Basic Analysis of Hacker News Posts

In [3]:
```
from csv import reader
hackernews = open('c:\\users\\daxto\\Jupyter DataSets\\hacker_news.csv', encod
hackernewsread = reader(hackernews)
hn = list(hackernewsread)
```

In [4]:
```
print(hn[:5])
```

```
[['id', 'title', 'url', 'num_points', 'num_comments', 'author', 'created_a
t'], ['12224879', 'Interactive Dynamic Video', 'http://www.interactivedynamic
video.com/', '386', '52', 'ne0phyte', '8/4/2016 11:52'], ['10975351', 'How to
Use Open Source and Shut the Fuck Up at the Same Time', 'http://hueniverse.co
m/2016/01/26/how-to-use-open-source-and-shut-the-fuck-up-at-the-same-time/',
'39', '10', 'josep2', '1/26/2016 19:30'], ['11964716', "Florida DJs May Face
Felony for April Fools' Water Joke", 'http://www.thewire.com/entertainment/20
13/04/florida-djs-april-fools-water-joke/63798/', '2', '1', 'vezycash', '6/2
3/2016 22:20'], ['11919867', 'Technology ventures: From Idea to Enterprise',
'https://www.amazon.com/Technology-Ventures-Enterprise-Thomas-Byers/dp/007352
3429', '3', '1', 'hswarna', '6/17/2016 0:01']]
```

Remove Headers and Verify

As seen above, the first row contains the headers of the data. This isn't data we are interested in analyzing so we will remove it in this step.

In [5]:
```
headers = hn[0]
hn = hn[1:]
```

In [6]:
```
print(hn[:5])
```

```
[['12224879', 'Interactive Dynamic Video', 'http://www.interactivedynamicvide
o.com/', '386', '52', 'ne0phyte', '8/4/2016 11:52'], ['10975351', 'How to Use
Open Source and Shut the Fuck Up at the Same Time', 'http://hueniverse.com/20
16/01/26/how-to-use-open-source-and-shut-the-fuck-up-at-the-same-time/', '3
9', '10', 'josep2', '1/26/2016 19:30'], ['11964716', "Florida DJs May Face Fe
lony for April Fools' Water Joke", 'http://www.thewire.com/entertainment/201
3/04/florida-djs-april-fools-water-joke/63798/', '2', '1', 'vezycash', '6/23/
2016 22:20'], ['11919867', 'Technology ventures: From Idea to Enterprise', 'h
ttps://www.amazon.com/Technology-Ventures-Enterprise-Thomas-Byers/dp/00735234
29', '3', '1', 'hswarna', '6/17/2016 0:01'], ['10301696', 'Note by Note: The
Making of Steinway L1037 (2007)', 'http://www.nytimes.com/2007/11/07/movies/0
7stein.html?_r=0', '8', '2', 'walterbell', '9/30/2015 4:12']]
```

Extracting Ask HN and Show HN Posts

In this project, we are interested only in posts beginning with 'Ask HN' or 'Show HN'. Below we will run a for loop to test if the title matches the criteria and then appends it to the appropriate

In [7]:
```
ask_posts = []
show_posts =[]
other_posts = []

for post in hn:
    title = post[1]
    if title.lower().startswith("ask hn"):
        ask_posts.append(post)
    elif title.lower().startswith("show hn"):
        show_posts.append(post)
    else:
        other_posts.append(post)

print(len(ask_posts))
print(len(show_posts))
print(len(other_posts))
```

```
1744
1162
17194
```

Calculating Average Number of Comments for Ask and Show Posts

In [8]:
```
total_ask_comments = 0

for row in ask_posts:
    comment = int(row[4])
    total_ask_comments += comment

avg_ask_comments = total_ask_comments / len(ask_posts)
print(avg_ask_comments)
```

```
14.038417431192661
```

In [9]:
```
total_show_comments = 0

for row in show_posts:
    comment = int(row[4])
    total_show_comments += comment

avg_show_comments = total_show_comments / len(show_posts)
print(avg_show_comments)
```

```
10.31669535283993
```

Results

On average, ask posts receive a higher comment count per post.

Analyzing Post Interaction by Time

A factor we would like to investigate more is the time of posting. This could potentially have an impact on the amount of interaction a post gets.

In [10]:
```
import datetime as dt
```

In [11]:
```
result_list = []

for row in ask_posts:
    result_list.append([row[6], int(row[4])])
```

For reference, `row[6]` is the value for the `created_at` time. `row[4]` is the number of comments on the post.

In [12]:
```
counts_by_hour = {}
comments_by_hour = {}
date_format = '%m/%d/%Y %H:%M'

for row in result_list:
    date = row[0]
    comments = int(row[1])
    time = dt.datetime.strptime(date, date_format)
    hour = time.strftime('%H')
    if hour in counts_by_hour:
        counts_by_hour[hour] += 1
        comments_by_hour[hour] += comments
    else:
        counts_by_hour[hour] = 1
        comments_by_hour[hour] = comments
```

Above we converted the date column into datetime objects where we could extract the hour in a consistent format. In addition, we created dictionaries that counted the number of comments and sum of comments per hour.

In [13]:
```
print(comments_by_hour)
print('\n')
print(counts_by_hour)
```

```
{'09': 251, '13': 1253, '10': 793, '14': 1416, '16': 1814, '23': 543, '12': 6
87, '17': 1146, '15': 4477, '21': 1745, '20': 1722, '02': 1381, '18': 1439,
'03': 421, '05': 464, '19': 1188, '01': 683, '22': 479, '08': 492, '04': 337,
'00': 447, '06': 397, '07': 267, '11': 641}


{'09': 45, '13': 85, '10': 59, '14': 107, '16': 108, '23': 68, '12': 73, '1
7': 100, '15': 116, '21': 109, '20': 80, '02': 58, '18': 109, '03': 54, '05':
46, '19': 110, '01': 60, '22': 71, '08': 48, '04': 47, '00': 55, '06': 44, '0
7': 34, '11': 58}
```

Above is a check on the two lists that we have to help us understand what math is required to find the average. Below this we find the average and put it into a list of lists which displays the average number of comments per post in a given hour.

In [14]:
```
# avg is calculated by comments / counts
# result is list of lists where first element is hour and second is average nu

avg_by_hour = []

for hourtime in comments_by_hour:
    avg_by_hour.append([hourtime, comments_by_hour[hourtime] / counts_by_hour[

avg_by_hour
```

Out[14]:
```
[['09', 5.5777777777777775],
 ['13', 14.741176470588234],
 ['10', 13.440677966101696],
 ['14', 13.233644859813085],
 ['16', 16.796296296296298],
 ['23', 7.985294117647059],
 ['12', 9.41095890410959],
 ['17', 11.46],
 ['15', 38.5948275862069],
 ['21', 16.009174311926607],
 ['20', 21.525],
 ['02', 23.810344827586206],
 ['18', 13.20183486238532],
 ['03', 7.796296296296297],
 ['05', 10.08695652173913],
 ['19', 10.8],
 ['01', 11.383333333333333],
 ['22', 6.746478873239437],
 ['08', 10.25],
 ['04', 7.170212765957447],
 ['00', 8.127272727272727],
 ['06', 9.022727272727273],
 ['07', 7.852941176470588],
 ['11', 11.051724137931034]]
```

After creating the list, we need to put it into a format that is more readable and condusive to decision making.

In [15]:
```
swap_avg_by_hour = []
```

In [16]:
```
for row in avg_by_hour:
    swap_avg_by_hour.append([row[1], row[0]])
```

In [17]:
```
sorted_swap = sorted(swap_avg_by_hour, reverse=True)
sorted_swap
```

Out[17]:
```
[[38.5948275862069, '15'],
 [23.810344827586206, '02'],
 [21.525, '20'],
 [16.796296296296298, '16'],
 [16.009174311926607, '21'],
 [14.741176470588234, '13'],
 [13.440677966101696, '10'],
 [13.233644859813085, '14'],
 [13.20183486238532, '18'],
 [11.46, '17'],
 [11.383333333333333, '01'],
 [11.051724137931034, '11'],
 [10.8, '19'],
 [10.25, '08'],
 [10.08695652173913, '05'],
 [9.41095890410959, '12'],
 [9.022727272727273, '06'],
 [8.127272727272727, '00'],
 [7.985294117647059, '23'],
 [7.852941176470588, '07'],
 [7.796296296296297, '03'],
 [7.170212765957447, '04'],
 [6.746478873239437, '22'],
 [5.5777777777777775, '09']]
```

In [18]:
```
print('Top 5 Hours for Ask Posts Comments')

for avg, hr in sorted_swap[:5]:
    time = dt.datetime.strptime(hr,'%H').strftime('%H:%M')
    print(
        '{} receives {:.2f} average comments per post'.format(time,avg)
    )
```

```
Top 5 Hours for Ask Posts Comments
15:00 receives 38.59 average comments per post
02:00 receives 23.81 average comments per post
20:00 receives 21.52 average comments per post
16:00 receives 16.80 average comments per post
21:00 receives 16.01 average comments per post
```

Results

At the end of this project we were able to display the data in a way that is much more readable and sort for the highest average interaction numbers by hour. As shown, Posts at 15:00 have a much higher average comment per post than other hours by a larger margin from 1st to 2nd than any of the other gaps.

If you were looking to post an article and desired high interaction, we would be comfortable making the recommendation to post around 3 p.m. as it could give you a small extra boost in your numbers.