Overview of Python Visualization Tools

译文, 原文链接: http://pbpython.com/visualization-tools-1.html (http://pbpython.com/visualization-tools-1.html)

Introduction ¶

在Python的世界里, 有不少优秀的数据可视化的工具. 有时, 即使很有经验的数据从业人员, 也面临着选择工具的难题(幸福的烦忧?) 这篇文章将会列出一些优秀, 广泛应用的Python工具, 并且结合简单的样例Sample提供给感兴趣的人们. 这些有:

- Pandas
- Seaborn
- ggplot
- Bokeh
- pygal
- Plotly

> jy注: 其中Bokeh, pygal 以前没有用过, 所以其它四种会有一些自己的标注. 这些工具如果尚未安装可先使用pip install

在演示的例子中, 我会使用Pandas进行的数据处理(data manipulation) 然后进行图形可视化. 大多数操作不需要借助于pandas. 但我觉得pandas + 图形工具是数据业务中非常常见的操作方式. 也是这里最先介绍的原因.

Matplotlib

Matplotlib可说是Python绘图的鼻祖, 它的功能非常强大, 对于大多数人(包括翻译的我)只使用了它的简单皮毛却以为掌握了这项技能, 实则真是惭愧. 这里不会是一篇Matplotlib的教程, (后面还要花工夫来介绍Pandas, Seaborn), 也希望感兴趣的你多去看些其它的资料.

Methodology

希望快速掌握某个工具的人希望用最短的时间记住它的方法论, 有人会说 使用 xxxx 的方式 比你这样做更好. 但实际上这篇文章的目标不是把每个工具发挥到极致, 更多的像是一个让大家了解的目的. 至于进门之后走向哪条路, 则是看官您的决定了.

在演示的过程中, 我觉得可能最大的难点是绘图中对 横轴x 纵轴y 的formatting过程, 以及用图形给出二者之间的合理关系. 而且, 对于一个大样本的数据集, formatting在每种工具都是花精力的. 一旦这部分搞定了, 剩下的就是大同小异的操作了.

另一点要指出的是: barplot 或许是最为简单图形, 当然这些工具也不限于此. 下面的例子提供了formatting, label的操作演示. (译者注: blah blah ... 关于一些页面调整, 格式统一的描述, 不译了)

最后, 我还将会使用Excel作为对比性的工具. 我认为好的Python输出将会用过Excel, 不管是用在报告, 演示, 邮件沟通 以及静态网页展示上. 当然有些人还会提出反对的观点, 好吧, 那不再讨论范围了.

Data Set

http://pbpython.com/extras/mn-budget-detail-2014.csv (http://pbpython.com/extras/mn-budget-detail-2014.csv)

这是我们的分析数据集, 先了解了解Pandas的人就知道能借助read_csv来完成读取, 封成一个DataFrame.

> 这里考虑到网速问题, 我还是先下载再进行了加载 . 一个非常简单的数据集, 三列: 表示 类别 - 描述 - 数量, 因为只有一列定量数据, 2列定类数据, 那能做的事, 也就不多了.

In [8]:
```
#! /usr/bin/python
# coding: utf-8
# Thursday, December 17, 2015

from __future__ import print_function
import re
import sys
import os
import matplotlib as mpl
from matplotlib import pyplot as plt
from pandas import DataFrame, Series
import pandas as pd
import numpy as np

%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

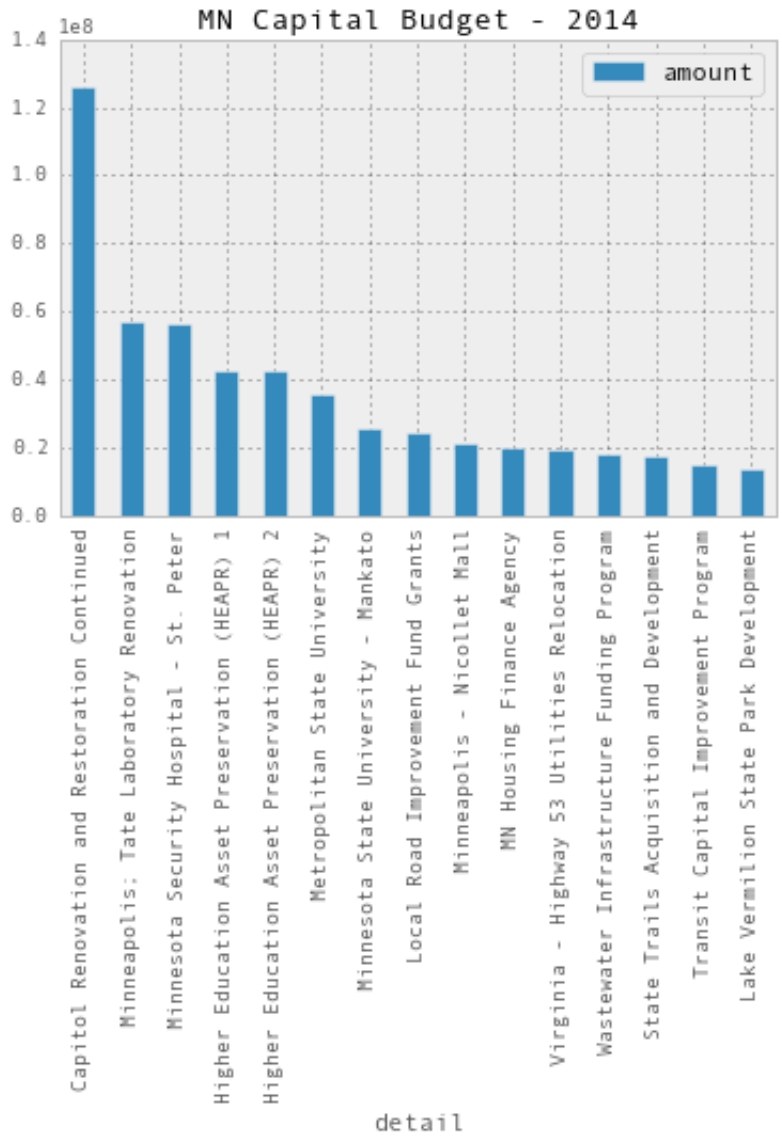
In [3]:
```
budget = pd.read_csv("~/Downloads/mn-budget-detail-2014.csv")

budget = budget.sort('amount', ascending=False)[:15]

budget.head(2)
```

Out[3]:

	category	detail	amount
46	ADMINISTRATION	Capitol Renovation and Restoration Continued	126300000
1	UNIVERSITY OF MINNESOTA	Minneapolis; Tate Laboratory Renovation	56700000

下面来直接使用pandas对象的plot方法来画个bar plot:

In [9]:
```
pd.options.display.mpl_style = 'default'
budget_plot = budget.plot(kind='bar', x=budget['detail'], title='MN
Capital Budget - 2014', legend=True)
```



从参数列表中看到我是用kind来指定图形类别, 指定了 x 轴使用的数据Label, Title, Legend等信息, 算是一个比较"重" 的图了.

保存图形到本地也非常方便, 用以下2行代码便能实现.

In [10]:
```
fig = budget_plot.get_figure()
fig.savefig("2014-mn-capital-budget.png")
```

这个简单例子暂时能让大家了解到Pandas作图的一个概念.

> 根据数据DataFrame来直接生成图形. 图形元素来自于各列 及 设定好的属性.

Seaborn

seaborn 可理解为在matplotlib基础之上, 使用pandas数据结构的更好的数据可视补充. 是由stanford 研究人员提出的. 因此, 对于数据分析人员是个相当友好的绘图项目.

项目地址在: seaborn (http://stanford.edu/~mwaskom/software/seaborn/introduction.html)

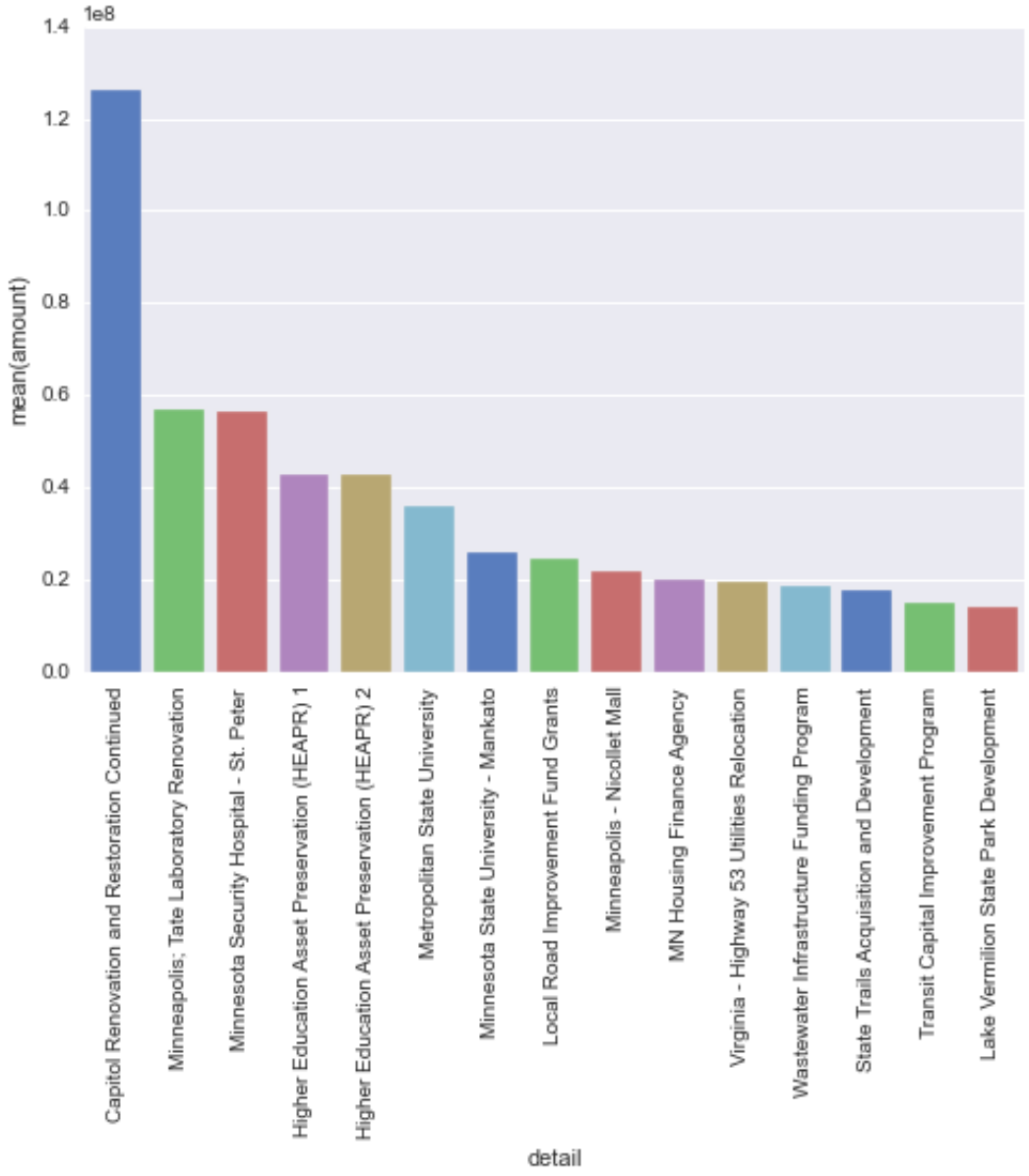
我这里的例子将会强调Seaborn提供的style设置的特点. 我们使用一些简单的color palette就能改变图形, 变得更"养眼". 因此, seaborn对一幅简单的图形就能做不少的操作.

标准的引入是这个样子的:

In [11]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [21]:
```
sns.set_style('darkgrid')
bar_plot = sns.barplot(data=budget, x='detail', y='amount', palette
='muted', order=budget['detail'].tolist())

plt.xticks(rotation=90)

# jy注: 在译者使用的Py3.4环境下 seaborn的 barplot方法 参数  x_order已经变更为 order
```

Out[21]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14
]),
 <a list of 15 Text xticklabel objects>)
```



大家能看到, 我这里使用了xticks的旋转参数, 对x轴的Label进行了旋转操作, 不然大家将会看到一幕 disaster

ggplot

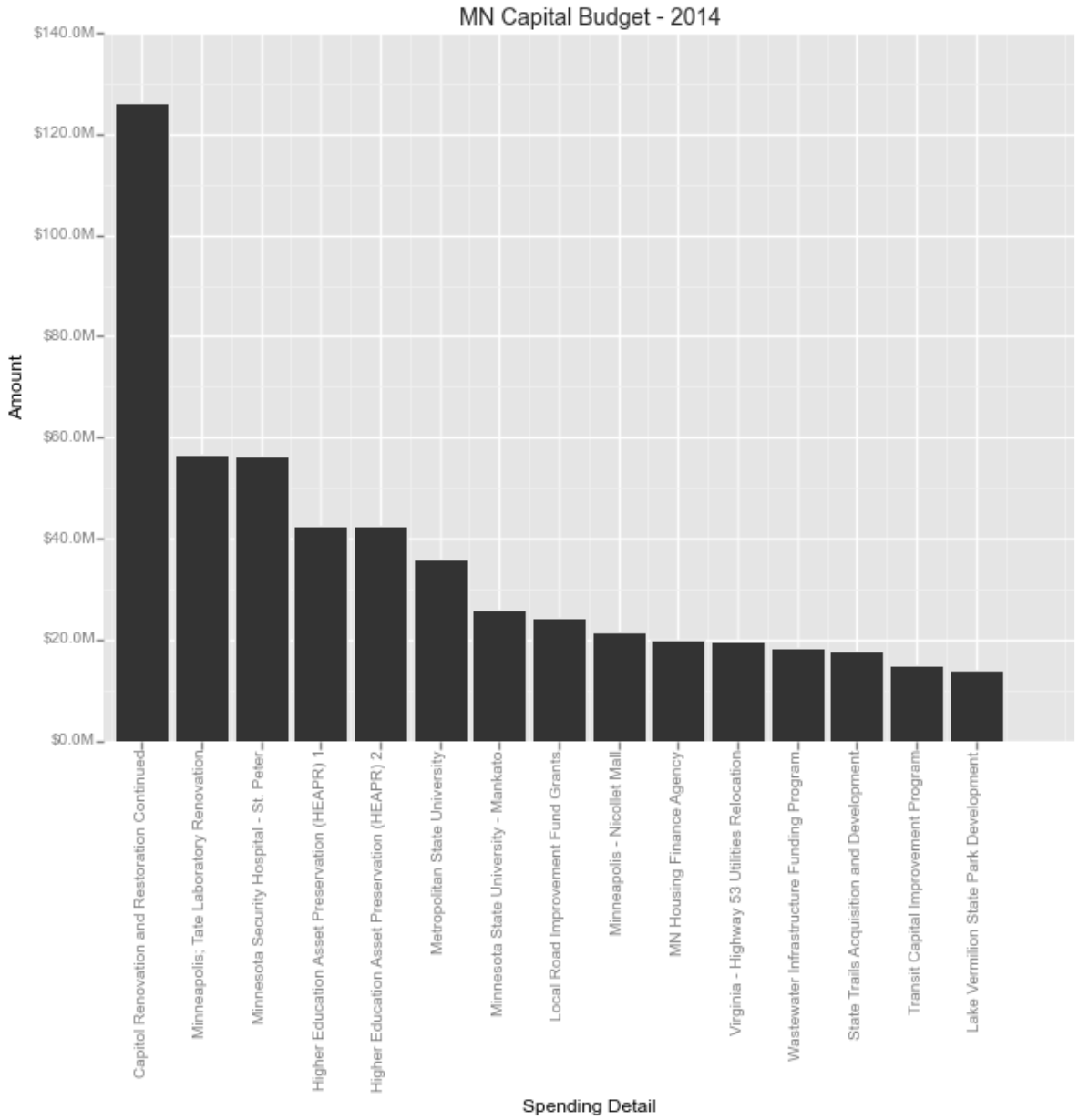
ggplot 同Seaborn 一样, 底层依赖于matplotlib, 也是为了让matplotlib作图更简单化. 与seaborn不同的是它是来自于R中ggplot2的一种Python实现. 所以有些R用户将会非常乐于见到如此.

虽然我没有用过R中的ggplot2 (译者注, 没关系, 还好我有点经验), 但尽管如此, 我也能看到ggpot包的强大之处. 我真希望这个库可以继续健壮发扬下去, 因为在我学习过程中, 它的有些特点是我刚开始还没领悟到的. 随着越来越多的积累, 才渐渐有种后知后觉. (当然也是在google的不少帮助下)

先来看这个例子:

In [23]:
```
import pandas as pd

from ggplot import *
```

In [26]:
```
p = ggplot(budget, aes(x='detail', y='amount')) + \
        geom_bar(stat='bar', labels=budget['detail'].tolist()) + \
        ggtitle('MN Capital Budget - 2014') + \
        xlab('Spending Detail') + \
        ylab('Amount') + scale_y_continuous(labels='millions') + \
        theme(axis_text_x=element_text(angle=90))

print(p)
```




```
<ggplot: (288940783)>
```

看上去 `print p` 用来输出图表是有点奇怪. 不过, 这其实是非常符合ggplot哲学的.

同上, 这里也对xlab进行了旋转90度. 对y纵轴做了百万级的格式, 让你读起图来更加直观.

保存起来很简单 使用:

In [27]:
```
ggsave(p, "mn-budget-capital-ggplot.png")
```

```
/Library/Frameworks/Python.framework/Versions/3.4/lib/python3.4/si
te-packages/ggplot/geoms/geom_bar.py:47: FutureWarning: comparison
to `None` will result in an elementwise object comparison in the f
uture.
  _reset = self.bottom == None or (self.ax != None and self.ax !=
ax)
Saving 11.0 x 8.0 in image.
```

Bokeh

Bokeh和之前三个工具不同, 区别是它不再依赖于matplotlib, 而且经常用来作为网页浏览器中的可视化. 作为交互式工具.

In [46]:
```
import pandas as pd

from bokeh.charts import Bar

data = dict(zip(details, amount))

print(data)
```

```
{'Higher Education Asset Preservation (HEAPR) 1': 42500000.0, 'Sta
te Trails Acquisition and Development': 17667000.0, 'Capitol Renov
ation and Restoration Continued': 126300000.0, 'Minneapolis - Nico
llet Mall': 21500000.0, 'Wastewater Infrastructure Funding Program
': 18333000.0, 'Transit Capital Improvement Program': 15000000.0,
'Local Road Improvement Fund Grants': 24356000.0, 'Higher Educatio
n Asset Preservation (HEAPR) 2': 42500000.0, 'MN Housing Finance A
gency': 20000000.0, 'Minnesota State University - Mankato': 258180
00.0, 'Metropolitan State University': 35865000.0, 'Minneapolis; T
ate Laboratory Renovation': 56700000.0, 'Minnesota Security Hospit
al - St. Peter': 56317000.0, 'Virginia - Highway 53 Utilities Relo
cation': 19500000.0, 'Lake Vermilion State Park Development': 1400
0000.0}
```

In [49]:
```
# details = budget['detail'].values.tolist()
# amount = list(budget['amount'].astype(float).values)

# bar = Bar(data, filename="bar.html")

# bar.title("MN - Capital Budget = 2014").xlabel("Detail").ylabel("
Amount")

# bar.show()

#> Bar 可能有点问题    -- 故此 先跳过了.
```

这里作图前就要做一些数据的处理

Pygal

Pygal用来创建 svg 图表, 如果相关依赖部署成功, 同样能完成文件的保存. svg 文件在处理交互图形时非常有用处的文件, 个人也觉得 Pygal 的效果是非常有吸引力的.

In [38]:
```
# 当前环境下没能安装成功, 先暂时跳过.
```

Out[38]:
```
['Capitol Renovation and Restoration Continued',
 'Minneapolis; Tate Laboratory Renovation',
 'Minnesota Security Hospital - St. Peter',
 'Higher Education Asset Preservation (HEAPR) 1',
 'Higher Education Asset Preservation (HEAPR) 2',
 'Metropolitan State University',
 'Minnesota State University - Mankato',
 'Local Road Improvement Fund Grants',
 'Minneapolis - Nicollet Mall',
 'MN Housing Finance Agency',
 'Virginia - Highway 53 Utilities Relocation',
 'Wastewater Infrastructure Funding Program',
 'State Trails Acquisition and Development',
 'Transit Capital Improvement Program',
 'Lake Vermilion State Park Development']
```

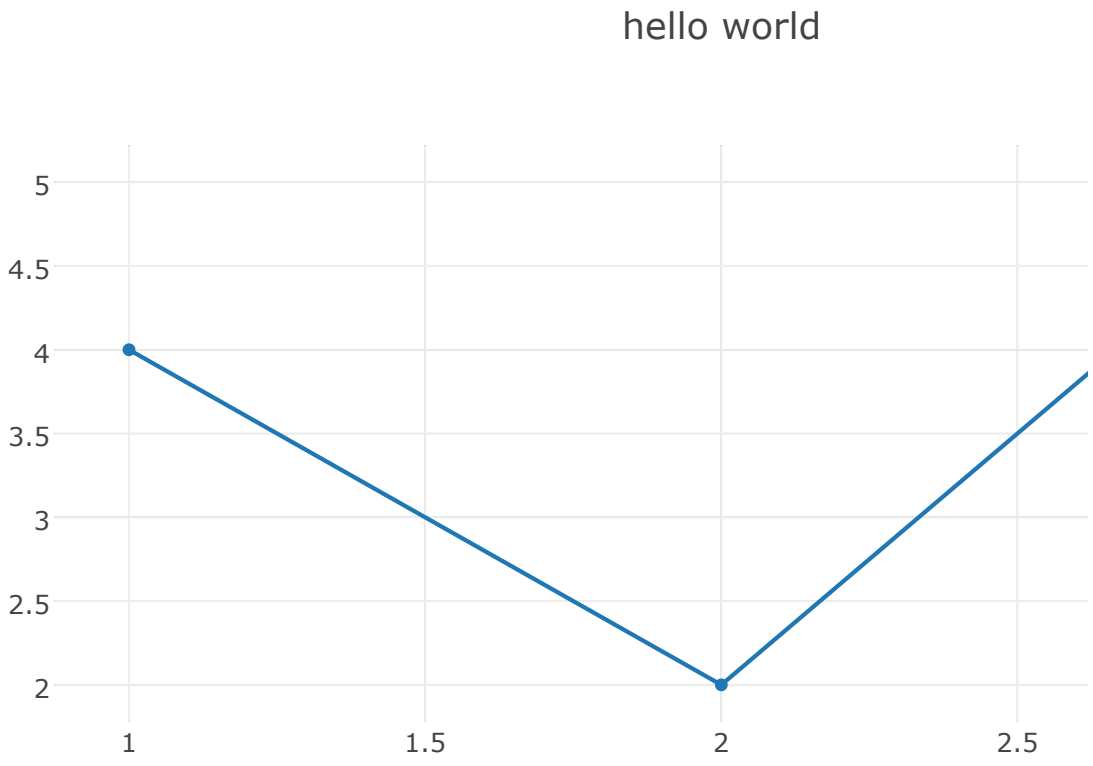
Plot.ly

plot.ly 是一个线上进行数据处理和作图的强大工具. 提供了多种API, Python是其中之一. 通过访问其官网, 便能查看API.

也许对一部分人而言, API的部署是花费时间的.

In [52]:
```
import plotly
print(plotly.__version__)           # version 1.9.x required
plotly.offline.init_notebook_mode() # run at the start of every not
ebook
plotly.offline.iplot({
    "data": [{
        "x": [1, 2, 3],
        "y": [4, 2, 5]
    }],
    "layout": {
        "title": "hello world"
    }
})
```

```
1.9.3
```



In [54]:
```
## converting a matplotlib graph into an interactive graph inside a
n IPython notebook

import matplotlib.pyplot as plt
import numpy as np
import plotly.plotly as py

import plotly.plotly as py
py.sign_in('staticora23d', 'y6qxwqeedy')
n = 50
x, y, z, s, ew = np.random.rand(5, n)
c, ec = np.random.rand(2, n, 4)
area_scale, width_scale = 500, 5

fig, ax = plt.subplots()
sc = ax.scatter(x, y, c=c,
                s=np.square(s)*area_scale,
                edgecolor=ec,
                linewidth=ew*width_scale)
ax.grid()

py.iplot_mpl(fig)
```

```
High five! You successfuly sent some data to your account on plotl
y. View your plot in your browser at https://plot.ly/~staticora23d
/0 or inside your plot.ly account where it is named 'plot from API
'
```

Out[54]:

In [55]:
```
## embedding a plotly graph inside an ipython notebook

import plotly.tools as tls

tls.embed("https://plot.ly/~streaming-demos/4")
```

Out[55]:

> jy注: 总体来说 从官网中提供的API 简单的 Get Start 就能看到 Plot.ly 作为一个商业化成型的工具 产出的结果是 so - beautiful. 值得一学. 而且提供的免费API key 还是能产出一些内容的.

有一点不一样的就是 产出结果刚刚是在 网页中的. 而非Python本原环境.

In [56]:
```
import plotly.plotly as py
import pandas as pd
from plotly.graph_objs import *


data = Data([
    Bar(
        x=budget["detail"],
        y=budget["amount"]
    )
])

layout = Layout(
    title='2014 MN Capital Budget',
    font=Font(
        family='Raleway, sans-serif'
    ),
    showlegend=False,
    xaxis=XAxis(
        tickangle=-45
    ),
    bargap=0.05
)

fig = Figure(data=data, layout=layout)
plot_url = py.plot(data,filename='MN Capital Budget - 2014')
py.image.save_as(fig, 'mn-14-budget.png')
```

Summary

Python系统里的作图说复杂也复杂, 说简单也简单.

good news: 我们有多种不同的工具去选择. 根据不同的场景来选择合适工具制图.

当然 也并不存在一个绝对好或者绝对坏的工具, 只能是因地制宜了.

以上几种工具的总体评价:

- Pandas: 在Pandas 触手可及的条件下就能快速作图, 因此是一种最为handable的工具.
- Seaborn: seaborn 提供更复杂化的方式 特别是在 color style方面, 是工程方向上的最佳方案;
- ggplot: 在R的基础 还是有非常promising的前途的;
- bokeh: 前端交互之选
- pygal: svg
- plotly: 同样拥有交互之选, 而且是web-based的输出效果最佳的方案.

> jy注: 后续我也会陆续整理 Seaborn, ggplot, plotly 在工作环境上的使用.

In [ ]: