MunichDataGeeks_Playing_with_data

Miguel Cabrera

November 25, 2013

1 Munich DataGeeks - Playing with the Meetup Data

Given that Datageeks Meetup is about cool things to do with data, let's see what we see if we do a bit of processing to the data that we have available.

Let's start by getting some nice defaults and setup some code needed. I based the style of the using the recommendations and code from the Harvard course on Data Science http://cs109.org/ - Totally recommended resource on learning both Data Science and how to do it with IPtyhon Notebooks.

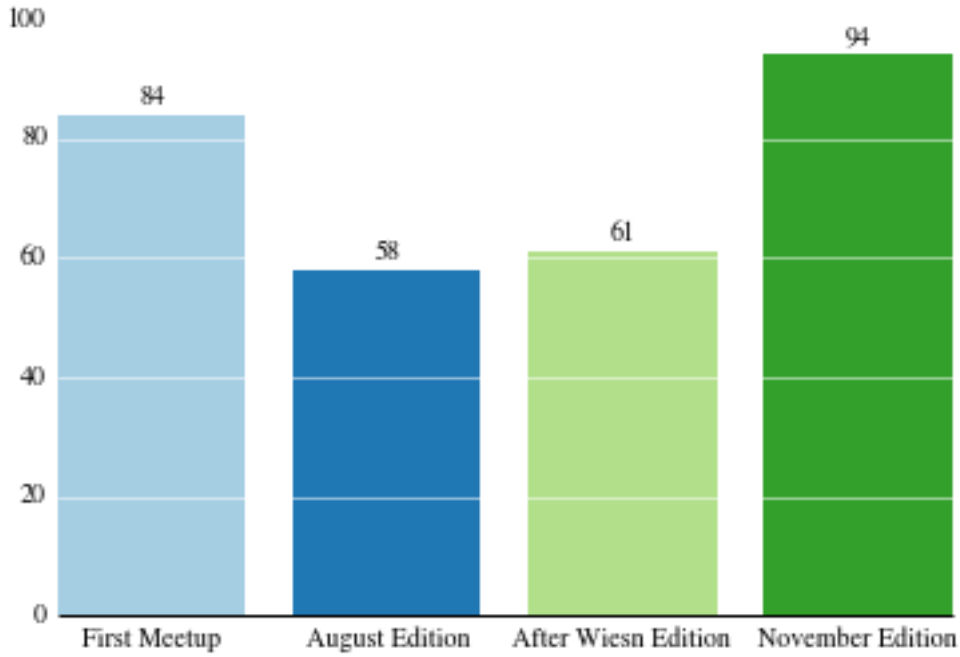
In [16]:
```
#import basic tools and change default colors

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib as mpl
import brewer2mpl
from matplotlib import rcParams

# Change the default colors
dark2_colors = brewer2mpl.get_map('Paired', 'Qualitative', 7).mpl_colors

rcParams['figure.figsize'] = (10, 6)
rcParams['figure.dpi'] = 200
rcParams['axes.color_cycle'] = dark2_colors
rcParams['lines.linewidth'] = 2
rcParams['axes.facecolor'] = 'white'
rcParams['font.size'] = 10
rcParams['patch.edgecolor'] = 'white'
rcParams['patch.facecolor'] = dark2_colors[0]
rcParams['font.family'] = 'StixGeneral'

%matplotlib inline
```

1.1 RSVP Comparisson

We have had already 4 meetups during 2013. Let's see the behaviour with this mega complex Visualization.

In [11]:
```
# Please don't make fun of my Python skills - nowadays it is not my primary programmin
labels = ['First Meetup', 'August Edition', 'After Wiesn Edition', 'November Edition']
numbers = [84,58,61,94]

x_pos = np.arange(len(labels))
box_colors =  brewer2mpl.get_map('Paired', 'Qualitative', 7).mpl_colors
#brewer2mpl.get_map('Set1', 'qualitative', 4).mpl_colors
```

```
plt.ylim([0,100])
plt.xticks(x_pos, labels)
ax = plt.subplot(111)

for (i,rsvp,color) in zip(x_pos,numbers,box_colors):
    ax.bar(i ,rsvp,align='center',color=color,linewidth=0)
    ax.annotate(r"%d" % rsvp,
                    (i, rsvp+1), va="bottom", ha="center")

# Remove top axes
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['left'].set_visible(False)

# remove ticks
ax.yaxis.set_ticks_position('none')
ax.xaxis.set_ticks_position('none')

ax.grid(axis = 'y', color ='white', linestyle='-')
plt.savefig('histogram',dpi=200)
plt.show()


plt.close()
```



1.2 Playing with Meetup.com API

It turns out you can get meta-info from the meetups via API. You just need a key to query it. So let's do the best you can do when you have data of users: *stalk people*.

So let's get the data:

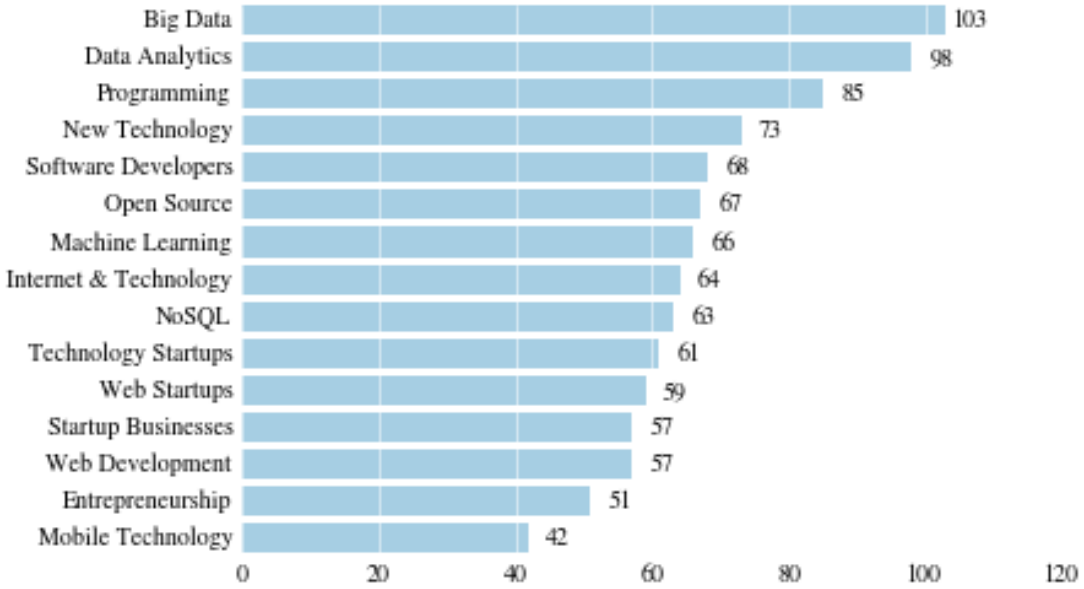
In [62]:
```
%%bash
curl "http://api.meetup.com/2/members?order=name&group_urlname=Munich-Datageeks&offset
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
                                 Dload  Upload   Total   Spent    Left
Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:--
--:--:--     0   0     0    0     0    0     0      0      0 --:--:--
0:00:01 --:--:--     0  18  259k   18 49341    0     0  22234       0
0:00:11  0:00:02  0:00:09 23905 100  259k  100  259k    0     0  95633
0  0:00:02  0:00:02 --:--:--   98k
```

In [19]:
```
#let's first create a nice function to  graph stuff again:
from chart_util import remove_border

def make_interest_count_graph(data_tuples):

    y_pos = np.arange(len(data_tuples))
    box_colors = brewer2mpl.get_map('Dark2', 'Qualitative', 7).mpl_colors
    #box_colors = brewer2mpl.get_map('Diverging', 'qualitative', 9).mpl_colors

    counts = [j  for  (i,j) in data_tuples]
    plt.yticks(y_pos, [i  for  (i,j) in data_tuples])


    ax = plt.subplot(111)

    for (i,count) in enumerate(counts):
        ax.barh(i ,count,align='center',linewidth=0)
        ax.annotate(str(count),
                    (count + 6 , i ), va="center", ha="right")

    remove_border(left=False, bottom=False)
    plt.grid(axis = 'x', color ='white', linestyle='-')

    plt.savefig('interests',dpi=300,bbox_inches='tight')
    plt.show()
    plt.close()
```

With the previously defined function we are going to write code to get the TOP-15 interests of the Datageeks community:

In [20]:
```
#let's not get fancy and download the data
import json
import operator
from collections import defaultdict

interest_counter = defaultdict(int)

#this is a rather big file but we can handle it
with open('members.json') as f:
    big_file = json.load(f)
    for user in big_file['results']:
        for interest in user['topics']:
            interest_counter[interest['name']] += 1


#Let's sort the list into a tuple
sorted_count = sorted(interest_counter.iteritems(),key=operator.itemgetter(1),reverse=
make_interest_count_graph(sorted_count[0:15][::-1])
```



1.3 Let's keep stalking people

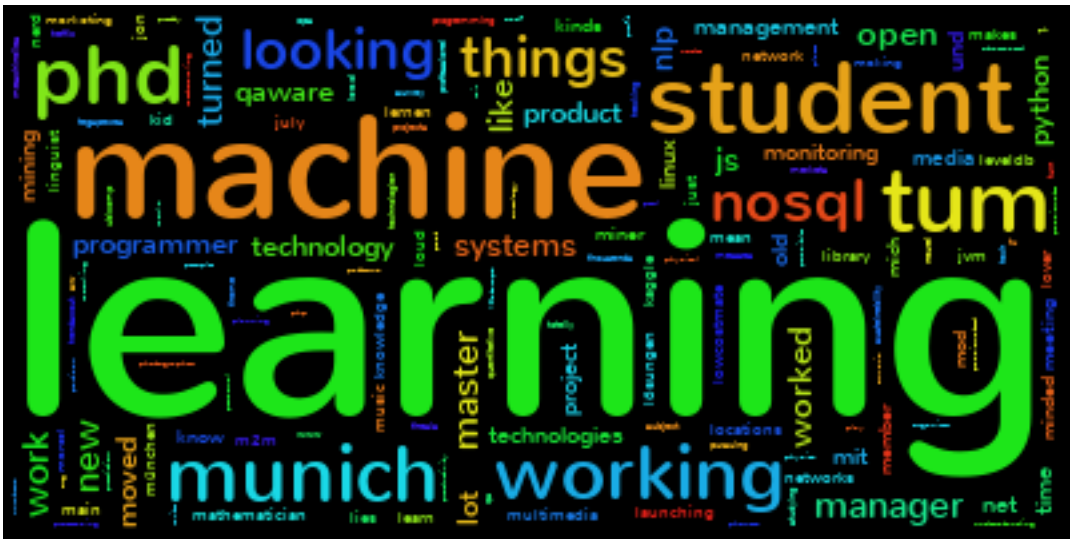
That was cool - nor particulary difficult but you would agree the bars looks cool :P I am quite interested in what people write on their bio. let's see if we can make a word cloud out of it and see if there are common terms.

In [2]:
```
# lets open the file again and count some words
# basically copied from http://peekaboo-vision.blogspot.de/2012/11/a-wordcloud-in-pyth
import sys
import numpy as np

sys.path.append('./word_cloud')
from wordcloud import make_wordcloud

import json
from sklearn.feature_extraction.text import CountVectorizer
from IPython.display import Image

bio_list = []
output_filename = "wordcloud.png"

with open('members.json') as f:
    big_file = json.load(f)
    for user in big_file['results']:
        if 'bio' in user:
            bio_list.append(user['bio'])

text  = ' '.join(bio_list)

cv = CountVectorizer(min_df=0, charset_error="ignore",
                     stop_words="english", max_features=150)
counts = cv.fit_transform([text]).toarray().ravel()


words = np.array(cv.get_feature_names())
#counts = counts / float(counts.max())

counts = make_wordcloud(words, counts, output_filename,font_path='/Users/miguel/Librar
```

```
Image(filename=output_filename)
```

```
/Users/miguel/anaconda/python.app/Contents/lib/python2.7/site-
packages/sklearn/feature_extraction/text.py:615: DeprecationWarning:
The charset_error parameter is deprecated as of version 0.14 and will
be removed in 0.16. Use decode_error instead.
  DeprecationWarning)
```

Out [2]:



1.4 Inspiration and Reference

Some interesting liks regarding visualization and data in general:

- https://drive.google.com/folderview?id=0BxYkKyLxfsNVd0xicUVDS1dIS0k&usp=sharing
- http://nbviewer.ipython.org/5357268
- http://nbviewer.ipython.org/urls/raw.github.com/cs109/content/master/lec_03_statistical_graphs.ipynb