': {
        'range': [0, 4],
        'showgrid': False,
    },
    'yaxis': {
        'range': [0, 4]
```

```
    },
    'shapes': [
        # Rectangle reference to the axes
        {
            'type': 'rect',
            'xref': 'x',
            'yref': 'y',
            'x0': 2.5,
            'y0': 0,
            'x1': 3.5,
            'y1': 2,
            'line': {
                'color': 'rgb(55, 128, 191)',
                'width': 3,
            },
            'fillcolor': 'rgba(55, 128, 191, 0.6)',
        },
        # Rectangle reference to the plot
        {
            'type': 'rect',
            'xref': 'paper',
            'yref': 'paper',
            'x0': 0.25,
            'y0': 0,
            'x1': 0.5,
            'y1': 0.5,
            'line': {
                'color': 'rgb(50, 171, 96)',
                'width': 3,
            },
            'fillcolor': 'rgba(50, 171, 96, 0.6)',
        },
    ]
}
fig = {
    'data': data,
```

```
    'layout': layout,
}
py.iplot(fig, filename='shapes-rectangle-ref')
```

Out[5]:



用矩形对时间序列区域进行高亮显示

Reference Highlighting Time Series Regions with Rectangle Shapes

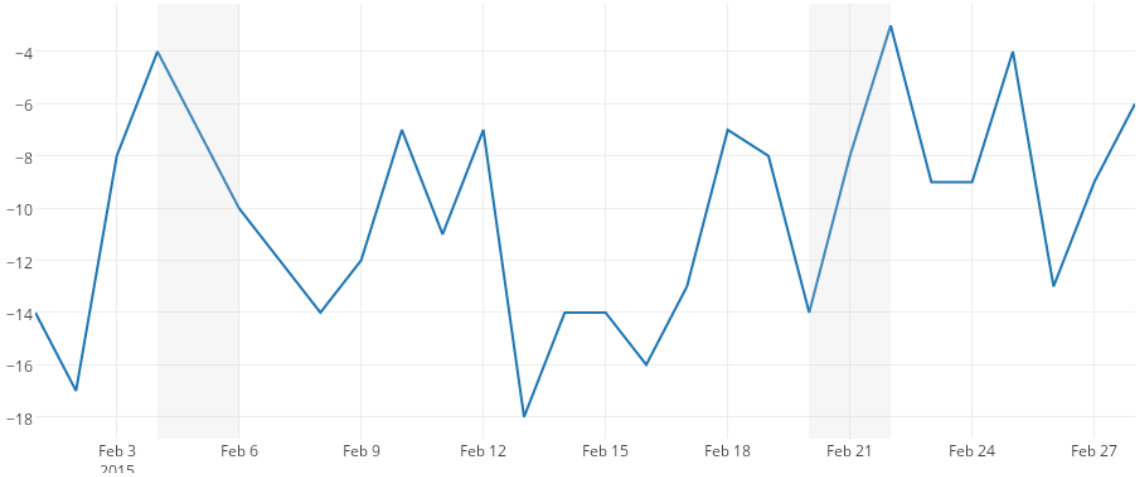
In [6]:

```
import plotly.plotly as py
import plotly.graph_objs as go

trace0 = go.Scatter(
    x=['2015-02-01', '2015-02-02', '2015-02-03', '2015-02-04', '2015-02-05',
       '2015-02-06', '2015-02-07', '2015-02-08', '2015-02-09', '2015-02-10',
       '2015-02-11', '2015-02-12', '2015-02-13', '2015-02-14', '2015-02-15',
       '2015-02-16', '2015-02-17', '2015-02-18', '2015-02-19', '2015-02-20',
       '2015-02-21', '2015-02-22', '2015-02-23', '2015-02-24', '2015-02-25',
       '2015-02-26', '2015-02-27', '2015-02-28'],
    y=[-14, -17, -8, -4, -7, -10, -12, -14, -12, -7, -11, -7, -18, -14, -14,
       -16, -13, -7, -8, -14, -8, -3, -9, -9, -4, -13, -9, -6],
    mode='line',
    name='temperature'
)
```

```
data = [trace0]
layout = {
    # to highlight the timestamp we use shapes and create a rectangular
    'shapes': [
        # 1st highlight during Feb 4 - Feb 6
        {
            'type': 'rect',
            # x-reference is assigned to the x-values
            'xref': 'x',
            # y-reference is assigned to the plot paper [0,1]
            'yref': 'paper',
            'x0': '2015-02-04',
            'y0': 0,
            'x1': '2015-02-06',
            'y1': 1,
            'fillcolor': '#d3d3d3',
            'opacity': 0.2,
            'line': {
                'width': 0,
            }
        },
        # 2nd highlight during Feb 20 - Feb 23
        {
            'type': 'rect',
            'xref': 'x',
            'yref': 'paper',
            'x0': '2015-02-20',
            'y0': 0,
            'x1': '2015-02-22',
            'y1': 1,
            'fillcolor': '#d3d3d3',
            'opacity': 0.2,
            'line': {
                'width': 0,
            }
        }
```

```
    ]
}
py.iplot({'data': data, 'layout': layout}, filename='timestamp-highlight')
```

Out[6]:



相对于轴线的圆

Reference Circles Positioned Relative to the Axes

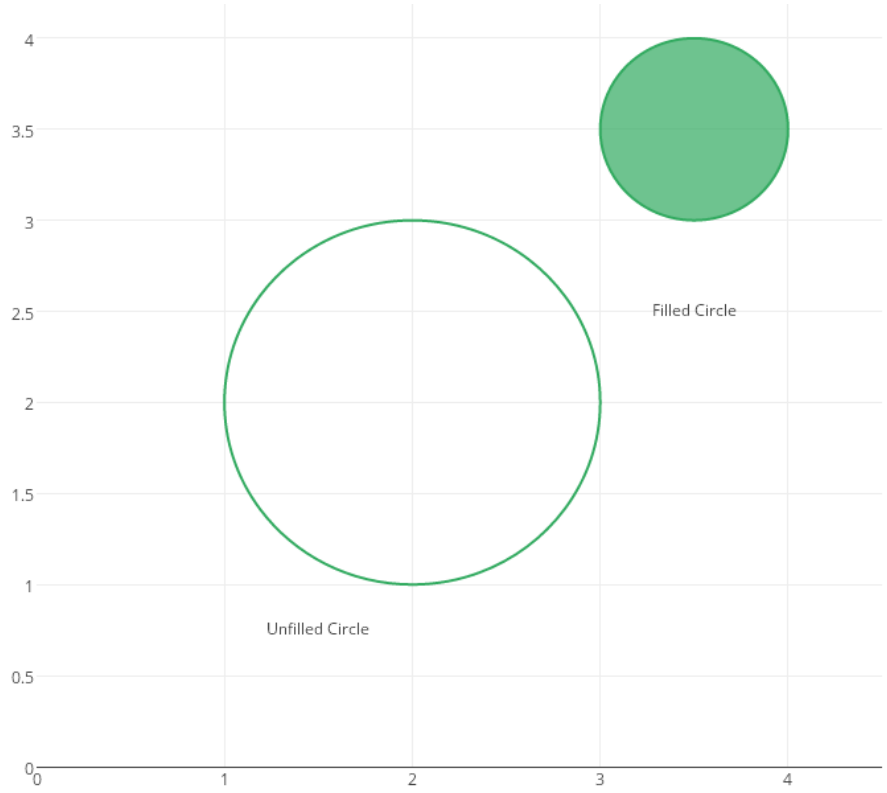
In [7]:

```
import plotly.plotly as py
import plotly.graph_objs as go

trace0 = go.Scatter(
    x=[1.5, 3.5],
    y=[0.75, 2.5],
    text=['Unfilled Circle',
          'Filled Circle'],
    mode='text',
)
data = [trace0]

layout = {
    'xaxis': {
        'range': [0, 4.5],
        'zeroline': False,
    },
    'yaxis': {
```

```
        'range': [0, 4.5]
    },
    'width': 800,
    'height': 800,
    'shapes': [
        # unfilled circle
        {
            'type': 'circle',
            'xref': 'x',
            'yref': 'y',
            'x0': 1,
            'y0': 1,
            'x1': 3,
            'y1': 3,
            'line': {
                'color': 'rgba(50, 171, 96, 1)',
            },
        },
        # filled circle
        {
            'type': 'circle',
            'xref': 'x',
            'yref': 'y',
            'fillcolor': 'rgba(50, 171, 96, 0.7)',
            'x0': 3,
            'y0': 3,
            'x1': 4,
            'y1': 4,
            'line': {
                'color': 'rgba(50, 171, 96, 1)',
            },
        },
    ]
}
fig = {
    'data': data,
```

```
    'layout': layout,
}
py.iplot(fig, filename='shapes-circle')
```

Out[7]:



用圆对散点图的聚集簇进行高亮显示

Reference Highlighting Clusters of Scatter Points with Circle Shapes

In [8]:

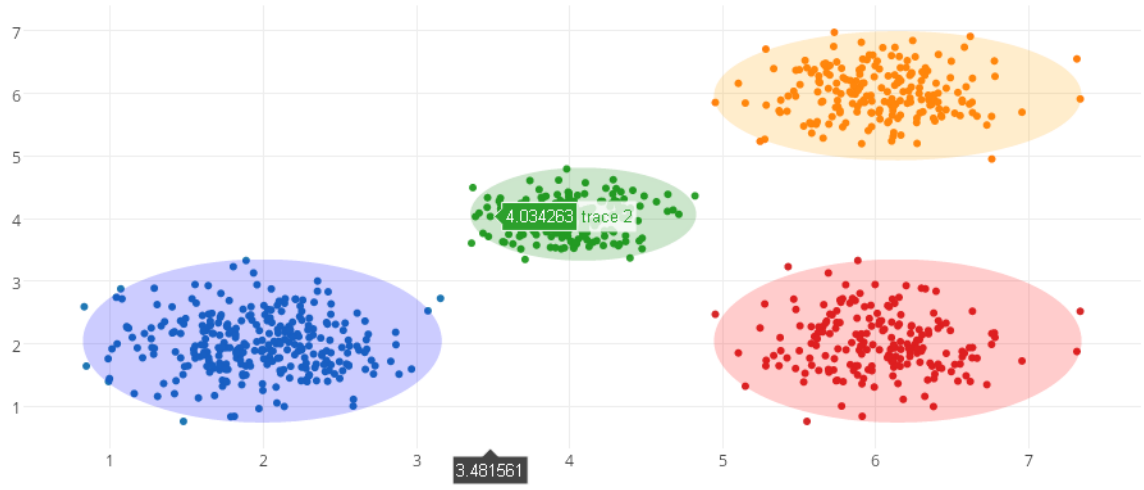
```
import plotly.plotly as py
import plotly.graph_objs as go

import numpy as np

x0 = np.random.normal(2, 0.45, 300)
y0 = np.random.normal(2, 0.45, 300)

x1 = np.random.normal(6, 0.4, 200)
y1 = np.random.normal(6, 0.4, 200)

x2 = np.random.normal(4, 0.3, 200)
```

```
y2 = np.random.normal(4, 0.3, 200)

trace0 = go.Scatter(
    x=x0,
    y=y0,
    mode='markers',
)
trace1 = go.Scatter(
    x=x1,
    y=y1,
    mode='markers'
)
trace2 = go.Scatter(
    x=x2,
    y=y2,
    mode='markers'
)
trace3 = go.Scatter(
    x=x1,
    y=y0,
    mode='markers'
)
layout = {
    'shapes': [
        {
            'type': 'circle',
            'xref': 'x',
            'yref': 'y',
            'x0': min(x0),
            'y0': min(y0),
            'x1': max(x0),
            'y1': max(y0),
            'opacity': 0.2,
            'fillcolor': 'blue',
            'line': {
                'color': 'blue',
```

```
        },
    },
    {
        'type': 'circle',
        'xref': 'x',
        'yref': 'y',
        'x0': min(x1),
        'y0': min(y1),
        'x1': max(x1),
        'y1': max(y1),
        'opacity': 0.2,
        'fillcolor': 'orange',
        'line': {
            'color': 'orange',
        },
    },
    {
        'type': 'circle',
        'xref': 'x',
        'yref': 'y',
        'x0': min(x2),
        'y0': min(y2),
        'x1': max(x2),
        'y1': max(y2),
        'opacity': 0.2,
        'fillcolor': 'green',
        'line': {
            'color': 'green',
        },
    },
    {
        'type': 'circle',
        'xref': 'x',
        'yref': 'y',
        'x0': min(x1),
        'y0': min(y0),
```

```
            'x1': max(x1),
            'y1': max(y0),
            'opacity': 0.2,
            'fillcolor': 'red',
            'line': {
                'color': 'red',
            },
        },
    ],
    'showlegend': False,
}
data = [trace0, trace1, trace2, trace3]
fig = {
    'data': data,
    'layout': layout,
}
py.iplot(fig, filename='clusters')
```

Out[8]:



使用圆绘制文氏图

Reference Venn Diagram with Circle Shapes

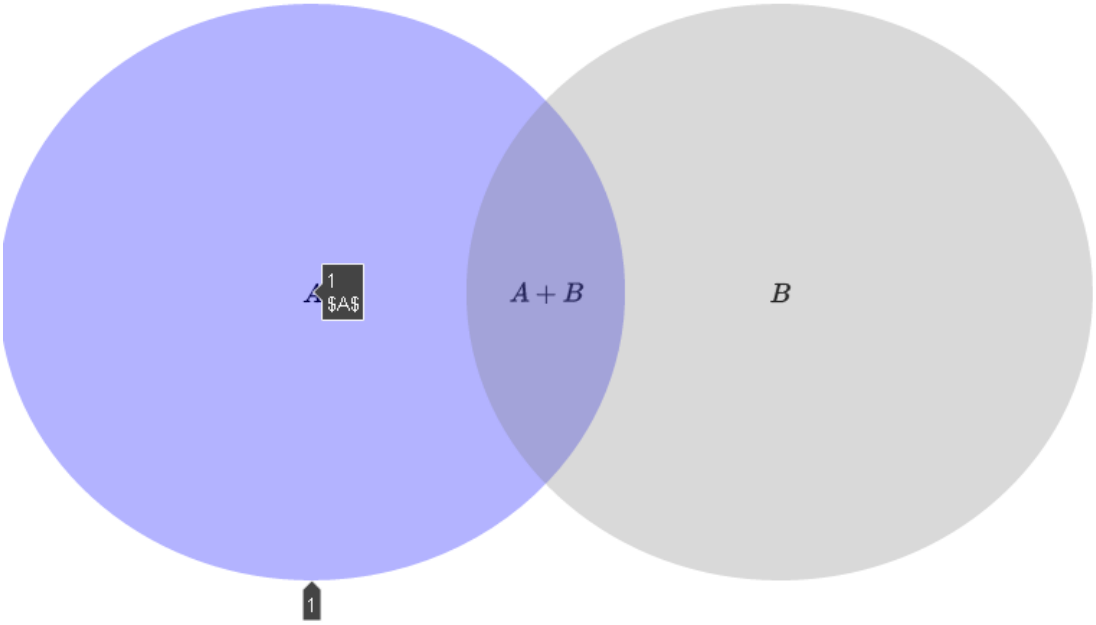
In [9]:

```
import plotly.plotly as py
```

```
import plotly.graph_objs as go

trace0 = go.Scatter(
    x=[1, 1.75, 2.5],
    y=[1, 1, 1],
    text=['$A$', '$A+B$', '$B$'],
    mode='text',
    textfont=dict(
        color='black',
        size=18,
        family='Arail',
    )
)

data = [trace0]

layout = {
    'xaxis': {
        'showticklabels': False,
        'autotick': False,
        'showgrid': False,
        'zeroline': False,
    },
    'yaxis': {
        'showticklabels': False,
        'autotick': False,
        'showgrid': False,
        'zeroline': False,
    },
    'shapes': [
        {
            'opacity': 0.3,
            'xref': 'x',
            'yref': 'y',
            'fillcolor': 'blue',
            'x0': 0,
```

```
            'y0': 0,
            'x1': 2,
            'y1': 2,
            'type': 'circle',
            'line': {
                'color': 'blue',
            },
        },
        {
            'opacity': 0.3,
            'xref': 'x',
            'yref': 'y',
            'fillcolor': 'gray',
            'x0': 1.5,
            'y0': 0,
            'x1': 3.5,
            'y1': 2,
            'type': 'circle',
            'line': {
                'color': 'gray',
            },
        }
    ],
    'margin': {
        'l': 20,
        'r': 20,
        'b': 100
    },
    'height': 600,
    'width': 800,
}
fig = {
    'data': data,
    'layout': layout,
}
py.iplot(fig, filename='venn-diagram')
```

Out[9]:



SVG 线径

Reference SVG Paths

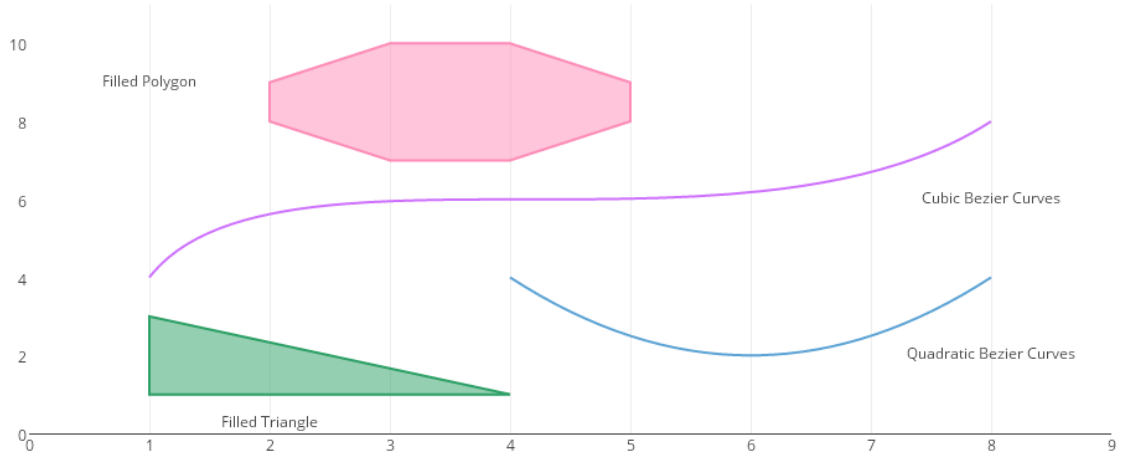
In [10]:

```
import plotly.plotly as py
import plotly.graph_objs as go

trace0 = go.Scatter(
    x=[2, 1, 8, 8],
    y=[0.25, 9, 2, 6],
    text=['Filled Triangle',
          'Filled Polygon',
          'Quadratic Bezier Curves',
          'Cubic Bezier Curves'],
    mode='text',
)
data = [trace0]
layout = {
```

```
    'xaxis': {
        'range': [0, 9],
        'zeroline': False,
    },
    'yaxis': {
        'range': [0, 11],
        'showgrid': False,
    },
    'shapes': [
        # Quadratic Bezier Curves
        {
            'type': 'path',
            'path': 'M 4,4 Q 6,0 8,4',
            'line': {
                'color': 'rgb(93, 164, 214,)',
            },
        },
        # Cubic Bezier Curves
        {
            'type': 'path',
            'path': 'M 1,4 C 2,8 6,4 8,8',
            'line': {
                'color': 'rgb(207, 114, 255)',
            },
        },
        # filled Triangle
        {
            'type': 'path',
            'path': ' M 1 1 L 1 3 L 4 1 Z',
            'fillcolor': 'rgba(44, 160, 101, 0.5)',
            'line': {
                'color': 'rgb(44, 160, 101)',
            },
        },
        # filled Polygon
        {
```

```
            'type': 'path',
            'path': ' M 3,7 L2,8 L2,9 L3,10, L4,10 L5,9 L5,8 L4,7 Z',
            'fillcolor': 'rgba(255, 140, 184, 0.5)',
            'line': {
                'color': 'rgb(255, 140, 184)',
            },
        },

    ]
}
fig = {
    'data': data,
    'layout': layout,
}
py.iplot(fig, filename='shapes-path')
```

Out[10]:



参考

Reference

See https://plot.ly/python/reference/#layout-shapes for more information and chart attribute options!

查看 https://plot.ly/python/reference/#layout-shapes，获取更多的信息与图表属性的选项！

输出静态图形

Static Image Export in Python

Plotly allows you to save static images of your plots. Save the image to your local computer, or embed it inside your Jupyter notebooks as a static image.

Plotly 允许你保存自己的静态图形。保存在你本地的计算机，或者作为静态图形嵌入到 Jupyter notebooks 中。

刚开始接触 Plotly？

New to Plotly?

Plotly's Python library is free and open source! Get started by downloading the client and reading the primer.

You can set up Plotly to work in online or offline mode, or in jupyter notebooks.

We also have a quick-reference cheatsheet (new!) to help you get started!!

Plotly 的 Python 库是免费并且开源的。Get started by downloading the client and reading the primer。

你可以设置 Plotly 以 online 或者 offline 的方式运行，也可以在 jupyter notebooks 中运行。

我们同样也有一个快速参考的 cheatsheet (new!)来帮助新手学习。

在线输出静态图形

Export Static Images Online

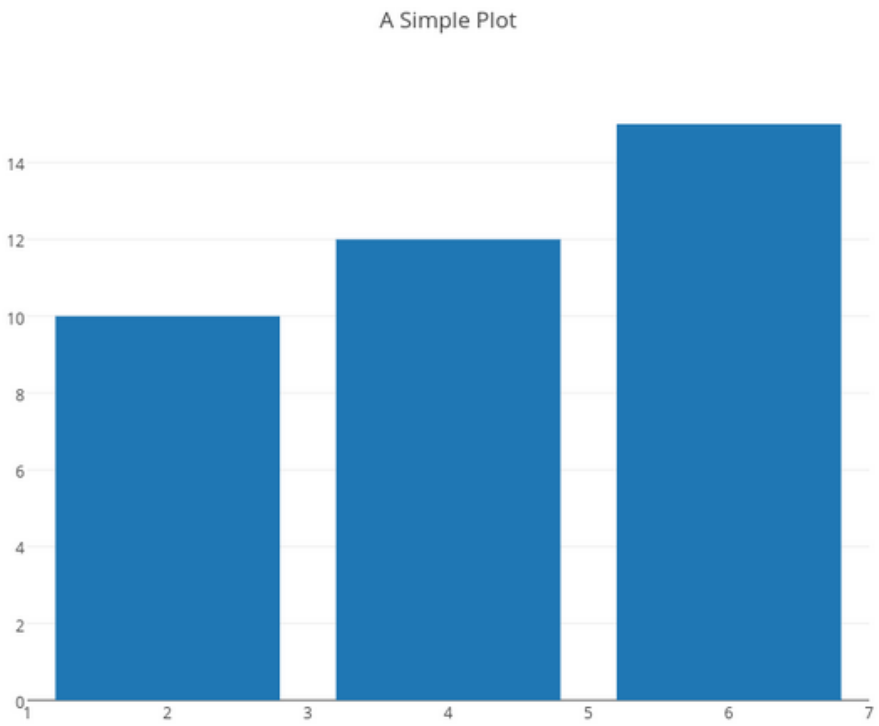
To save the image, you need to login to plotly using your credentials (username and API Key).

你需要在 plotly 上注册你的凭证（用户名和 API 码）才可以保存静态图形

In [1]:

```
import plotly.plotly as py
import plotly.graph_objs as go

py.sign_in('DemoAccount', '2qdyfjyr7o') # Replace the username, and API key with your credentials.

trace = go.Bar(x=[2, 4, 6], y= [10, 12, 15])
data = [trace]
layout = go.Layout(title='A Simple Plot', width=800, height=640)
fig = go.Figure(data=data, layout=layout)

py.image.save_as(fig, filename='a-simple-plot.png')

from IPython.display import Image
Image('a-simple-plot.png')
```

Out[1]:



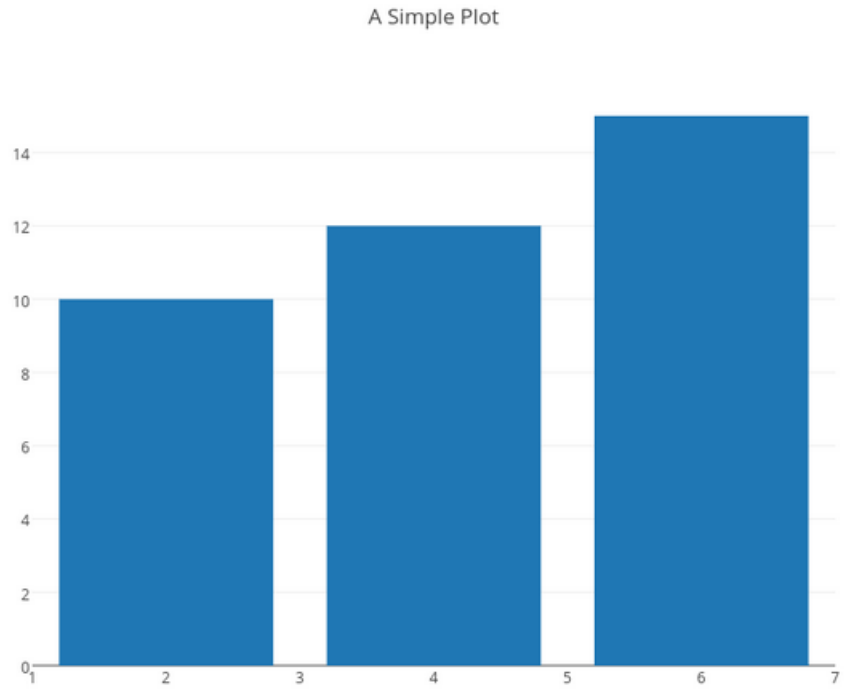
把图形嵌入到 Jupyter Notebooks

Embed Static Images in Jupyter Notebooks

In [2]:

```
py.image.ishow(fig)
```

Out[2]:



从在线图表中获取图形

Retrieve an Image from an Existing Online Chart

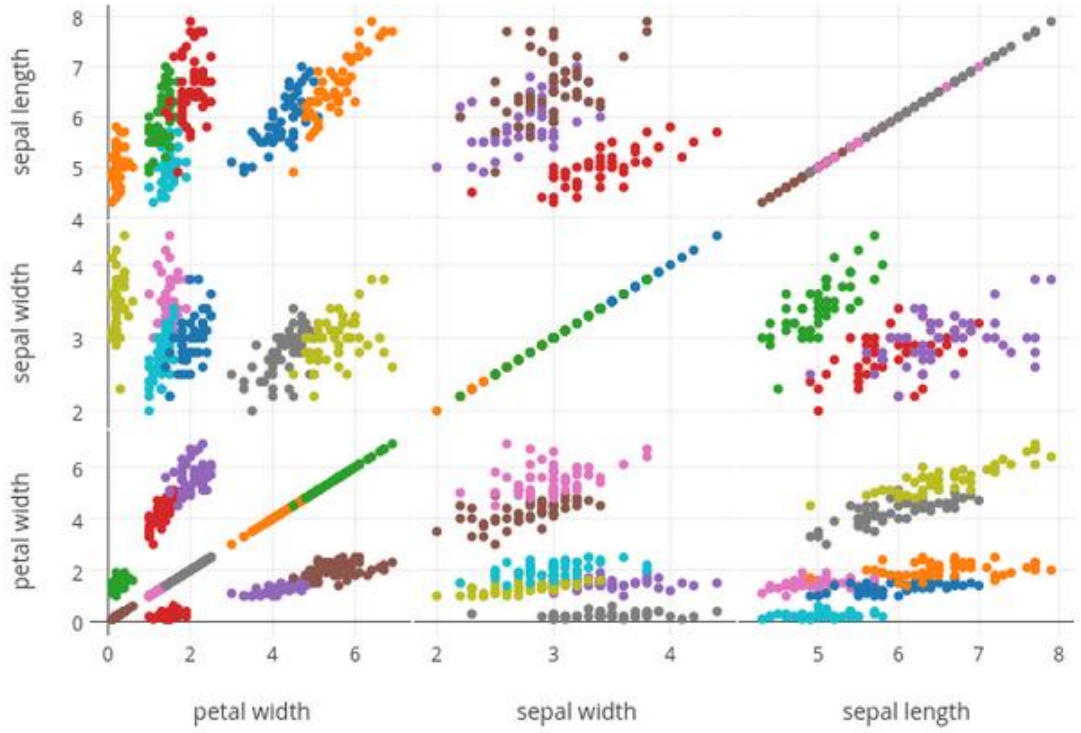
To export an image of a chart you (or someone else) have already created, first you can retreive it using get_figure method, and then save it.

想要输出你（或者别人）已经创建的图表中的某个图形，首先你可以使用 get_figure 方法获取它，然后再保存

In [3]:

```
fig = py.get_figure('chris', '1638')
py.image.save_as(fig,'chris-plot.png')

Image('chris-plot.png') # Display a static image
```

Out[3]:



支持的格式

Supported Formats

The common image formats: 'PNG', 'JPG/JPEG' are supported. In addition, formats like 'EPS', 'SVG' and 'PDF' are also supported.

普通的图像格式：支持'PNG', 'JPG/JPEG'。并且像'EPS', 'SVG' 和'PDF'一类的格式同样支持。

Note: The SVG, EPS and PDF Formats are *only* available for Plotly Professional users. You can get more details on our pricing page

提示：SVG, EPS 和 PDF 只能在 Plotly 专业版中购买使用。更多信息请查询我们的价格页。

To access the image in a particular format, you can either:

获取特殊格式的图像，你可以采取以下方法：

• append the format extension to the plot url. i.e. the JPG version of the plot:https://plot.ly/~chris/1638 is available at : https://plot.ly/~chris/1638.jpg
把扩展格式添加到 plot 地址，例如,plot 的 JPG 版本地址：https://plot.ly/~chris/1638 可在 https://plot.ly/~chris/1638.jpg 中获取
• Add the appropriate extension to the save_as method: py.image.save_as(fig, 'chris-plot.jpg')
• 添加合适的扩展到存档可以使用方法： py.image.save_as(fig, 'chris-plot.jpg')

在保存图像数据到内存

Saving Image Data in Memory

The data from the charts can also be stored in the memory. This reduces the overhead of saving the image in the hard disk. It is particularly useful when the image needs to be embedded (for example in email reports).

图表中的数据同样可以被保存在内存中，这 可以减少在硬盘中保存图像的系统开销。当图像需要被嵌入的时候这将特别有用。

In [4]:

```
import requests
image_bytes = requests.get('https://plot.ly/~chris/1638.jpg').content
```

离线输出静态图像

Export Static Image Offline

We can use the same iplot and plot functions as before, but if we want to download these same images, then we'll have to include an additional argument called image. image will define the format of the image file that is to be downloaded.

像之前一样，我们可以使用一些 iplot 和 plot 函数，但是如果我们要下载这些相同的图像，那么我们必须包括一个额外的叫做 image 的参数。Image 定义将被下载的图像格式文件

In [5]:

```
import plotly.offline as offline import plotly.graph_objs as go offline.init_notebook_mode()
offline.iplot({'data': [{'y': [4, 2, 3, 4]}], 'layout': {'title': 'Test Plot', 'font': dict(size=16)}},
image='png')
import plotly.offline as offline
import plotly.graph_objs as go
offline.plot({'data': [{'y': [4, 2, 3, 4]}], 'layout': {'title': 'Test Plot', 'font': dict(size=16)}},
image='png')
```

Note that you can also define the height, width and filename of the image. See help(offline.iplot) or help(offline.plot) for more information.

注意你也可以定义图像的高度，宽度和文件名。更多信息请看帮助(offline.iplot) 或帮助(offline.plot)

参考

Reference

In [6]:

```
help(py.image)
Help on class image in module plotly.plotly.plotly:

class image
 | Helper functions wrapped around plotly's static image generation api.
 |
 | Class methods defined here:
 |
 | ishow(cls, figure_or_data, format='png', width=None, height=None, scale=None) from
__builtin__.classobj
 |     Display a static image of the plot described by `figure_or_data`
 |     in an IPython Notebook.
 |
 |     positional arguments:
 |     - figure_or_data: The figure dict-like or data list-like object that
 |                       describes a plotly figure.
 |                       Same argument used in `py.plot`, `py.iplot`,
 |                       see https://plot.ly/python for examples
 |     - format: 'png', 'svg', 'jpeg', 'pdf'
 |     - width: output width
 |     - height: output height
 |     - scale: Increase the resolution of the image by `scale` amount
 |          Only valid for PNG and JPEG images.
 |
 |     example:
 |     ```
 |     import plotly.plotly as py
 |     fig = {'data': [{'x': [1, 2, 3], 'y': [3, 1, 5], 'type': 'bar'}]}
 |     py.image.ishow(fig, 'png', scale=3)
 |
```

```
 |  save_as(cls, figure_or_data, filename, format=None, width=None, height=None, scale=None)
from __builtin__.classobj
 |      Save a image of the plot described by `figure_or_data` locally as
 |      `filename`.
 |
 |      Valid image formats are 'png', 'svg', 'jpeg', and 'pdf'.
 |      The format is taken as the extension of the filename or as the
 |      supplied format.
 |
 |      positional arguments:
 |      - figure_or_data: The figure dict-like or data list-like object that
 |                        describes a plotly figure.
 |                        Same argument used in `py.plot`, `py.iplot`,
 |                        see https://plot.ly/python for examples
 |      - filename: The filepath to save the image to
 |      - format: 'png', 'svg', 'jpeg', 'pdf'
 |      - width: output width
 |      - height: output height
 |      - scale: Increase the resolution of the image by `scale` amount
 |            Only valid for PNG and JPEG images.
 |
 |      example:
 |      ```
 |      import plotly.plotly as py
 |      fig = {'data': [{'x': [1, 2, 3], 'y': [3, 1, 5], 'type': 'bar'}]}
 |      py.image.save_as(fig, 'my_image.png', scale=3)
 |      ```
 |
 | ----------------------------------------------------------------------
 | Static methods defined here:
 |
 | get(figure_or_data, format='png', width=None, height=None, scale=None)
 |      Return a static image of the plot described by `figure_or_data`.
 |
 |      positional arguments:
 |      - figure_or_data: The figure dict-like or data list-like object that
 |                        describes a plotly figure.
 |                        Same argument used in `py.plot`, `py.iplot`,
 |                        see https://plot.ly/python for examples
 |      - format: 'png', 'svg', 'jpeg', 'pdf'
 |      - width: output width
 |      - height: output height
 |      - scale: Increase the resolution of the image by `scale`
 |             amount (e.g. `3`)
 |             Only valid for PNG and JPEG images.
 |
 |      example:
 |      ```
 |      import plotly.plotly as py
 |      fig = {'data': [{'x': [1, 2, 3], 'y': [3, 1, 5], 'type': 'bar'}]}
 |      py.image.get(fig, 'png', scale=3)
In [7]:
help(offline.plot)
Help on function plot in module plotly.offline.offline:

plot(figure_or_data, show_link=True, link_text='Export to plot.ly', validate=True,
output_type='file', include_plotlyjs=True, filename='temp-plot.html', auto_open=True, image=None,
image_filename='plot_image', image_width=800, image_height=600)
    Create a plotly graph locally as an HTML document or string.
```

```
Example:
```
from plotly.offline import plot
import plotly.graph_objs as go

plot([go.Scatter(x=[1, 2, 3], y=[3, 2, 6])], filename='my-graph.html')
# We can also download an image of the plot by setting the image parameter
# to the image format we want
plot([go.Scatter(x=[1, 2, 3], y=[3, 2, 6])], filename='my-graph.html'
    image='jpeg')
```
More examples below.

figure_or_data -- a plotly.graph_objs.Figure or plotly.graph_objs.Data or
                  dict or list that describes a Plotly graph.
                  See https://plot.ly/python/ for examples of
                  graph descriptions.

Keyword arguments:
show_link (default=True) -- display a link in the bottom-right corner of
   of the chart that will export the chart to Plotly Cloud or
   Plotly Enterprise
link_text (default='Export to plot.ly') -- the text of export link
validate (default=True) -- validate that all of the keys in the figure
   are valid? omit if your version of plotly.js has become outdated
   with your version of graph_reference.json or if you need to include
   extra, unnecessary keys in your figure.
output_type ('file' | 'div' - default 'file') -- if 'file', then
   the graph is saved as a standalone HTML file and `plot`
   returns None.
   If 'div', then `plot` returns a string that just contains the
   HTML <div> that contains the graph and the script to generate the
   graph.
   Use 'file' if you want to save and view a single graph at a time
   in a standalone HTML file.
   Use 'div' if you are embedding these graphs in an HTML file with
   other graphs or HTML markup, like a HTML report or an website.
include_plotlyjs (default=True) -- If True, include the plotly.js
   source code in the output file or string.
   Set as False if your HTML file already contains a copy of the plotly.js
   library.
filename (default='temp-plot.html') -- The local filename to save the
   outputted chart to. If the filename already exists, it will be
   overwritten. This argument only applies if `output_type` is 'file'.
auto_open (default=True) -- If True, open the saved file in a
   web browser after saving.
   This argument only applies if `output_type` is 'file'.
image (default=None |'png' |'jpeg' |'svg' |'webp') -- This parameter sets
   the format of the image to be downloaded, if we choose to download an
   image. This parameter has a default value of None indicating that no
   image should be downloaded.
image_filename (default='plot_image') -- Sets the name of the file your image
   will be saved to. The extension should not be included.
image_height (default=600) -- Specifies the height of the image in `px`.
image_width (default=800) -- Specifies the width of the image in `px`.
```

IPython 与 Python 对比

Discussion of key differences between IPython and Python
讨论 IPython 和 Python 之间的核心区别

不同之处

What is the difference between IPython and Python?

While these two names are quite similar, they refer to entirely different things.

IPython 和 Python 有什么不同呢?

即使这两个名字十分相似，但它们几乎是完全不同的两样东西。

Python is a general-purpose programming language. It was created in the late 1980s by Guido van Rossum. It is now one of the most popular languages in the world. It is routinely used by system administrators and web developers. Also, many scientists are using Python thanks to libraries such as NumPy, SciPy, pandas, and matplotlib. The ease of use of Python and its dynamic nature make it a very productive language.

Python 是通用编程语言。 它是在 80 年代末由 Guido van Rossum 创造. 现在它是世界上非常流行的编程语言。它经常被系统管理员和网站开发人员使用。由于有 NumPy，SciPy，pandas 和 matplotlib 等科学计算的库，很多科学家也用 Python 了。 Python 的每个应用和它活跃的生态环境使得它成为非常有生产力的编程语言。

IPython is an interactive command-line terminal for Python. It was created by Fernando Perez in 2001. IPython offers an enhanced read-eval-print loop (REPL) environment particularly well adapted to scientific computing.

IPython 是一个 Python 的交互命令行终端。它是由 Fernando Perez 在

2001 年创造。 IPython 提供一个增强的“读取-求值-输出”循环(REPL) 环境，特别适合科学计算。

```
cyrille@gigabyte:~$ ipython
Python 3.4.3 |Anaconda 2.3.0 (64-bit)| (default, Jun  4 2015, 15:29:08)
Type "copyright", "credits" or "license" for more information.

IPython 3.2.0 -- An enhanced Interactive Python.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://anaconda.org
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: print("Hello world!")
Hello world!

In [2]: 2 * 3
Out[2]: 6

In [3]:
```

In other words, IPython is a powerful *interface* to the Python language. But it is certainly not the only one. Besides IPython, the most common way to use Python is to write *scripts*, files with the .py extension.

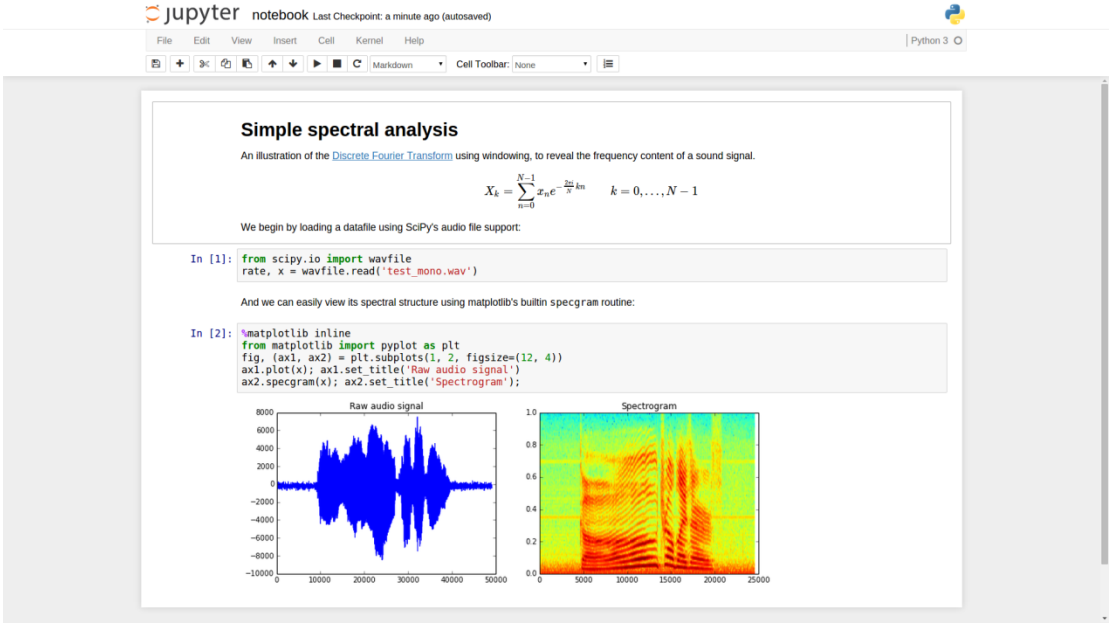
即是说，IPython 是一个 Python 编程语言的强大的*交互界面* 。但是它一定不是唯一的方式。除了 IPython, 另外使用 Python 的通用方法是写*脚本*，一种由.py 作后缀的文件。

A script contains a list of commands to execute in order. It runs from start to finish and display some output. On the contrary, with IPython, you generally write one command at a time and you get the results instantly. This is a completely different way of working with Python. When analyzing data or running computational models, you need this sort of interactivity to explore them efficiently.

一个脚本包含执行顺序的命令列表。它从开始运行到结束同时显示一些输出。相对地， 在 IPython 里你通常写一次一个命令，然后即时获得这个命令的结果。这是两种完全不同的方式使用 Python。 当你要分析数据或者模型计算时，你需要这种交互方式，来有效地显示结果。

Notebook

In 2011, IPython introduced a new tool named the **Notebook**. Inspired by scientific programs like Mathematica or Sage, the Notebook offers a modern and powerful web interface to Python.
在 2011 年， IPython 引入一个叫 **Notebook** 的新工具。 受到 Mathematica 或者 Sage 这些科学计算软件的启发 ，这个 Notebook 提供一个现代的强大的 Python 网络交互界面。



Compared to the original IPython terminal, the Notebook offers a more convenient text editor, the possibility to write rich text, and improved graphical capabilities. Also, since it is a web interface, it can integrate many of the existing web libraries for data visualization, including *plotly.js*.
对比原来 IPython 的终端，这个 Notebook 提供一个更方便的文本编辑器，可以写富文本格式，和改善图形处理能力。还因为它是一个网络接口，所以它可以为数据可视化，集成大量现成的网络库，包括 *plotly.js*.

In 2015, the IPython developers made a major code reorganization of their ever-growing project. The Notebook is now called the Jupyter Notebook. This interface can be used not only with Python but with dozens **of** other languages such as R and Julia. IPython is now the name of the Python backend (aka kernel).
在 2015 年，IPython 的开发者重新组织了他们不断增长的项目的主要代码。这个 Notebook 现在叫做 Jupyter Notebook.。这个交互界面不但可以使用 Python，还

可以用十几种其它语言，比如 R 语言和 Julia 语言。IPython 现在是 Python 后端(aka kernel)的名字。

In conclusion, IPython and Jupyter are great interfaces to the Python language. If you're learning Python, using the IPython terminal or the Jupyter Notebook is highly recommended.
总的来说，IPython 和 Jupyter 都是伟大的 Python 语言交互界面。如果你正学习 Python 语言，强烈推荐你用 IPython 终端或者 Jupyter Notebook。

This was a guest article written by Cyrille Rossant, author of Learning IPython for Interactive Computing and Data Visualization, second edition and IPython Interactive Computing and Visualization Cookbook.

这编介绍文章是由 Cyrille Rossant 编写, 他是《Learning IPython for Interactive Computing and Data Visualization》第二版和《IPython Interactive Computing and Visualization Cookbook》的作者。

20、Notebook 教程

IPython notebook tutorial on how to install, run, and use IPython for interactive matplotlib plotting, data analysis, and publishing code
IPython notebook 教程教你安装，运行，和使用 IPython 来和 matplotlib plotting 交互，数据分析，发布代码。

介绍

Introduction
IPython has a beautiful Notebook that lets you write and execute code, analyze data, embed content, and share reproducible work. IPython lets you easily share your code, data, plots, and explanation in one Notebook. Publishing is flexible: PDF, HTML, ipynb, dashboards, slides, and more. Code cells are based on an input and output format. For example:
IPython 有一个可以让你编写和执行代码，分析数据，插入内容，分享可重复的工作，的漂亮记事本（Notebook）。

IPython 让你很容易地分享你的代码，数据，和更多。代码单元是建立在输入输出格式的基础上。例如：

In [1]:
```
print "hello world"
```
```
hello world
```

There are a few ways to use an IPython Notebook. For Windows users, you'll need setuptools. This Notebook uses Python 2; Python 3 offers exciting new opportunities for IPython.

有几种使用 IPython Notebook 的方法。对于 Windows 用户，你需要 setuptools（安装工具）。这个 Notebook 用 Python2；Python3 提供令人兴奋的机会体验 IPython。

If you have setuptools or pip installed, you can open a terminal and type: $ pip install ipython.

如果你安装了 setuptools 或者 pip ，你可以打开终端输入：$ pip install ipython.

Anaconda and Enthought allow you to download a desktop version of IPython Notebook.

Anaconda 和 Enthought 允许你下载桌面版的 IPython Notebook。

coLaboratory allows you to run Notebooks using Google Chrome.

coLaboratory 允许你用 Google 的 Chrome 运行 Notebook。

Docker containers let you run Notebooks.

Docker 容器让你运行 Notebooks。

Domino, Authorea, and Wakari offer web-based Notebooks.

Domino, Authorea, 和 Wakari 提供基于网络的 Notebooks。

tmpnb launches a temporary online Notebook for individual users.

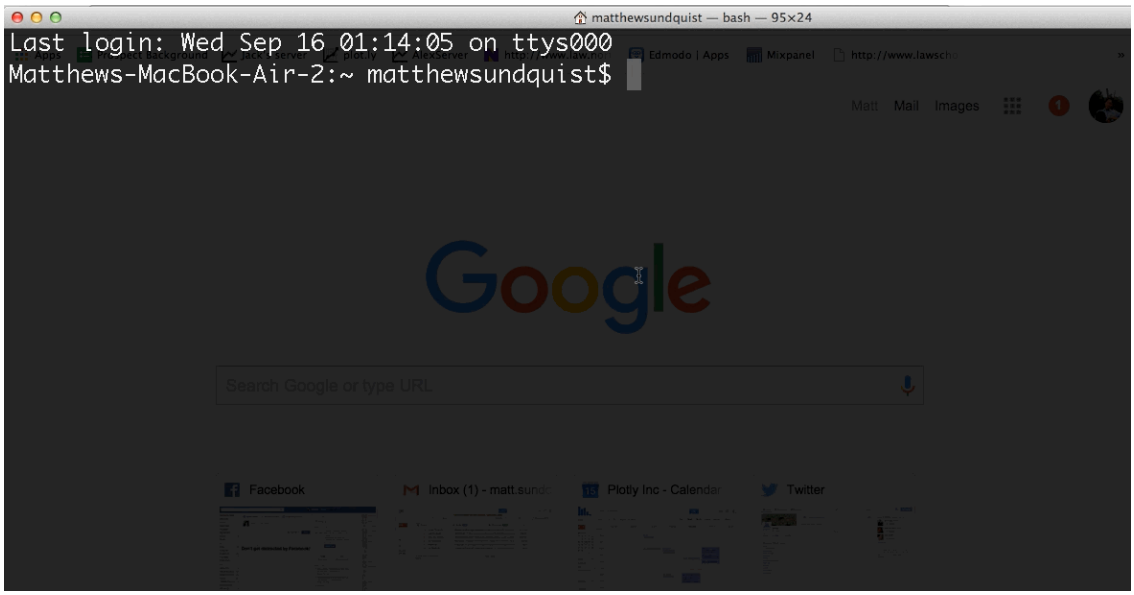
tmpnb 为单独的用户开启一个临时的在线 Notebooks。

Hosted services like Sciencebox let users launch a prebuilt virtual machine.

像 Sciencebox 的主机服务让用户开启一个预编译的虚拟机。

Once you've installed the Notebook, you start from your terminal by calling $ ipython notebook. This will open a browser on alocalhost to the URL of your Notebooks, by default http://127.0.0.1:8888. Windows users need to open up their Command Prompt. You'll see a dashboard with all your Notebooks. You can launch your Notebooks from there. The Notebook has the advantage of looking the same when you're coding and publishing. You just have all the options to move code, run cells, change kernels, and use Markdown when you're running a NB. Here's what a Notebook looks like in action if you call it from your terminal.

一旦你已经安装完 Notebooks，你可以从终端里调用 $ ipython notebook 开始。这样将打开一个浏览器，URL 是你的 Notebooks 的本地地址，默认是：http://127.0.0.1:8888。Windows 用户需要打开命令提示符。你将看到你所有的 Notebooks 的控制面板。你可以从这里开启你的 Notebooks。Notebooks 的优点是你编写的和你发布的东东看上去一样。当你运行一个 NB 时，你拥有移动代码，运行单元，更改内核，和用 Markdown 编写的权限。如果你在终端上运行 Notebooks，这是为什么 Notebook 看上去像在操作。



帮助命令

Helpful commands

IPython supports tab completion. You can type object_name.<TAB> to view an object’ s attributes. For tips on cell magics, running Notebooks, and exploring objects, check out the IPython docs. IPython also has a few helpful commands like:

IPython 支持用 tab 键补全。你可以键入 object_name.<TAB>来查看对象的属性。并于 cell 的魔法能力，运行 Notebooks，和浏览对象的建议，参看 IPython docs。IPython 也有几个有帮助的命令，如：

In [6]:

```
help # introduction and overview of features
```

Out[6]:

Type help() for interactive help, or help(object) for help about object.

In [7]:

```
%quickref # opens quick reference
```

The Notebook defaults to C:\Users\USERNAME or the folder where you've run the Notebook. You can call ipython notebook --help-all then change your directory if needed. You can also use %run to run local Python scripts you've written.

Notebook 默认目录是 C:\Users\USERNAME 或者你运行 Notebooks 的目录，如果你需要，你可以调用 ipython notebook --help-all 然后 change your directory 。 你也可以运行 %run 来执行你写的脚本。

IPython has keyboard shortcuts. Shift-Enter will run a cell, Ctrl-Enter will run a cell in-place, Alt-Enter will run a cell and insert another below.

IPython 有快捷键。 Shift-Enter 运行单元, Ctrl-Enter 就地执行单元, Alt-Enter 执行单元并插入某个单元下面。

包管理

Package management

When installing packages in IPython, you either need to install the package in your actual shell, or run the ! prefix, e.g.:

当要在 IPython 中安装软件包，你需要在你实际的命令行里安装软件包，或者带! 前缀执行，如：

!pip install packagename

You may want to reload submodules if you've edited the code in one. IPython comes with automatic reloading magic. You can reload all changed modules before executing a new line.

如果你仅一次编辑代码，你可能想重新加载子模块（ reload submodules ）。IPython 带有自动重新加载的神奇能力。在执行新一行前，你可以重新加载所有更改过的模块。

%load_ext autoreload

%autoreload 2

SciPy & NumPy 模块

SciPy 和 NumPy

SciPy is a Python-based ecosystem of packages for math, science, and engineering. We'll show a quick example below. NumPyis a package for scientific computing with tools for algebra, random number generation, integrating with databases, and managing data. NumPy functions have been imported into the SciPy namespace so users don't have to differentiate between them.

SciPy 是基于 Python 生态的数学库，用于数学，科学，工程计算等。我们在下面将展示一个快速例子。NumPy 是一个科学计算库，用于处理代数，随机数生成，数据库集成，和 管理数据。NumPy 的函数已经导入到 SciPy 的命名空间中，所以你不必区分他们。

In [7]:

```
import scipy as sp
import numpy as np
```

In [8]:

```
s = sp.randn(100)
print len(s)
```

100

We can print the mean of our data into our Notebook. Since the returned value is a NumPy array we can find and print descriptive statistics about the random numbers we created with SciPy.

我们可以打印我们数据的平均数到 Notebook。我们用 SciPy 生成返回值，而这返回值是我们找到和打印关于随机数的描述性统计的 NumPy 数组。

In [9]:

```
print("Mean : {0:8.6f}".format(s.mean()))
```

Mean : -0.118815

Arrays are the central part of NumPy and are more efficient lists of Python. The elements of a NumPy array have to be of the same type, usually float or int.

数组是 NumPy 的中心部分，和更有效的 Python 列表。NumPy 数组的元素必需是相同的类型，通常是浮点或者整型。

In [10]:

```
x = np.array([42,47,11], int)
x = np.array([42,47,11], int)
x
```

Out[10]:

array([42, 47, 11])

pandas 模块

If you have a tabular data structure, pandas is the way to go. Dataframes are easy to make, and handle data better than Python lists or tuples. The 10

minutes to pandas guide is a good introduction and a useful exercise. So is Michael Hansen'stutorial.

如果你有一个表格式的数据结构，用 Pandas 是正确的打开方式。数据帧容易生成，和比 Python 列表或者多维数组好处理。10 minutes to pandas 教程是一个好介绍和练习。Michael Hansen'stutorial 也是一样好。

In [11]:

```
import pandas as pd
```

One can create a dataframe in pandas with the handy functions in read_csv. You can read from a URL or a local file. Here we'll make a dataframe from a matplotlib plot we'll make below with Plotly. You can append .py, .r, .m, .jl, .json, .js, .png, .pdf, .png, .svg, .embed, .xlsx, and .csv to any Plotly URL to see extensions of the figure.

你可以在 Pandas 里用 read_csv 函数生成数据帧。你可以从 URL 或者本地文件读取内容。这里我们从一个 matplotlib 图表生成数据帧，这个图表是我们从下面 Ploty 里生成的。参看扩展名列表，你可以用 .py, .r, .m, .jl, .json, .js, .png, .pdf, .png, .svg, .embed, .xlsx, and .csv 来做 Plotly URL 的后缀。

In [12]:

```
df = pd.read_csv("https://plot.ly/~MattSundquist/20387.csv")
```

We can describe the dataframe we've just created.

我们刚刚生成的描述性数据帧。

In [13]:

```
df.describe() #
```

Out[13]:

	line0_volts	line0_time, line1_time, line3_time
count	200.000000	200.000000
mean	0.434498	0.995000

	line0_volts	line0_time, line1_time, line3_time
std	0.243705	0.578792
min	0.136695	0.000000
25%	0.224812	0.497500
50%	0.369728	0.995000
75%	0.608055	1.492500
max	1.000000	1.990000

We can also examine how many rows we have. Calling the columns property gives you columns.

我们还可以计算一共有多少行。调用 columns 属性给你列数。

In [14]:

```
len(df)
```

Out[14]:

200

Calling .head() will print the first five rows and column headers by default, or we can specify a number.

调用.head()将默认打印前五行和每列的标头，或者我们可以指定一个数来显示。

In [15]:

```
df.head()  # examine dataframe
```

Out[15]:

	line0_volts	line0_time, line1_time, line3_time
0	1.000000	0.00

	line0_volts	line0_time, line1_time, line3_time	
1	0.990050	0.01	
2	0.980199	0.02	
3	0.970446	0.03	
4	0.960789	0.04	

We can rename our columns once we've made a dataframe.

一旦我们已经生成了数据帧，我们可以一次改变所有列的名字

In [16]:

```
df.columns = ["volts_1", "time_1", "volts_2", "volts_2",
              "time_2", "volts_4"]
```

In [17]:

```
df.head(1)
```

Out[17]:

	volts_1	time_1	volts_2
0	1.0	0.0	0.0

We can use pandas for statistics and to examine our rows and columns.

我们可以用 Pandas 来统计行数和列数。

In [18]:

```
df.volts_1.head()
```

Out[18]:

```
0    1.000000
1    0.990050
2    0.980199
3    0.970446
4    0.960789
Name: volts_1, dtype: float64
```

In [19]:

df.volts_1.std()
Out[19]:
0.24370527032639769

Most pandas functions also work on an entire dataframe. For example, calling std() calculates the standard deviation for each column.
大量 Pandas 函数可以处理整个数据帧。例如，调用 std()计算每列的标准差。
In [20]:

```
df.std()
```

Copy to clipboard!Copy to clipboard!
Out[20]:
volts_1 0.243705
time_1 0.578792
volts_2 0.708881
volts_2 0.324605
time_2 0.591608
volts_4 0.340231
dtype: float64

用 matplotlib 内联画图

Plotting with matplotlib inline

You can use matplotlib inside your IPython Notebook by calling %matplotlib inline, which has the advantage of keeping your plots in one place. If you're having trouble running matplotlib, here are a few common solutions. %matplotlib inline activates the inline backend and calls images as static pngs. A new option--%matplotlib notebook--lets you interact with the plot in a Notebook. This works in IPython 3.x; for older IPython versions, use %matplotlib nbagg.

你可以在 IPython Notebook 内部通过调用%matlotlib inline 去使用 matplotlib，它的好处是可以让你保持在一个地方绘图。如果你在使用 matplotlib 遇到问题，这里有一些常用的解决方案。%matplotlib%激活内联后端和调用静态图如 png。新选项--%matplotlib notebook--让你在 Notebook 与绘图交互，它在 IPython3.x 可用，对于老版本的 IPython，则使用%matplotlib nbagg。

In [21]:

```
%matplotlib inline
import matplotlib.pyplot as plt
#side-stepping mpl backend 反复使用 mpl 后端
import matplotlib.gridspec as gridspec # subplots 子图
```

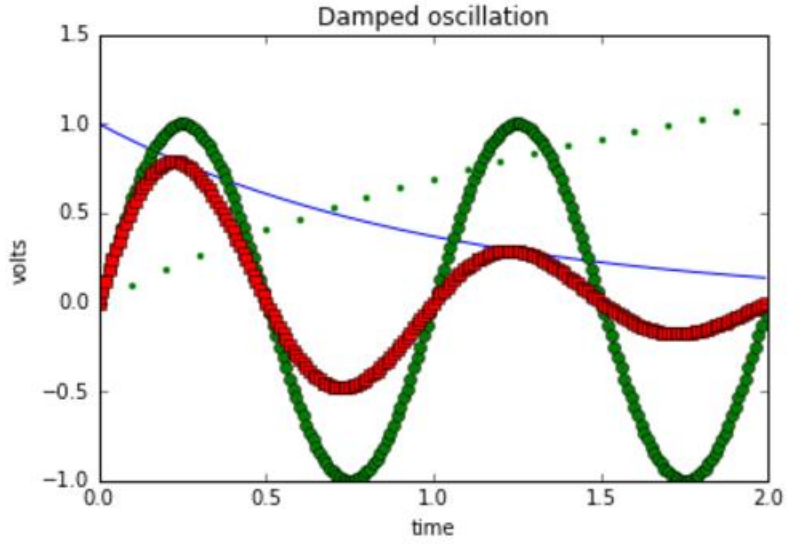
In [22]:

```
import plotly.plotly as py
from plotly.graph_objs import *
import plotly.tools as tls
```

In [23]:

```
fig1 = plt.figure()
# Make a legend for specific lines.使图例成为特殊直线
import matplotlib.pyplot as plt
import numpy as np

t1 = np.arange(0.0, 2.0, 0.1)
t2 = np.arange(0.0, 2.0, 0.01)
# note that plot returns a list of lines.  The "l1, = plot" usage# extracts the first element of the list into l1 using tuple# unpacking.  So l1 is a Line2D instance, not a sequence of lines
#注意绘图返回直线列表，l1=plot 语句通过解包元组的方式提取出 l1 列表的第一个元素。所以 l1 是一个 2 维直线的实例，不是一个直线序列。
l1, = plt.plot(t2, np.exp(-t2))
l2, l3 = plt.plot(t2, np.sin(2 * np.pi * t2), '--go', t1, np.log(1 + t1), '.')
l4, = plt.plot(t2, np.exp(-t2) * np.sin(2 * np.pi * t2), 'rs-.')
plt.xlabel('time')
```

```
plt.ylabel('volts')
plt.title('Damped oscillation')
plt.show()
```



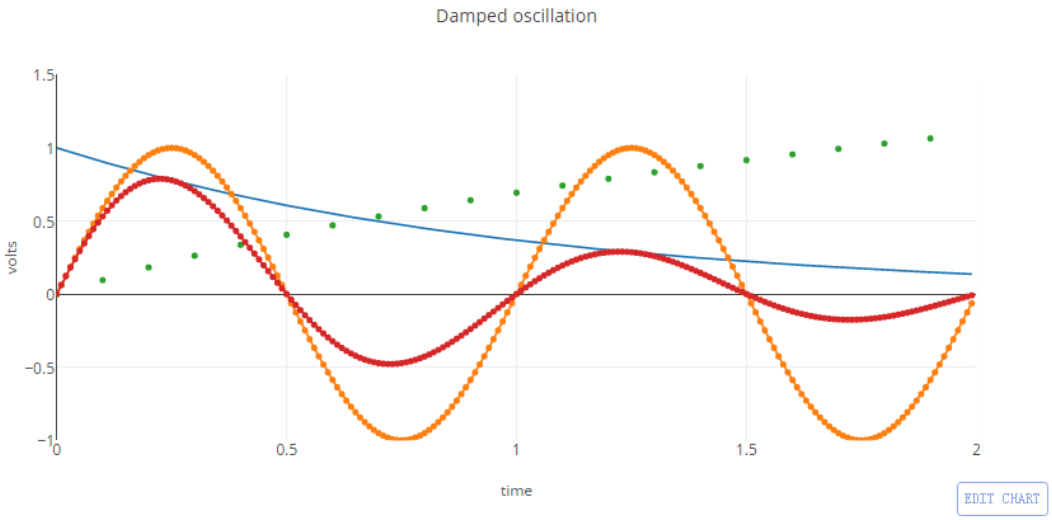
Now we can do a bit of interactive plotting. Head to the Plotly getting started page to get a key and install the API. Calling the plot with iplot automaticallly generates an interactive version of the matplotlib plot inside the Notebook in an iframe. You can control the privacy with sharing set to public, private, or secret. We'll use strip_style to apply the Plotly defaults. You can filter zoom by clicking and dragging and see text if you hover your mouse.

现在我们可以做一些交互性的绘图，去 Plotly getting started 页面得到密钥并安装 API，在 Notebook 调用 plot 会自动加载 iplot 并生成交互型版本的 matplotlib 绘图。你可以通过将共享设置成公用，私用或秘密来控制隐私性。我们可以使用 strip_style 去设置 Plotly 默认值。当你悬停鼠标时可以通过点击拖曳来过滤缩放以便于看到文本。

In [24]:

```
py.iplot_mpl(fig1, strip_style = True, filename='ipython/mpl_example')
```

Out[24]:



用 mpld3 绘图

Plotting with mpld3

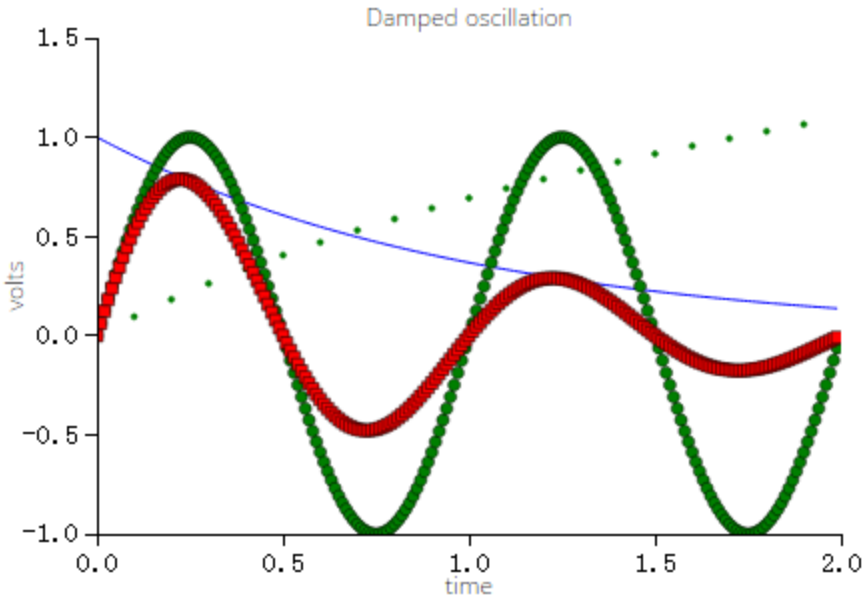
We can use mpld3 to make interactive plots in the Notebook from matplotlib figures.

我们可以通过 mpld3 使 Notebook 的 matplotlib 图像成为交互型绘图。

In [26]:

```
import mpld3
mpld3.display(fig1)
```

Out[26]:



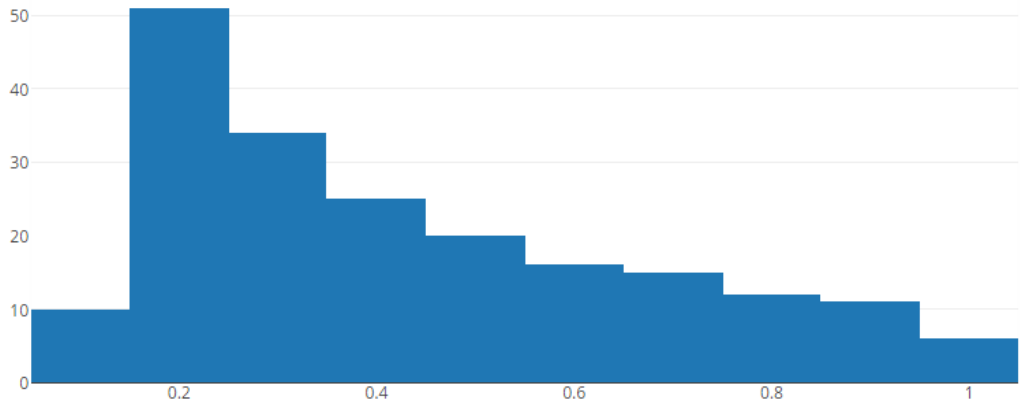
We can make interactive plots from the same dataframe we made with pandas above.

同样我们可以把 dataframe 做成交互型绘图,我们前面 pandas 也可以如此。

In[30]:

```
volts_histogram_plot = [{'x': df['volts_1'], 'type': 'histogram'}]
data_histogram = Data(volts_histogram_plot)
fig_histogram = Figure(data=data_histogram)
py.iplot(fig_histogram, filename='pandas/volts_histogram')
```

Out[30]:

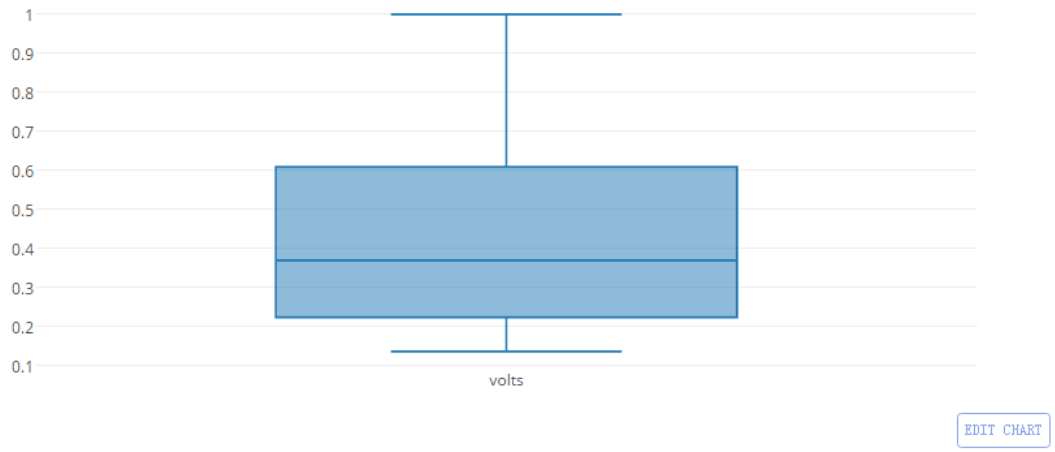


In [31]:

```
volts_jitter_plot = [{'y': df['volts_1'], 'name': 'volts','type': 'box',}]
data_jitter = Data(volts_jitter_plot)
fig_jitter = Figure(data=data_jitter)
```

```
py.iplot(fig_jitter, filename='pandas/volts_boxplot')
```

Out[31]:



For plotting directly from a dataframe, you can use cufflinks.
如果要从 dataframe 直接画图，你可以使用 cufflinks。

绘制交互性地图

Plotting Interactive Maps

Let's make a map. We'll read in the data from another Plotly graph showing the number of electoral votes per state in the U.S. As before, if we add .csv to the end of the plot URL, we can use pandas to make a dataframe.
让我们来画地图，我们读取其它 Plotly 图中的数据演示了美国每一州的选票数据。正如以前，当我们添加.csv 至最后一行的绘图 URL，我们可以使用 pandas 做一个 dataframe。

In[41]:

```
# Learn about API authentication here: https://plot.ly/python/getting-started
#了解一下官方 API 在此：https://plot.ly/python/getting-started
# Find your api_key here: https://plot.ly/settings/api
#找到你的 api 密钥在此: https://plot.ly/settings/api
import plotly.plotly as py
df = pd.read_csv('https://plot.ly/~Dreamshot/5718/electoral-college-votes-by-us-state/.csv')
for col in df.columns:
    df[col] = df[col].astype(str)
```

In [42]:

```
df.head()
```

Out[42]:

输出一张表格，无法复制。

In [43]:

```
df.columns = ["state", "votes"] # change column names 改变列名
```

In [44]:

```
df.head(1)
```

Out[44]:

输出一张表格，无法复制。

Now we can make an interactive D3.js graph directly from pandas. See the pandas maps documentation to learn more.

现在我们直接从 pandas 做了一个交互型 D3.js 图像，阅读 pandas maps 文档了解更多。

In [45]:

```
scl = [[0.0, 'rgb(242,240,247)'],[0.2, 'rgb(218,218,235)'],[0.4, 'rgb(188,189,220)'],\
            [0.6, 'rgb(158,154,200)'],[0.8, 'rgb(117,107,177)'],[1.0, 'rgb(84,39,143)']]
df['text'] = df['state']
    data = [dict(
    type='choropleth',
    colorscale = scl,
    autocolorscale = False,
    locations = df['state'],
    z = df['votes'].astype(float),
    locationmode = 'USA-states',
    text = df['text'],
    hoverinfo = 'location+z',
    marker = dict(
        line = dict (
            color = 'rgb(255,255,255)',
            width = 2
        )
    ),
    colorbar = dict(
        title = "Votes"
```

```
        ))]
    layout = dict(
        title = '2016 Electoral College Votes<br>(Hover for breakdown)',
        geo = dict(
            scope='usa',
            projection=dict( type='albers usa' ),
            showlakes = True,
            lakecolor = 'rgb(255, 255, 255)'
        ))
        fig = dict(data=data, layout=layout)
    py.iplot(fig, validate=False, filename='d3-electoral-map')
```

Out[45]:

3D 绘图

3D Plotting

Using Numpy and Plotly, we can make interactive 3D plots in the Notebook as well.

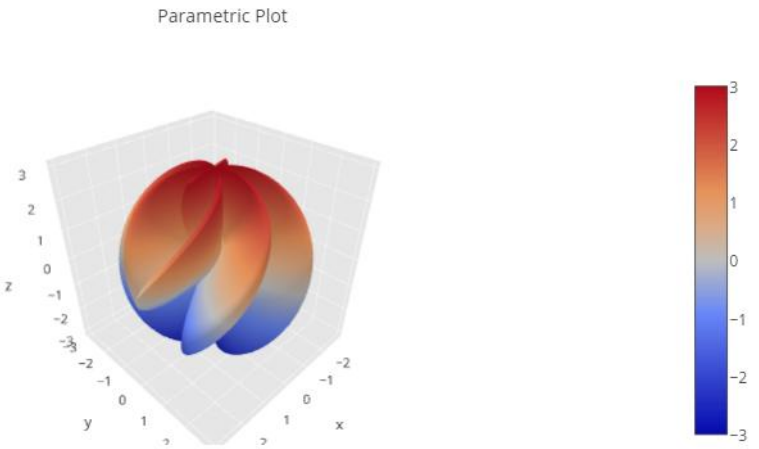
使用 Numpy 和 Plotly,我们在 Notebook 也可以使用交互型 3D plots。

In [46]:

```
    import plotly.plotly as py
    from plotly.graph_objs import *
    import numpy as np
    s = np.linspace(0, 2 * np.pi, 240)t = np.linspace(0, np.pi, 240)tGrid, sGrid =
np.meshgrid(s, t)
    r = 2 + np.sin(7 * sGrid + 5 * tGrid)  # r = 2 + sin(7s+5t)
    x = r * np.cos(sGrid) * np.sin(tGrid)  # x = r*cos(s)*sin(t)
    y = r * np.sin(sGrid) * np.sin(tGrid)  # y = r*sin(s)*sin(t)
    z = r * np.cos(tGrid)                  # z = r*cos(t)
    surface = Surface(x=x, y=y, z=z)data = Data([surface])
    layout = Layout(
        title='Parametric Plot',
        scene=Scene(
            xaxis=XAxis(
                gridcolor='rgb(255, 255, 255)',
```

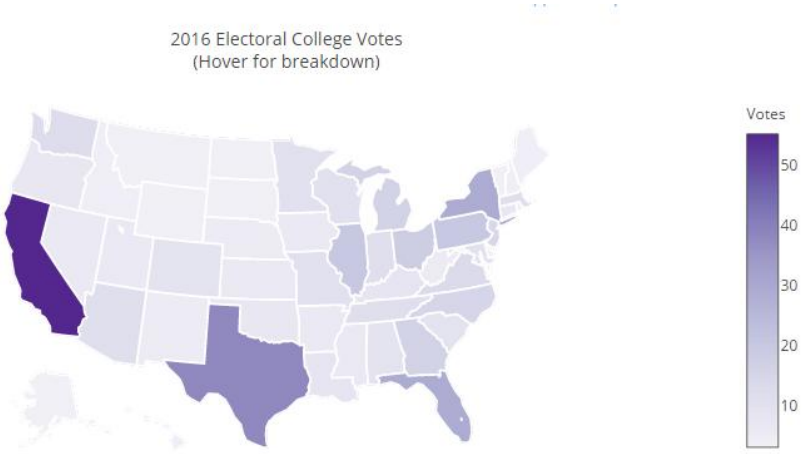
```
            zerolinecolor='rgb(255, 255, 255)',
            showbackground=True,
            backgroundcolor='rgb(230, 230,230)'
        ),
        yaxis=YAxis(
            gridcolor='rgb(255, 255, 255)',
            zerolinecolor='rgb(255, 255, 255)',
            showbackground=True,
            backgroundcolor='rgb(230, 230,230)'
        ),
        zaxis=ZAxis(
            gridcolor='rgb(255, 255, 255)',
            zerolinecolor='rgb(255, 255, 255)',
            showbackground=True,
            backgroundcolor='rgb(230, 230,230)'
        )
    ))
fig = Figure(data=data, layout=layout)py.iplot(fig, filename='Parametric_plot')
```

Out[46]:



Note the possible interactions.

注意所有可能的交互。



Seaborn 绘图

Plotting with Seaborn

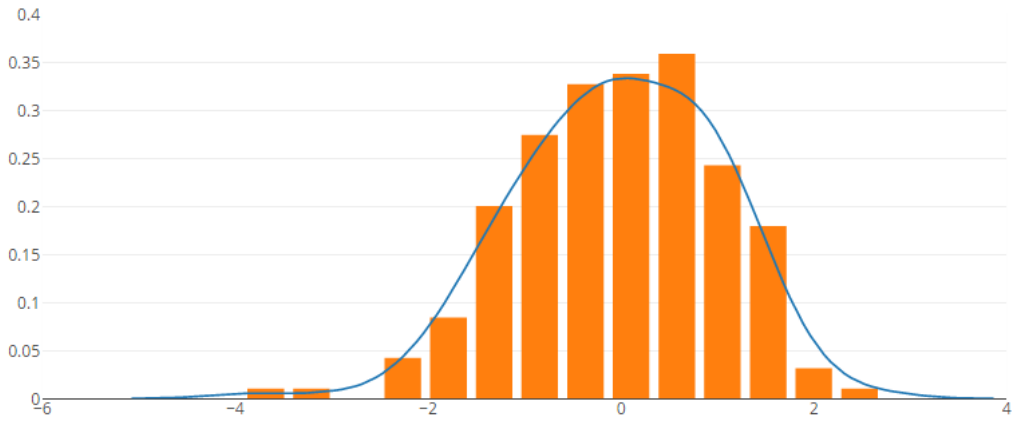
Seaborn is focused on statistical plotting and plot types. Here we show how you can combine plot types

Seaborn 关注在统计图和绘图类型上，这里我们演示如何绑定绘图类型。

In [49]:

```
    from numpy.random
    import randnfrom scipy
    import statsimport matplotlib as mplimport seaborn as sns
    In [50]:
    fig16 = plt.figure()
    sns.set_palette("hls")mpl.rc("figure", figsize=(8, 4))data =
randn(200)sns.distplot(data);
    py.iplot_mpl(fig16, strip_style = True)
```

Out[50]:



R 语言魔法

Rmagic

Rmagic lets us run R in our Notebook and embed ggplot2 plots. You can execute code in R, and pull some of the results back into the Python namespace. The return value is determined when rpy2 returns the result of evaluating the final line. Multiple R lines can be executed by joining them with semicolons.

Rmagic 让我们可以在 Notebook 运行 R 和内置的 ggplot2 绘图。你可以运行 R 脚本并拿出结果放在 Python 命名空间。当 rpy2 返回评估最行一条直线时返回值已经被确定，多元 R 直线可以通过分号联结他们来执行。

In [52]:

```
%load_ext rpy2.ipython
```

In [53]:

```
%R X=c(1,4,5,7); sd(X); mean(X)
```

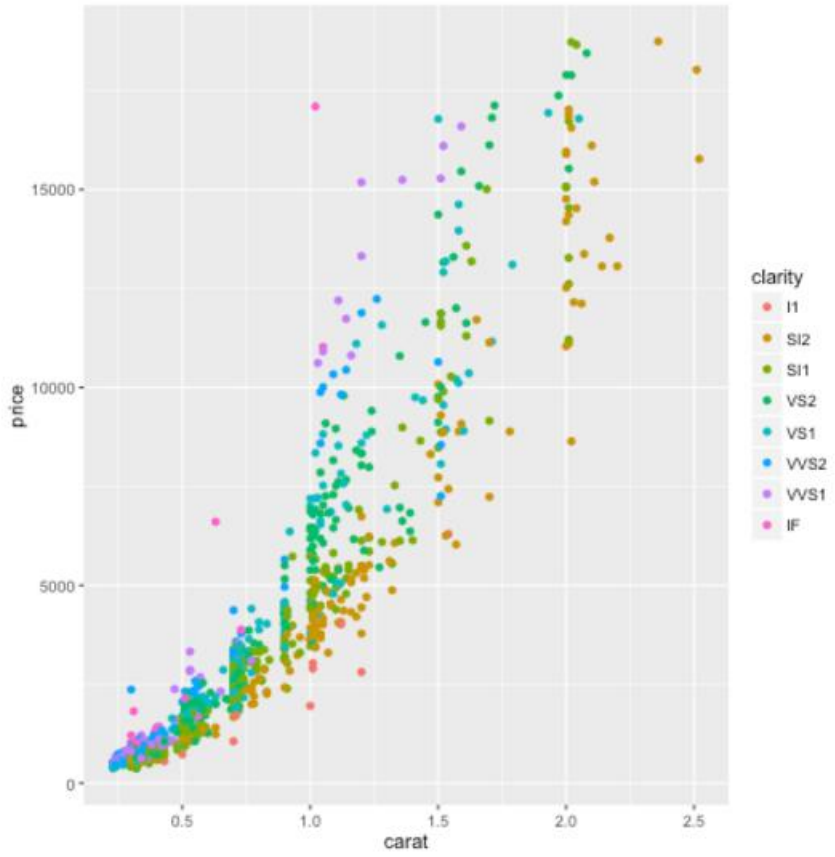
Out[53]:

array([4.25])

In [54]:

```
%%R library(ggplot2)
```

```
dsamp <- diamonds[sample(nrow(diamonds), 1000), ]
qplot(carat, price, data=dsamp, colour=clarity)
```



内置

Embedding

The Notebook allows us to embed iframes. For example, from YouTube.

Notebook 允许我们内置 iframes，例如 YouTube。

In [55]:

```
from IPython.display import YouTubeVideo
```

In [56]:

```
YouTubeVideo("p86BPM1GV8M")
```

Out[56]:

We can embed graphs. The plotly ggplot2 figure converter lets us make the plot above into an interactive plot. Then you can call the plot in the NB.

我们可以内置图片，plotly ggplot2 图片转换器可以让我们把上面图片转成交互型绘图，然后你可以在 NB 调用该图。

In [57]:

```
tls.embed('https://plot.ly/~MattSundquist/20391/price-vs-carat/')
```

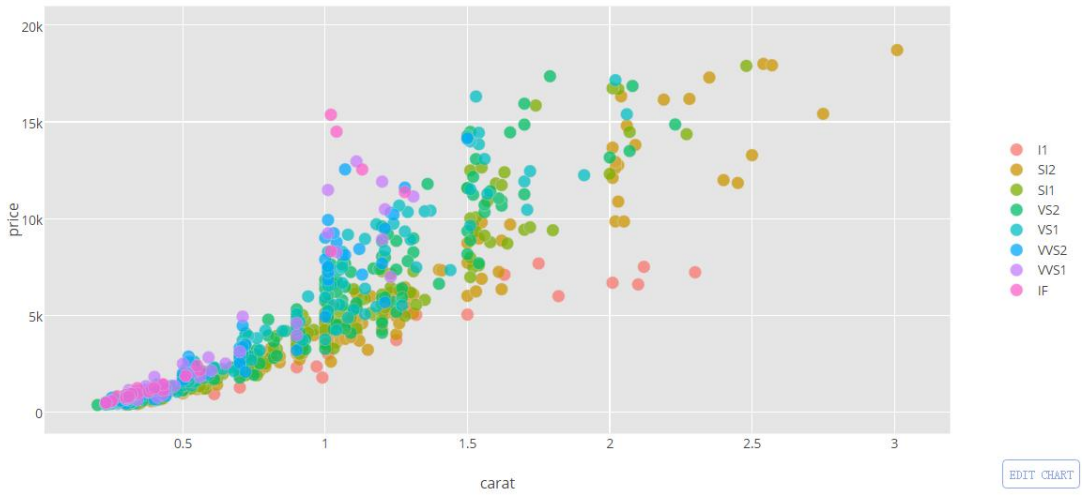
Out[57]:

LaTeX 版式

We can embed LaTeX inside a Notebook by putting a $$ around our math, then run the cell as a Markdown cell. For example, the cell below is $$c = \sqrt{a^2 + b^2}$$, but the Notebook renders the expression.

$$c = \sqrt{a^2 + b^2}$$

Or, you can display output from Python, as seen here.



我们可以在 Notebook 通过$$括起公式来内置 LaTex，然后运行这个单元格就像 Markdown 单元格一样。例如，下面的单元格是$$c = \sqrt{a^2 + b^2}$$，但是 Notebook 拒绝表达这个公式。

$$c = \sqrt{a^2 + b^2}$$

或者，你可以在 Python 输出结果，在这里看。

In [58]:

```
from IPython.display import display, Math, Latexdisplay(Math(r'F(k) = \int_{-\infty}^{\infty} f(x) e^{2\pi i k} dx'))
$$F(k) = \int_{-\infty}^{\infty} f(x) e^{2\pi i k} dx$$
```

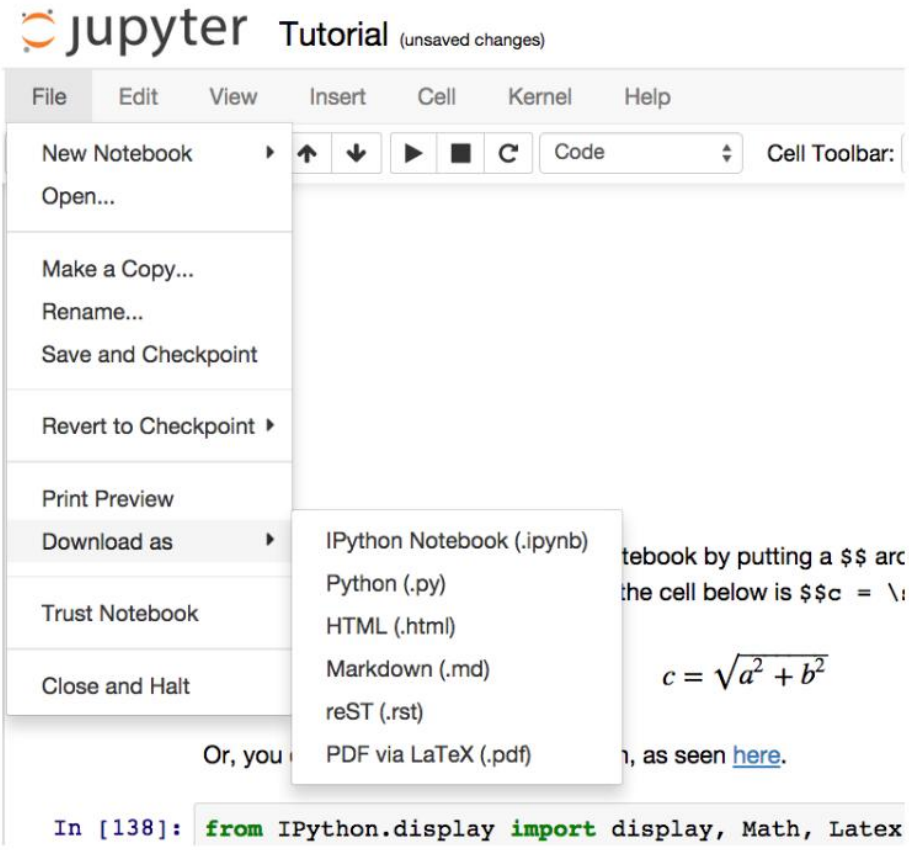
发布

Publishing

We can export the Notebook as an HTML, PDF, .py, .ipynb, Markdown, and reST file. You can also turn your NB into a slideshow. For publishing IPython Notebooks directly to GitHub pages, you can use publisher. You can publish Notebooks on GitHub, and they will be generated as a Notebook on nbviewer.ipython.org. More advanced users can consider using git for version control.

我们可以把 Notebook 导出为 HTML,PDF,.py,.ipynb,Markdown 和 reST 文件。你可以把 NB 转成幻灯片。直接在 GitHub 发布

IPython Notebooks,他们可以在 nbviewer.ipython.org 生成 Notebook。更多资深用户可以考虑 使用 git 版本控制。



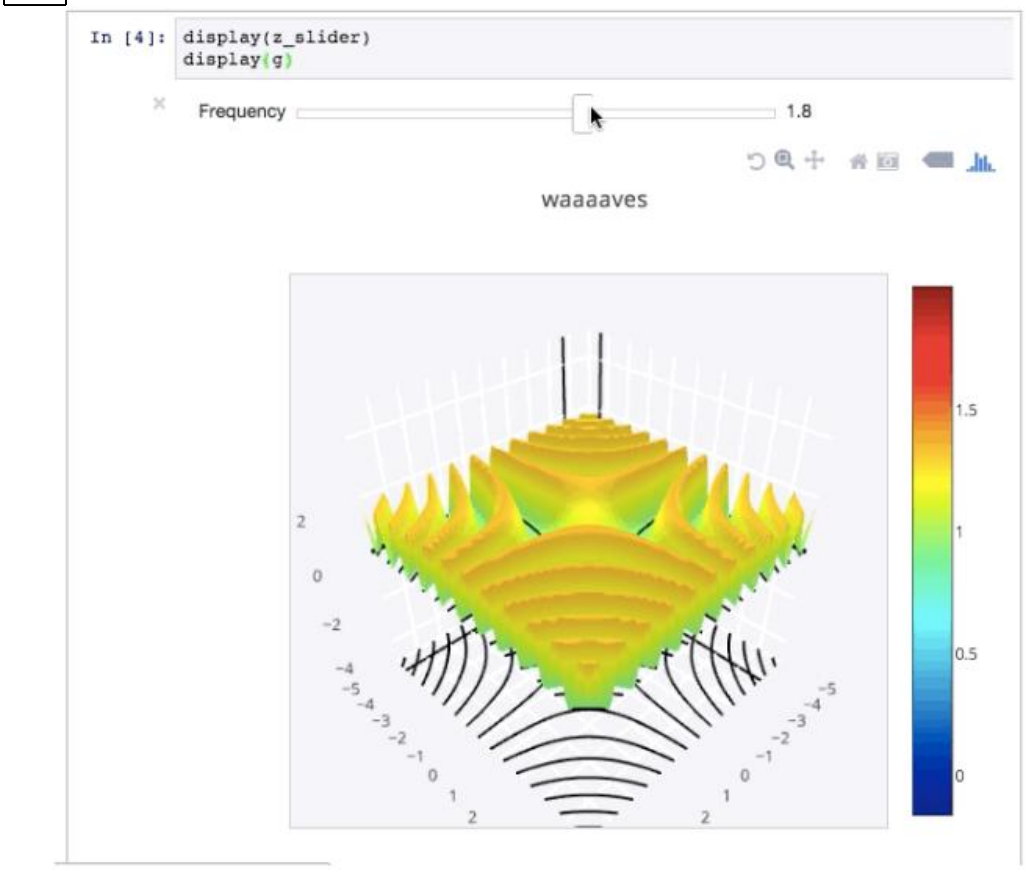
IPython 组件

IPython widgets

IPython widgets allow you to add sliders, widgets, search boxes, and more to your Notebook. See the widget docs for more information. For others to be able to access your work, they'll need IPython. Or, you can use a cloud-based NB option so others can run your work.

IPython 组件允许你在 Notebook 增加滑动窗口，组件，查找框或更多。见 widget docs 了解更多。为了让其它人可以使用的你的作品，他们需要 IPython。或者你可以使用基于云的 NB 选项这样其

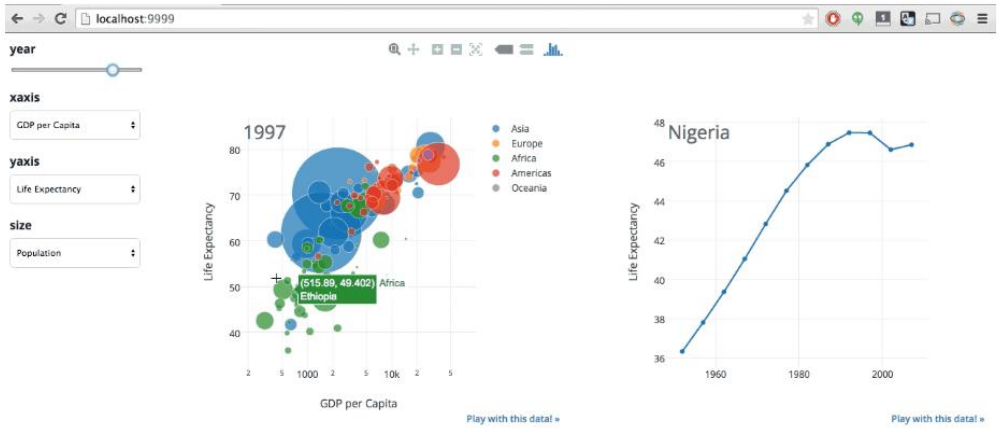
它人可以运行你的作品。



发布 Dash 应用

For users looking to ship and productionize Python apps, dash is an assemblage of Flask, Socketio, Jinja, Plotly and boiler plate CSS and JS for easily creating data visualization web-apps with your Python data analysis backend.

为了用户方便上手和生产化 Python 应用程序，dash 可聚合 Flask,Socketio,jinja,Plotly,模板文件 CSS 和 JS，并很容易创建数据可视化的网络应用，成为你 Python 数据分析后端。

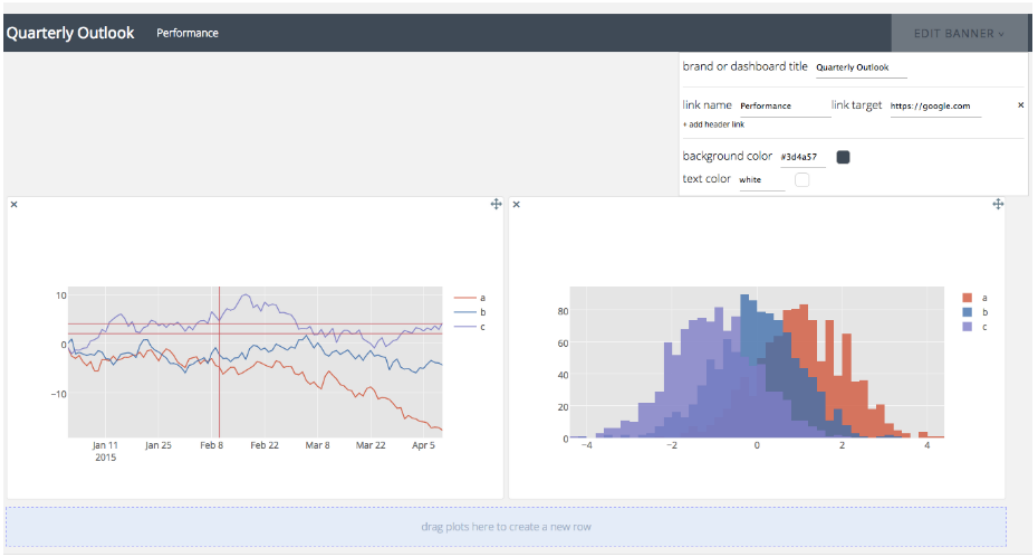


Publishing Dash Apps

发布仪表盘

Users publishing interactive graphs can also use dashboards.ly to arrange a plot with a drag and drop interface. These dashboards can be published, embedded, and shared.

用户发布交互型图也可以使用 dashbords.ly 通过拖曳和清除接口来排列图片。这些 dashboards 可以发布，内置和共享。

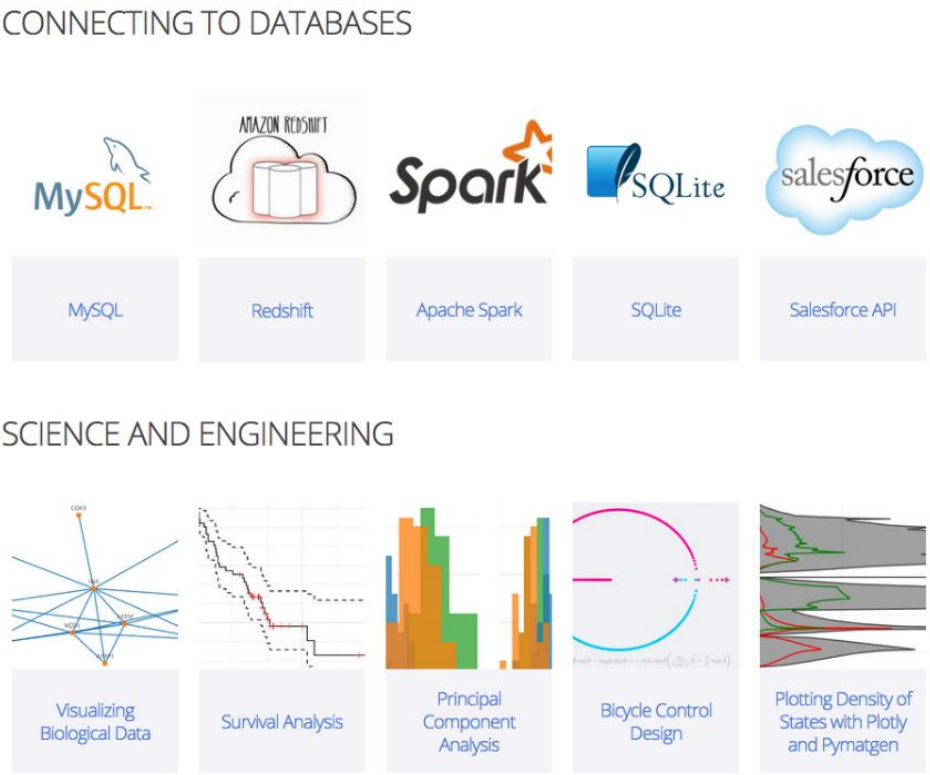


Publishing dashboards

IPython 画廊

IPython gallery

For more IPython tutorials, see the IPython gallery.



At the end of a Notebook, we can style a notebook with these three lines of code.

更多 IPython 教程，见 IPython gallery。Notebook 最终部分，我们可以看到定制 notebook 风格的三行代码。

```
from IPython.core.display import HTML
import urllib2
HTML(urllib2.urlopen('http://bit.ly/1Bf5Hft').read())
```

金融图形

Financial Charts

绘制 ohlc 简化蜡烛图

ohlc-charts

OHLC Charts in Python 使用
How to make interactive OHLC charts in Python with Plotly. Six examples of OHLC charts with Pandas, time series, and yahoo finance data.
如何在 python 中使用 plotly 构建交互式蜡烛图。下面六个使用 pandas,时间序列，和雅虎财经数据构建示例。
OHLC Charts in Python
使用 python 绘制蜡烛图

```
import plotly
plotly.__version__
```

'1.7.9'

使用 pandas 构建简单蜡烛图

Simple OHLC Chart with Pandas

```
import plotly.plotly as py
from plotly.tools import FigureFactory as FF
from datetime import datetime

import pandas.io.data as web

df = web.DataReader("aapl", 'yahoo', datetime(2008, 8, 15), datetime(2008, 10, 15))
fig = FF.create_ohlc(df.Open, df.High, df.Low, df.Close, dates=df.index)

py.iplot(fig, filename='finance/aapl-ohlc')
```

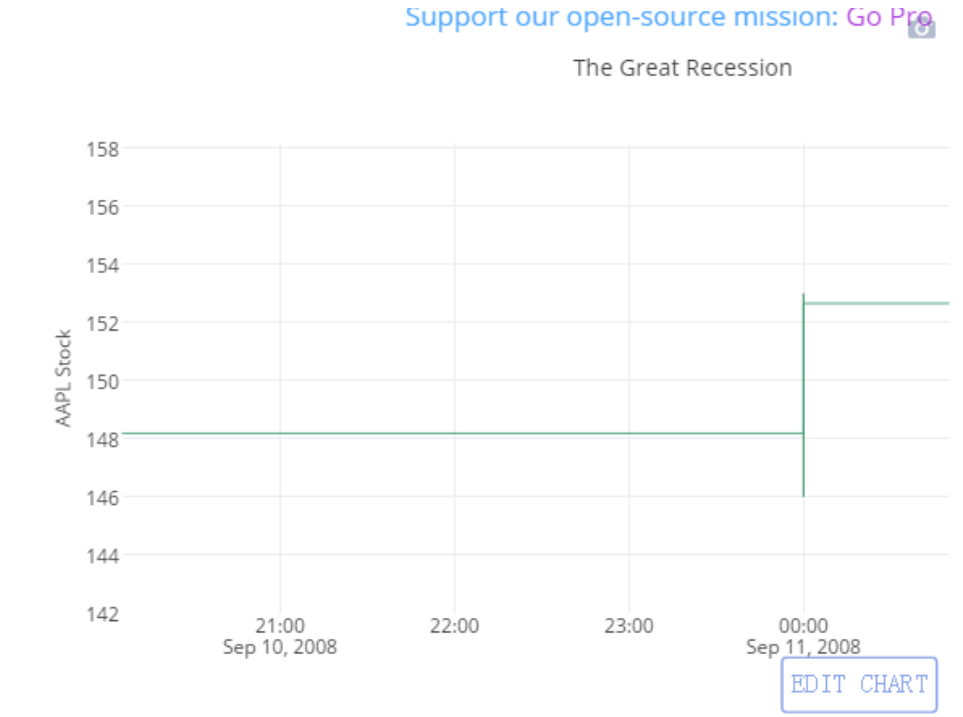
Out[3]:

使用文本和注释自定义图形

Customizing the Figure with Text and Annotations

```
import plotly.plotly as py
from plotly.tools import FigureFactory as FF

import pandas.io.data as web

df = web.DataReader("aapl", 'yahoo', datetime(2008, 8, 15), datetime(2008, 10, 15))
fig = FF.create_ohlc(df.Open, df.High, df.Low, df.Close, dates=df.index)

# Update the fig - all options here: https://plot.ly/python/reference/#Layout
fig['layout'].update({
    'title': 'The Great Recession',
    'yaxis': {'title': 'AAPL Stock'},
    'shapes': [{
        'x0': '2008-09-15', 'x1': '2008-09-15', 'type': 'line',
        'y0': 0, 'y1': 1, 'xref': 'x', 'yref': 'paper',
        'line': {'color': 'rgb(40,40,40)', 'width': 0.5}
    }],
    'annotations': [{
        'text': "the fall of Lehman Brothers",
        'x': '2008-09-15', 'y': 1.02,
        'xref': 'x', 'yref': 'paper',
        'showarrow': False, 'xanchor': 'left'
    }]
})

py.iplot(fig, filename='finance/aapl-recession-ohlc', validate=False)
```

Out[4]:



自定义蜡烛图颜色

Custom OHLC Chart Colors

```
import plotly.plotly as py
from plotly.tools import FigureFactory as FF
from plotly.graph_objs import Line, Marker
from datetime import datetime

import pandas.io.data as web

df = web.DataReader("aapl", 'yahoo', datetime(2008, 1, 1), datetime(2009, 4, 1))

# Make increasing ohlc sticks and customize their color and name
fig_increasing = FF.create_ohlc(df.Open, df.High, df.Low, df.Close, dates=df.index,
    direction='increasing', name='AAPL',
    line=Line(color='rgb(150, 200, 250)'))

# Make decreasing ohlc sticks and customize their color and name
fig_decreasing = FF.create_ohlc(df.Open, df.High, df.Low, df.Close, dates=df.index,
    direction='decreasing',
    line=Line(color='rgb(128, 128, 128)'))

# Initialize the figure
fig = fig_increasing

# Add decreasing data with .extend()
fig['data'].extend(fig_decreasing['data'])

py.iplot(fig, filename='finance/aapl-ohlc-colors', validate=False)
```

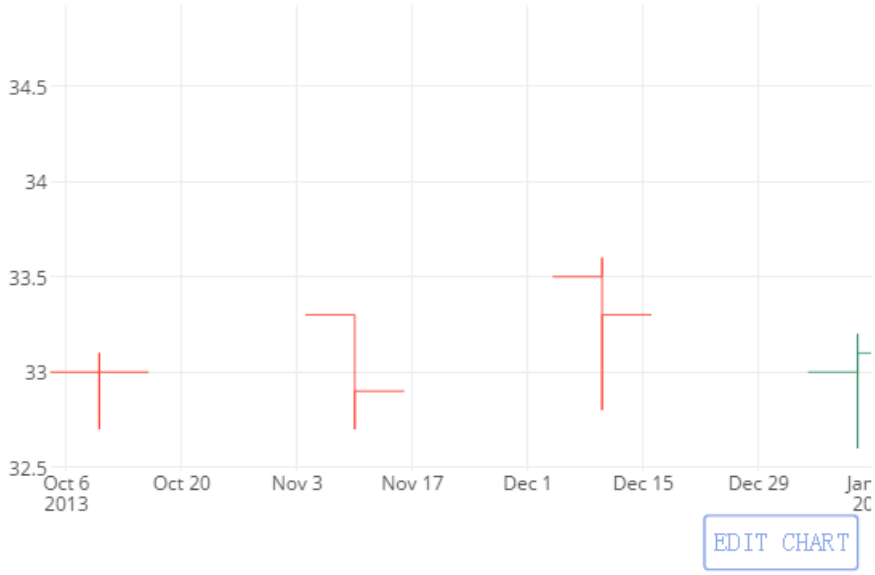
Out[5]:

时间序列绘图示例

Simple Example with datetime Objects

```
import plotly.plotly as py
from plotly.tools import FigureFactory as FF

from datetime import datetime

# Add data
open_data = [33.0, 33.3, 33.5, 33.0, 34.1]
high_data = [33.1, 33.3, 33.6, 33.2, 34.8]
low_data = [32.7, 32.7, 32.8, 32.6, 32.8]
close_data = [33.0, 32.9, 33.3, 33.1, 33.1]
dates = [datetime(year=2013, month=10, day=10),
        datetime(year=2013, month=11, day=10),
        datetime(year=2013, month=12, day=10),
        datetime(year=2014, month=1, day=10),
        datetime(year=2014, month=2, day=10)]

# Create ohlc
fig = FF.create_ohlc(open_data, high_data,
    low_data, close_data, dates=dates)

py.iplot(fig, filename='finance/simple-ohlc', validate=False)
```

Out[6]:



Reference

参考文档

Reference

```
help(FF.create_ohlc)

Help on function create_ohlc in module plotly.tools:

create_ohlc(open, high, low, close, dates=None, direction='both', **kwargs)
    BETA function that creates an ohlc chart

    :param (list) open: opening values
    :param (list) high: high values
    :param (list) low: low values
    :param (list) close: closing
    :param (list) dates: list of datetime objects. Default: None
    :param (string) direction: direction can be 'increasing', 'decreasing',
        or 'both'. When the direction is 'increasing', the returned figure
        consists of all units where the close value is greater than the
        corresponding open value, and when the direction is 'decreasing',
        the returned figure consists of all units where the close value is
        less than or equal to the corresponding open value. When the
        direction is 'both', both increasing and decreasing units are
        returned. Default: 'both'
```

```
:param kwargs: kwargs passed through plotly.graph_objs.Scatter.
    These kwargs describe other attributes about the ohlc Scatter trace
    such as the color or the legend name. For more information on valid
    kwargs call help(plotly.graph_objs.Scatter)

:rtype (dict): returns a representation of an ohlc chart figure.

Example 1: Simple OHLC chart from a Pandas DataFrame
```
import plotly.plotly as py
from plotly.tools import FigureFactory as FF
from datetime import datetime

import pandas.io.data as web

df = web.DataReader("aapl", 'yahoo', datetime(2008, 8, 15), datetime(2008, 10, 15))
fig = FF.create_ohlc(df.Open, df.High, df.Low, df.Close, dates=df.index)

py.plot(fig, filename='finance/aapl-ohlc')
```

Example 2: Add text and annotations to the OHLC chart
```
import plotly.plotly as py
from plotly.tools import FigureFactory as FF
from datetime import datetime

import pandas.io.data as web

df = web.DataReader("aapl", 'yahoo', datetime(2008, 8, 15), datetime(2008, 10, 15))
fig = FF.create_ohlc(df.Open, df.High, df.Low, df.Close, dates=df.index)

# Update the fig - all options here: https://plot.ly/python/reference/#Layout
fig['layout'].update({
    'title': 'The Great Recession',
    'yaxis': {'title': 'AAPL Stock'},
    'shapes': [{
        'x0': '2008-09-15', 'x1': '2008-09-15', 'type': 'line',
        'y0': 0, 'y1': 1, 'xref': 'x', 'yref': 'paper',
        'line': {'color': 'rgb(40,40,40)', 'width': 0.5}
    }],
    'annotations': [{
        'text': "the fall of Lehman Brothers",
        'x': '2008-09-15', 'y': 1.02,
        'xref': 'x', 'yref': 'paper',
        'showarrow': False, 'xanchor': 'left'
    }]
})

py.plot(fig, filename='finance/aapl-recession-ohlc', validate=False)
```

Example 3: Customize the OHLC colors
```
import plotly.plotly as py
from plotly.tools import FigureFactory as FF
from plotly.graph_objs import Line, Marker
from datetime import datetime

import pandas.io.data as web
```
```

```
df = web.DataReader("aapl", 'yahoo', datetime(2008, 1, 1), datetime(2009, 4, 1))

# Make increasing ohlc sticks and customize their color and name
fig_increasing = FF.create_ohlc(df.Open, df.High, df.Low, df.Close, dates=df.index,
    direction='increasing', name='AAPL',
    line=Line(color='rgb(150, 200, 250)'))

# Make decreasing ohlc sticks and customize their color and name
fig_decreasing = FF.create_ohlc(df.Open, df.High, df.Low, df.Close, dates=df.index,
    direction='decreasing',
    line=Line(color='rgb(128, 128, 128)'))

# Initialize the figure
fig = fig_increasing

# Add decreasing data with .extend()
fig['data'].extend(fig_decreasing['data'])

py.iplot(fig, filename='finance/aapl-ohlc-colors', validate=False)
```

Example 4: OHLC chart with datetime objects

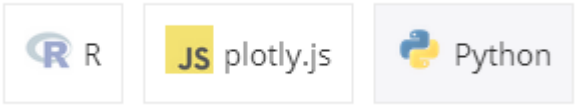
```
import plotly.plotly as py
from plotly.tools import FigureFactory as FF

from datetime import datetime

# Add data
open_data = [33.0, 33.3, 33.5, 33.0, 34.1]
high_data = [33.1, 33.3, 33.6, 33.2, 34.8]
low_data = [32.7, 32.7, 32.8, 32.6, 32.8]
close_data = [33.0, 32.9, 33.3, 33.1, 33.1]
dates = [datetime(year=2013, month=10, day=10),
        datetime(year=2013, month=11, day=10),
        datetime(year=2013, month=12, day=10),
        datetime(year=2014, month=1, day=10),
        datetime(year=2014, month=2, day=10)]

# Create ohlc
fig = FF.create_ohlc(open_data, high_data,
    low_data, close_data, dates=dates)

py.iplot(fig, filename='finance/simple-ohlc', validate=False)
```

蜡烛图

Candlestick Charts in python

How to make interactive candlestick charts in Python with Plotly. Six examples of candlestick charts with Pandas, time series, and yahoo finance data.

如何在 Python 用 plotly 制作蜡烛交互图表。六个蜡烛图的例子使用 pandas,时间序列和雅虎金融数据。



In [1]:

```
import plotly
plotly.__version__
```

Out[1]:

```
'2.0.0'
```

pandas 蜡烛图案例

Simple Example with Pandas

In [2]:

```
import plotly.plotly as py
import plotly.figure_factory as FF
from datetime import datetime

import pandas_datareader.data as web

df = web.DataReader("aapl", 'yahoo', datetime(2007, 10, 1), datetime(2009, 4, 1))
fig = FF.create_candlestick(df.Open, df.High, df.Low, df.Close, dates=df.index)py.iplot(fig,
filename='finance/aapl-candlestick', validate=False)
```

Out[2]:

定制图表文字注解

Customizing the Figure with Text and Annotations

In [3]:

```
import plotly.plotly as py
import plotly.figure_factory as FF
from datetime import datetime
import pandas_datareader.data as web
fig = FF.create_candlestick(df.Open, df.High, df.Low, df.Close, dates=df.index)
# Update the fig - all options here: https://plot.ly/python/reference/#Layout
fig['layout'].update({
        'title': 'The Great Recession',
        'yaxis': {'title': 'AAPL Stock'},
        'shapes': [{
        'x0': '2007-12-01', 'x1': '2007-12-01',
        'y0': 0, 'y1': 1, 'xref': 'x', 'yref': 'paper',
        'line': {'color': 'rgb(30,30,30)', 'width': 1}
    }],
    'annotations': [{
        'x': '2007-12-01', 'y': 0.05, 'xref': 'x', 'yref': 'paper',
        'showarrow': False, 'xanchor': 'left',
        'text': 'Official start of the recession'
    }]
})
py.iplot(fig, filename='finance/aapl-recession-candlestick', validate=False)
```

Out[3]:

定制蜡烛图颜色

Custom Candlestick Colors

In [4]:

```
import plotly.plotly as py
import plotly.figure_factory as FF
from plotly.graph_objs import Line, Marker
from datetime import datetime
import pandas_datareader.data as web
df = web.DataReader("aapl", 'yahoo', datetime(2008, 1, 1), datetime(2009, 4, 1))
fig = FF.create_candlestick(df.Open, df.High, df.Low, df.Close, dates=df.index)

# Make increasing ohlc sticks and customize their color and name
fig_increasing = FF.create_candlestick(df.Open, df.High, df.Low, df.Close, dates=df.index,
        direction='increasing', name='AAPL',
        marker=Marker(color='rgb(150, 200, 250)'),
        line=Line(color='rgb(150, 200, 250)'))

# Make decreasing ohlc sticks and customize their color and name
fig_decreasing = FF.create_candlestick(df.Open, df.High, df.Low, df.Close, dates=df.index,
        direction='decreasing',
        marker=Marker(color='rgb(128, 128, 128)'),
        line=Line(color='rgb(128, 128, 128)'))

# Initialize the figure
fig = fig_increasing

# Add decreasing data with .extend()
fig['data'].extend(fig_decreasing['data'])

py.iplot(fig, filename='finance/aapl-candlestick-custom', validate=False)
```

Out[4]:

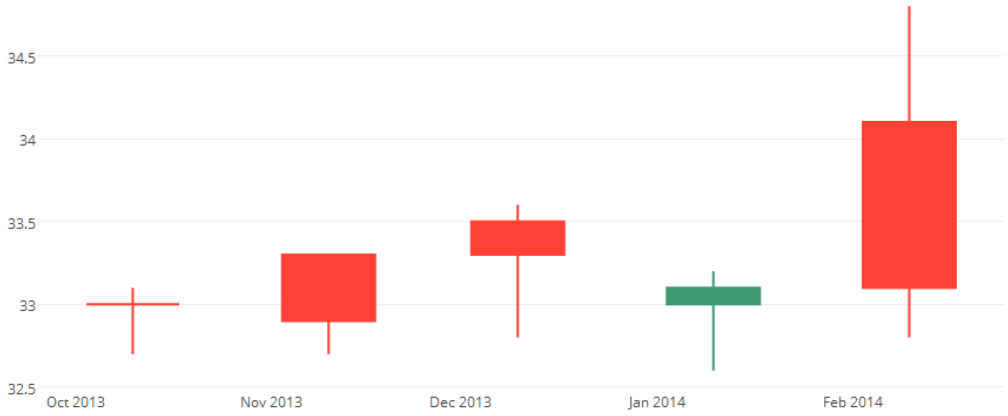
时间序列绘图案例

Simple Example with datetime Objects

In [5]:

```
import plotly.plotly as py
import plotly.figure_factory as FF

from datetime import datetime

# Add data
open_data = [33.0, 33.3, 33.5, 33.0, 34.1]
high_data = [33.1, 33.3, 33.6, 33.2, 34.8]
low_data = [32.7, 32.7, 32.8, 32.6, 32.8]
close_data = [33.0, 32.9, 33.3, 33.1, 33.1]
dates = [datetime(year=2013, month=10, day=10),
         datetime(year=2013, month=11, day=10),
         datetime(year=2013, month=12, day=10),
         datetime(year=2014, month=1, day=10),
         datetime(year=2014, month=2, day=10)]

# Create ohlc
fig = FF.create_candlestick(open_data, high_data,
    low_data, close_data, dates=dates)

py.iplot(fig, filename='finance/simple-candlestick', validate=False)
```

Out[5]:



在蜡烛图添加曲线图

Simple Example Adding a Trace to a Candlestick Chart

In [6]:

```
import plotly.plotly as py
import plotly.figure_factory as FF
from plotly.graph_objs import *

from datetime import datetime
```

```
import pandas_datareader.data as web

# Create Candlestick
df = web.DataReader("aapl", 'yahoo', datetime(2007, 10, 1), datetime(2008, 3, 31))
fig = FF.create_candlestick(df.Open, df.High, df.Low, df.Close, dates=df.index)

# Create Line of open values
add_line = Scatter(
        x=df.index,
        y=df.Open,
        name= 'Open Vals',
        line=Line(color='black')
    )

fig['data'].extend([add_line])
py.iplot(fig, filename='candlestick_and_trace', validate=False)
```

Out[6]:

参考

Reference

In [7]:

```
help(FF.create_candlestick)
Help on function create_candlestick in module plotly.figure_factory._candlestick:

create_candlestick(open, high, low, close, dates=None, direction='both', **kwargs)
    BETA function that creates a candlestick chart

    :param (list) open: opening values
    :param (list) high: high values
    :param (list) low: low values
    :param (list) close: closing values
    :param (list) dates: list of datetime objects. Default: None
    :param (string) direction: direction can be 'increasing', 'decreasing',
        or 'both'. When the direction is 'increasing', the returned figure
        consists of all candlesticks where the close value is greater than
        the corresponding open value, and when the direction is
        'decreasing', the returned figure consists of all candlesticks
```

```
    where the close value is less than or equal to the corresponding
    open value. When the direction is 'both', both increasing and
    decreasing candlesticks are returned. Default: 'both'
:param kwargs: kwargs passed through plotly.graph_objs.Scatter.
    These kwargs describe other attributes about the ohlc Scatter trace
    such as the color or the legend name. For more information on valid
    kwargs call help(plotly.graph_objs.Scatter)

:rtype (dict): returns a representation of candlestick chart figure.

Example 1: Simple candlestick chart from a Pandas DataFrame
```
import plotly.plotly as py
from plotly.figure_factory import create_candlestick
from datetime import datetime

import pandas.io.data as web

df = web.DataReader("aapl", 'yahoo', datetime(2007, 10, 1), datetime(2009, 4, 1))
fig = create_candlestick(df.Open, df.High, df.Low, df.Close, dates=df.index)
py.plot(fig, filename='finance/aapl-candlestick', validate=False)
```

Example 2: Add text and annotations to the candlestick chart
```
fig = create_candlestick(df.Open, df.High, df.Low, df.Close, dates=df.index)
# Update the fig - all options here: https://plot.ly/python/reference/#Layout
fig['layout'].update({
    'title': 'The Great Recession',
    'yaxis': {'title': 'AAPL Stock'},
    'shapes': [{
        'x0': '2007-12-01', 'x1': '2007-12-01',
        'y0': 0, 'y1': 1, 'xref': 'x', 'yref': 'paper',
        'line': {'color': 'rgb(30,30,30)', 'width': 1}
    }],
    'annotations': [{
        'x': '2007-12-01', 'y': 0.05, 'xref': 'x', 'yref': 'paper',
        'showarrow': False, 'xanchor': 'left',
        'text': 'Official start of the recession'
    }]
})
py.plot(fig, filename='finance/aapl-recession-candlestick', validate=False)
```

Example 3: Customize the candlestick colors
```
import plotly.plotly as py
from plotly.figure_factory import create_candlestick
from plotly.graph_objs import Line, Marker
from datetime import datetime

import pandas.io.data as web

df = web.DataReader("aapl", 'yahoo', datetime(2008, 1, 1), datetime(2009, 4, 1))

# Make increasing candlesticks and customize their color and name
fig_increasing = create_candlestick(df.Open, df.High, df.Low, df.Close, dates=df.index,
    direction='increasing', name='AAPL',
    marker=Marker(color='rgb(150, 200, 250)'),
    line=Line(color='rgb(150, 200, 250)'))
```
```

```
# Make decreasing candlesticks and customize their color and name
fig_decreasing = create_candlestick(df.Open, df.High, df.Low, df.Close, dates=df.index,
    direction='decreasing',
    marker=Marker(color='rgb(128, 128, 128)'),
    line=Line(color='rgb(128, 128, 128)'))

# Initialize the figure
fig = fig_increasing

# Add decreasing data with .extend()
fig['data'].extend(fig_decreasing['data'])

py.iplot(fig, filename='finance/aapl-candlestick-custom', validate=False)
```

Example 4: Candlestick chart with datetime objects

```
import plotly.plotly as py
from plotly.figure_factory import create_candlestick

from datetime import datetime

# Add data
open_data = [33.0, 33.3, 33.5, 33.0, 34.1]
high_data = [33.1, 33.3, 33.6, 33.2, 34.8]
low_data = [32.7, 32.7, 32.8, 32.6, 32.8]
close_data = [33.0, 32.9, 33.3, 33.1, 33.1]
dates = [datetime(year=2013, month=10, day=10),
        datetime(year=2013, month=11, day=10),
        datetime(year=2013, month=12, day=10),
        datetime(year=2014, month=1, day=10),
        datetime(year=2014, month=2, day=10)]

# Create ohlc
fig = create_candlestick(open_data, high_data,
    low_data, close_data, dates=dates)

py.iplot(fig, filename='finance/simple-candlestick', validate=False)
```

离线 Python Plotly 绘图

Offline Plots in Plotly in Python

版本检查

Plotly Offline 使得交互试的图表到本地系统。而不是存储这些图表到一个服务器，你的数据和你的图表会保留在你本地的系统里。当你打算分享的时候，你只需要通过一个 Plotly account 把他们发布到网络上，或者到你的公司的内部里 Plotly Enterpise
如果想要开始使用 Plotly Office，更新 plotly 到 1.9.x 或者更高版本

```
from plotly import __version__
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot

print(__version__) # requires version >= 1.9.0
```

plotly 命令行离线命令

你可以从命令行启动一个 Python 脚本来画图。一旦执行这个脚本，他就会一边绘图一边打开一个网页浏览器。

```
from plotly.graph_objs import Scatter, Figure, Layout

plot([Scatter(x=[1, 2, 3], y=[3, 1, 6])])
```

Out[2]:

'file:///Users/Chelsea/Repos/documentation/_posts/python/offline/temp-plot.html'

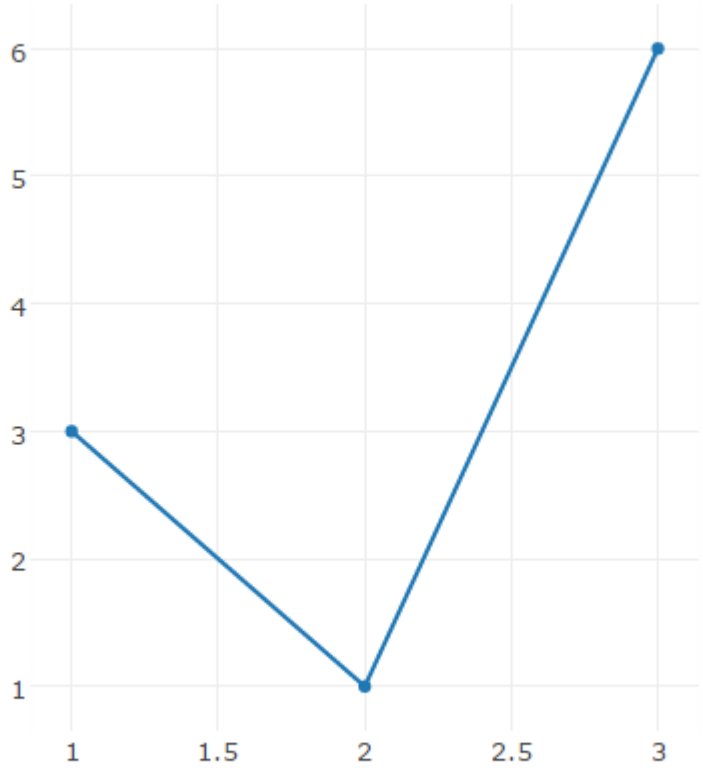
在 notebook 里生成离线图表

你也可以在一个 jupyter notebook 环境里面绘制你的 offline 图表，首先，你需要初始化 plotly notebook 模式，就像下面一样：

```
init_notebook_mode(connected=True)
```

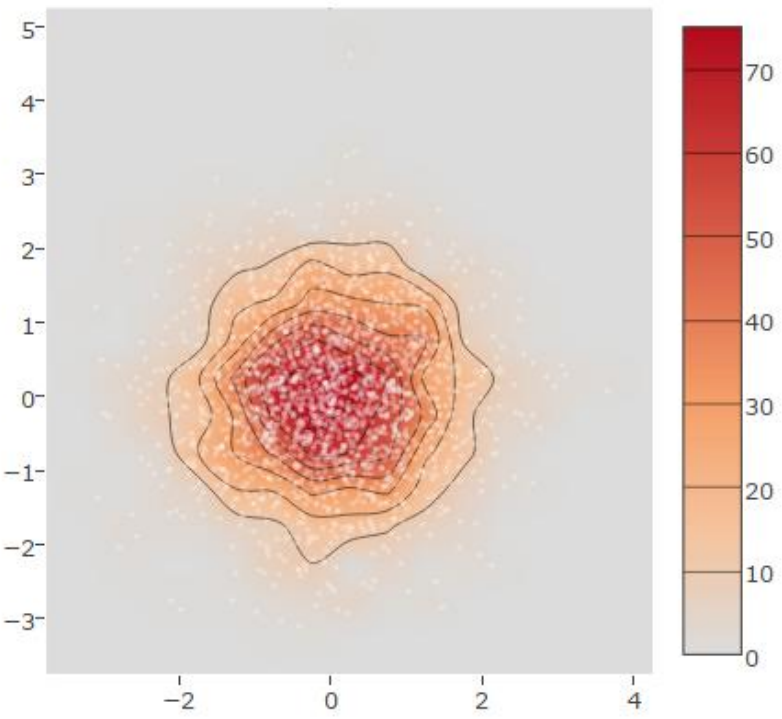
使用 ploly.offline 在每个 IPython notebook 开始的时候运行（Run at the start of every ipython notebook to use plotly.offline. ）。他会把 plotly.js 源文件注入 notebook 里面。

```
iplot([{"x": [1, 2, 3], "y": [3, 1, 6]}])
```

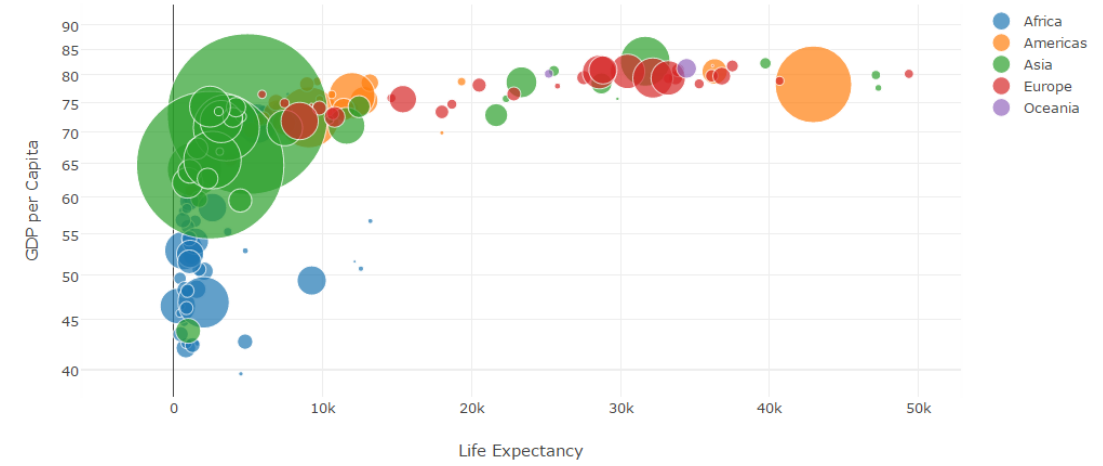


Export to plot.ly »

```
from plotly.graph_objs import *
import numpy as np

x = np.random.randn(2000)
y = np.random.randn(2000)
iplot([Histogram2dContour(x=x, y=y, contours=Contours(coloring='heatmap')),
    Scatter(x=x, y=y, mode='markers', marker=Marker(color='white', size=3,
opacity=0.3))], show_link=False)
```

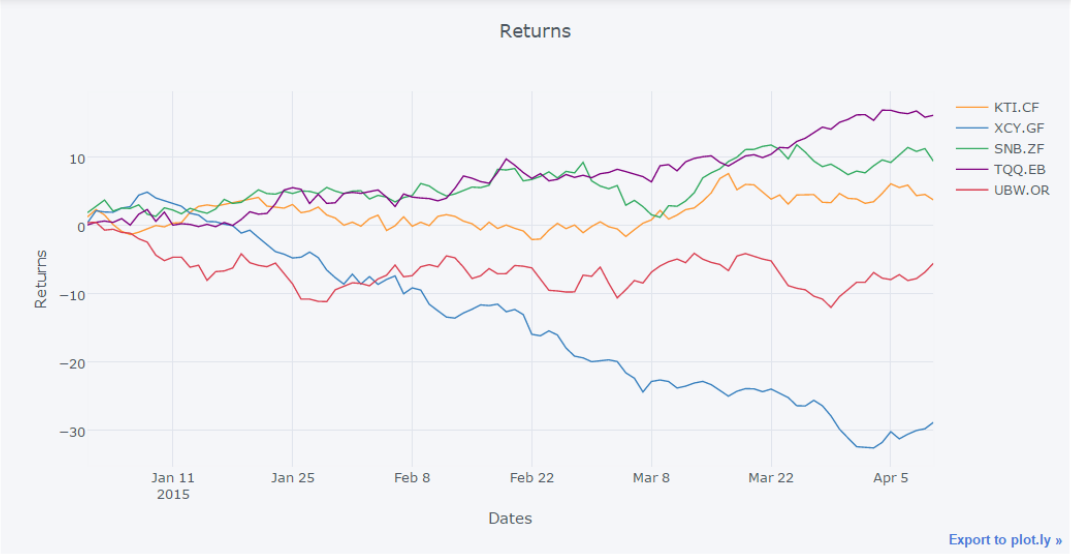



```
    from plotly.graph_objs import *
    import pandas as pd

    df = pd.read_csv('https://plot.ly/~etpinard/191.csv')

    iplot({
        'data': [
            Scatter(x=df[continent+', x'],
                    y=df[continent+', y'],
                    text=df[continent+', text'],
                    marker=Marker(size=df[continent+', size'], sizemode='area',
sizeref=131868,),
                    mode='markers',
                    name=continent) for continent in ['Africa', 'Americas', 'Asia', 'Europe',
'Oceania']
        ],
        'layout': Layout(xaxis=XAxis(title='Life Expectancy'), yaxis=YAxis(title='GDP per
Capita', type='log'))
    }, show_link=False)
```



使用 Cufflinks 绘制 Plotly 离线图表

```
import cufflinks as cf
iplot(cf.datagen.lines().iplot(asFigure=True,
                               kind='scatter',xTitle='Dates',yTitle='Returns',title='Returns'))
```



使用云端绘图

在 plotly.plotly 的所有方法都会与一个 Plotly Cloud 或者 Plotly Enterprise 联系在一起。
get_figure 从 plot.ly 或者 Plotly Enterprise 下载一个 figure。
想要下载 figure，你需要提供资格证:https://plot.ly/python/getting-started/

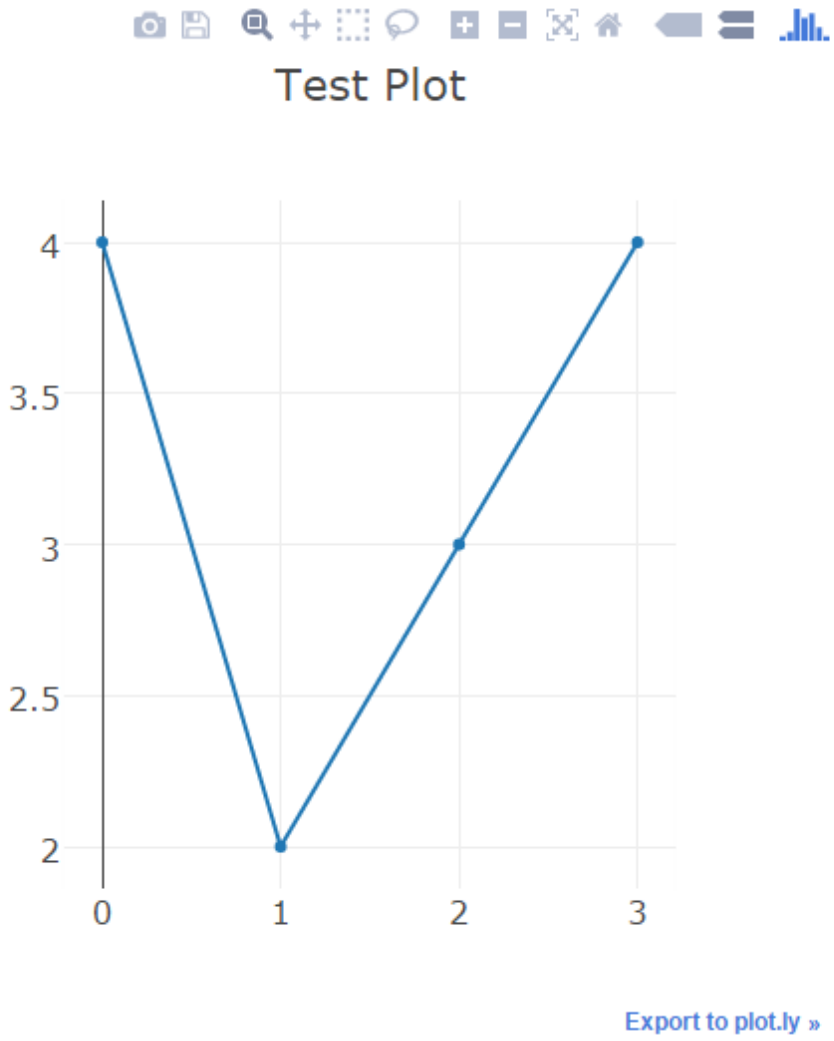
```
import plotly.plotly as py
```

```
fig = py.get_figure('https://plot.ly/~jackp/8715', raw=True)
iplot(fig)
```

导出图形文件

在 IPython notebook 工作环境下，从 plotly offline 生成的图表也可以被存储为一个 image，如下所示：

```
import plotly.offline as offline
import plotly.graph_objs as go

offline.init_notebook_mode()

offline.iplot({'data': [{'y': [4, 2, 3, 4]}],
               'layout': {'title': 'Test Plot',
                          'font': dict(size=16)}},
              image='png')
```



当使用 plotly offline 和 plot()的时候，你可以用如下所示的代码保存一个 image。注意，你必须生成 graph 并且打开 the file 来保存 image

```
import plotly.offline as offline
import plotly.graph_objs as go

offline.plot({'data': [{'y': [4, 2, 3, 4]}],
              'layout': {'title': 'Test Plot',
                         'font': dict(size=16)}},
             image='png')
```

OUT: 'file://d:\\zw_own\\notebooks\\temp-plot.html'

参考

Call help(plotly.offline.iplot) or help(plotly.offline.plot) for more information:

```
import plotlyhelp(plotly.offline.iplot)
```

```
Help on function iplot in module plotly.offline.offline:

iplot(figure_or_data, show_link=True, link_text='Export to plot.ly', validate=True,
image=None, filename='plot_image', image_width=800, image_height=600)
    Draw plotly graphs inside an IPython notebook without
    connecting to an external server.
    To save the chart to Plotly Cloud or Plotly Enterprise, use
    `plotly.plotly.iplot`.
    To embed an image of the chart, use `plotly.image.ishow`.

    figure_or_data -- a plotly.graph_objs.Figure or plotly.graph_objs.Data or
                      dict or list that describes a Plotly graph.
                      See https://plot.ly/python/ for examples of
                      graph descriptions.

    Keyword arguments:
    show_link (default=True) -- display a link in the bottom-right corner of
                                of the chart that will export the chart to
                                Plotly Cloud or Plotly Enterprise
    link_text (default='Export to plot.ly') -- the text of export link
    validate (default=True) -- validate that all of the keys in the figure
                               are valid? omit if your version of plotly.js
                               has become outdated with your version of
                               graph_reference.json or if you need to include
                               extra, unnecessary keys in your figure.
    image (default=None |'png' |'jpeg' |'svg' |'webp') -- This parameter sets
       the format of the image to be downloaded, if we choose to download an
       image. This parameter has a default value of None indicating that no
       image should be downloaded.
    filename (default='plot') -- Sets the name of the file your image
       will be saved to. The extension should not be included.
    image_height (default=600) -- Specifies the height of the image in `px`.
    image_width (default=800) -- Specifies the width of the image in `px`.

    Example:
    ```
    from plotly.offline import init_notebook_mode, iplot
```
```

```
init_notebook_mode()
iplot([{'x': [1, 2, 3], 'y': [5, 2, 7]}])
# We can also download an image of the plot by setting the image to the
format you want. e.g. `image='png'`
iplot([{'x': [1, 2, 3], 'y': [5, 2, 7]}], image='png')
```