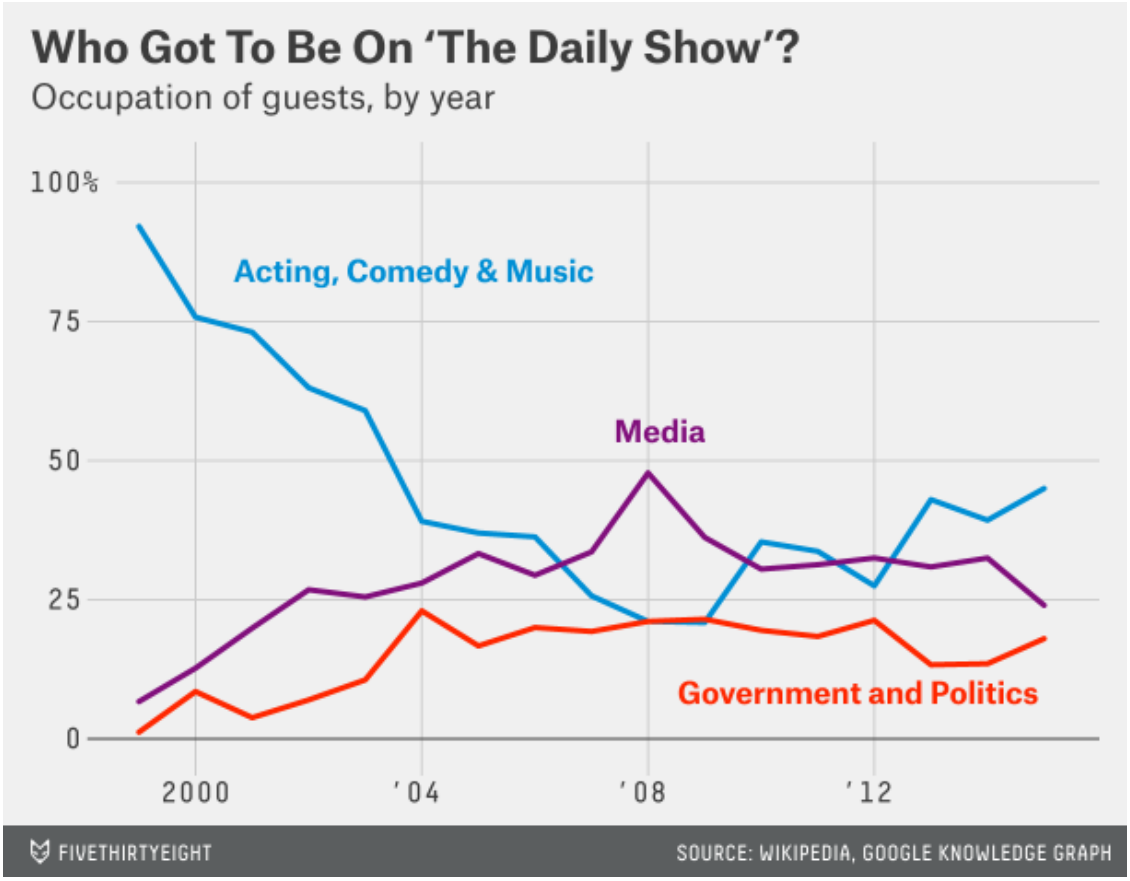
Mimicking Daily Show's Visualization

February 20, 2022

This visualization demo is an attempt to mimic a visualization on Jon Stewart's guests on The Daily Show. https://fivethirtyeight.com/features/every-guest-jon-stewart-ever-had-on-the-daily-show/

[1]:
```
from IPython.display import Image
Image("hickey-datalab-dailyshow.png")
```

[1]:



0.1 Imports

[2]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import matplotlib.ticker as mtick
from matplotlib.dates import DateFormatter
import pandas as pd
import numpy as np
```

0.2 Load Data

[3]:
```
guests = pd.read_csv('daily_show_guests.csv')
```

[4]:
```
guests
```

[4]:

	YEAR	GoogleKnowlege_Occupation	Show	Group	Raw_Guest_List
0	1999	actor	1/11/99	Acting	Michael J. Fox
1	1999	Comedian	1/12/99	Comedy	Sandra Bernhard
2	1999	television actress	1/13/99	Acting	Tracey Ullman
3	1999	film actress	1/14/99	Acting	Gillian Anderson
4	1999	actor	1/18/99	Acting	David Alan Grier
…	…	…	…	…	…
2688	2015	biographer	7/29/15	Media	Doris Kearns Goodwin
2689	2015	director	7/30/15	Media	J. J. Abrams
2690	2015	stand-up comedian	8/3/15	Comedy	Amy Schumer
2691	2015	actor	8/4/15	Acting	Denis Leary
2692	2015	comedian	8/5/15	Comedy	Louis C.K.

[2693 rows x 5 columns]

0.3 Data Wrangling

[5]:
```
grouped2 = guests.groupby(['YEAR','Group'])
grouped2 = grouped2.count().reset_index().
 ↪drop(columns=['GoogleKnowlege_Occupation','Raw_Guest_List']).
 ↪rename(columns={'Show':'count','YEAR':'year','Group':'group'})
grouped2
```

[5]:

	year	group	count
0	1999	Acting	108
1	1999	Comedy	25
2	1999	Media	11
3	1999	Musician	17
4	1999	Politician	2
..	…	…	…
219	2015	Misc	3
220	2015	Musician	5

```
221  2015  Political Aide      3
222  2015      Politician     14
223  2015         Science      1

[224 rows x 3 columns]
```

[6]:
```
conditions = [
    (grouped2['group'] == 'Acting'),
    (grouped2['group'] == 'Comedy'),
    (grouped2['group'] == 'Musician'),
    (grouped2['group'] == 'Media'),
    (grouped2['group'] == 'Government'),
    (grouped2['group'] == 'Politician')
]

values = [1,1,1,2,3,3]

grouped2['gid'] = np.select(conditions,values)
```

[7]:
```
grouped2
```

[7]:
```
     year           group  count  gid
0    1999          Acting    108    1
1    1999          Comedy     25    1
2    1999           Media     11    2
3    1999        Musician     17    1
4    1999      Politician      2    3
..    …               …    …  …
219  2015            Misc      3    0
220  2015        Musician      5    1
221  2015  Political Aide      3    0
222  2015      Politician     14    3
223  2015         Science      1    0

[224 rows x 4 columns]
```

[8]:
```
grouped2 = grouped2.groupby(['year','gid']).agg({'count':'sum'}).reset_index()
grouped2
```

[8]:
```
     year  gid  count
0    1999    1    150
1    1999    2     11
2    1999    3      2
3    2000    0      6
4    2000    1    125
..    …  …    …
62   2014    3     19
```

```
63  2015    0     17
64  2015    1     45
65  2015    2     24
66  2015    3     14

[67 rows x 3 columns]
```

[9]:
```
t = grouped2.groupby('year').agg({'count':'sum'}).reset_index()
t
```

[9]:
```
    year  count
0   1999    163
1   2000    165
2   2001    156
3   2002    157
4   2003    159
5   2004    161
6   2005    162
7   2006    160
8   2007    140
9   2008    161
10  2009    163
11  2010    164
12  2011    163
13  2012    160
14  2013    165
15  2014    163
16  2015    100
```

[10]:
```
grouped3 = grouped2.merge(t, on='year').rename(columns={'count_x':
 ↪'countid','count_y':'sum'})
grouped3
```

[10]:
```
    year  gid  countid  sum
0   1999    1      150  163
1   1999    2       11  163
2   1999    3        2  163
3   2000    0        6  165
4   2000    1      125  165
..   …  …       …  …
62  2014    3       19  163
63  2015    0       17  100
64  2015    1       45  100
65  2015    2       24  100
66  2015    3       14  100

[67 rows x 4 columns]
```

[11]:
```
grouped4 = grouped3.groupby('year').apply(lambda x: 100*x/x.sum())
grouped4['year'] = grouped3['year']
grouped4['gid'] = grouped3['gid']
grouped4 = grouped4.drop(columns=['sum']).rename(columns={'countid':'prop'})
grouped4
```

[11]:

	year	gid	prop
0	1999	1	92.024540
1	1999	2	6.748466
2	1999	3	1.226994
3	2000	0	3.636364
4	2000	1	75.757576
..	…	…	…
62	2014	3	11.656442
63	2015	0	17.000000
64	2015	1	45.000000
65	2015	2	24.000000
66	2015	3	14.000000

[67 rows x 3 columns]

[12]:
```
conditions = [
    (grouped4['gid'] == 0),
    (grouped4['gid'] == 1),
    (grouped4['gid'] == 2),
    (grouped4['gid'] == 3)
]

values = ["Other","Acting, Comedy & Music","Media","Government and Politics"]

grouped4['gid_str'] = np.select(conditions,values)
```

[13]:
```
grouped4 = grouped4[grouped4['gid'] != 0]
grouped4
```

[13]:

	year	gid	prop	gid_str
0	1999	1	92.024540	Acting, Comedy & Music
1	1999	2	6.748466	Media
2	1999	3	1.226994	Government and Politics
4	2000	1	75.757576	Acting, Comedy & Music
5	2000	2	12.727273	Media
6	2000	3	7.878788	Government and Politics
8	2001	1	73.076923	Acting, Comedy & Music
9	2001	2	19.230769	Media
10	2001	3	3.205128	Government and Politics
12	2002	1	63.057325	Acting, Comedy & Music
13	2002	2	24.840764	Media

```
14  2002    3   5.732484  Government and Politics
16  2003    1  58.490566   Acting, Comedy & Music
17  2003    2  25.786164                   Media
18  2003    3  10.062893  Government and Politics
20  2004    1  39.130435   Acting, Comedy & Music
21  2004    2  27.950311                   Media
22  2004    3  21.739130  Government and Politics
24  2005    1  37.037037   Acting, Comedy & Music
25  2005    2  33.333333                   Media
26  2005    3  14.197531  Government and Politics
28  2006    1  36.250000   Acting, Comedy & Music
29  2006    2  29.375000                   Media
30  2006    3  17.500000  Government and Politics
32  2007    1  25.714286   Acting, Comedy & Music
33  2007    2  33.571429                   Media
34  2007    3  15.714286  Government and Politics
36  2008    1  21.118012   Acting, Comedy & Music
37  2008    2  47.826087                   Media
38  2008    3  16.770186  Government and Politics
40  2009    1  20.858896   Acting, Comedy & Music
41  2009    2  36.196319                   Media
42  2009    3  19.018405  Government and Politics
44  2010    1  35.365854   Acting, Comedy & Music
45  2010    2  30.487805                   Media
46  2010    3  17.073171  Government and Politics
48  2011    1  33.742331   Acting, Comedy & Music
49  2011    2  30.674847                   Media
50  2011    3  15.950920  Government and Politics
52  2012    1  27.500000   Acting, Comedy & Music
53  2012    2  32.500000                   Media
54  2012    3  20.000000  Government and Politics
56  2013    1  43.030303   Acting, Comedy & Music
57  2013    2  30.909091                   Media
58  2013    3  10.909091  Government and Politics
60  2014    1  39.263804   Acting, Comedy & Music
61  2014    2  32.515337                   Media
62  2014    3  11.656442  Government and Politics
64  2015    1  45.000000   Acting, Comedy & Music
65  2015    2  24.000000                   Media
66  2015    3  14.000000  Government and Politics
```

0.4 Data Gathering and Plotting

[14]:
```
sns.set_context("talk")
```

[16]:
```
fig = sns.lineplot(data=grouped4, x="year", y="prop", hue="gid_str", palette =␣
↪['deepskyblue','darkorchid','orangered'])
```

```
fig.set(xlabel=None,ylabel=None,title="Who Got To Be On 'The Daily Show'?")
fig.yaxis.set_major_formatter(mtick.PercentFormatter())
fig.set_xticklabels(['\'{:g}'.format(x%100) for x in fig.get_xticks()])
fig.set_ylim(0,100)
fig.legend(title=None)
```

```
C:\Users\aKost\AppData\Local\Temp/ipykernel_12840/394409180.py:4: UserWarning:
FixedFormatter should only be used together with FixedLocator
  fig.set_xticklabels(['\'{:g}'.format(x%100) for x in fig.get_xticks()])
```

[16]: <matplotlib.legend.Legend at 0x23265705d00>