Random Numbers with Numpy

In [1]:
```
!date
```

```
Wed 31 Jul 2024 12:30:52 PM EDT
```

In [84]:
```
from IPython import display
display.Image('/home/harlohutch77/pictures/python-powered.gif', height='400', width='800')
```

Out[84]:
```
<IPython.core.display.Image object>
```

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import cufflinks as cf
%matplotlib inline
```

In [2]:
```
random_np = np.random.rand(10,10)
```

In [3]:
```
#print(random_np)
random_np
```

Out[3]:
```
array([[0.00635504, 0.0490324 , 0.76110284, 0.87170052, 0.33780623,
        0.99419889, 0.76980762, 0.91130703, 0.35662454, 0.36161909],
       [0.72575937, 0.56225131, 0.79370178, 0.9185687 , 0.60201769,
        0.11240611, 0.42262567, 0.49467631, 0.55551128, 0.95792147],
       [0.14968013, 0.6910054 , 0.61501196, 0.03605317, 0.495974  ,
        0.87464075, 0.97543406, 0.61356411, 0.8798132 , 0.93777962],
       [0.36980428, 0.989536  , 0.22049018, 0.35263687, 0.36459099,
        0.09411022, 0.30468607, 0.16166835, 0.18236617, 0.23583817],
       [0.86491284, 0.41474606, 0.90840017, 0.46707026, 0.33301829,
        0.66021698, 0.32786698, 0.74108378, 0.33650237, 0.90784347],
       [0.80572714, 0.83699729, 0.49500351, 0.94061493, 0.99650686,
        0.89940936, 0.69421902, 0.88943489, 0.91762509, 0.72743447],
       [0.14812814, 0.28671956, 0.86702516, 0.78816146, 0.17239631,
        0.74738439, 0.60486512, 0.96040521, 0.97237063, 0.92277847],
       [0.82265362, 0.57085957, 0.28155722, 0.79523545, 0.99661183,
        0.78002487, 0.56503064, 0.93101637, 0.90114526, 0.19940371],
       [0.80743802, 0.85924607, 0.80996799, 0.48594322, 0.3719352 ,
        0.26770949, 0.62143178, 0.86894491, 0.62158801, 0.98995063],
       [0.7374486 , 0.4398042 , 0.52052676, 0.70860086, 0.50469669,
        0.11066127, 0.31267052, 0.09692034, 0.4621508 , 0.92833155]])
```

Created a dataset in Numpy and dataframe in Pandas with irrational numbers

In [4]:
```
random_pd = pd.DataFrame(np.random.rand(10,10))
random_pd
```

Out[4]:

	0	1	2	3	4	5	6	7	8	9
0	0.151768	0.492096	0.390975	0.478734	0.430338	0.382600	0.044891	0.521525	0.548217	0.383386
1	0.507693	0.752845	0.033231	0.968689	0.820568	0.591919	0.257081	0.093309	0.163772	0.675155
2	0.885118	0.737906	0.330337	0.123447	0.303592	0.513245	0.352208	0.342806	0.016411	0.036564
3	0.193544	0.985897	0.908139	0.553818	0.544846	0.267044	0.318080	0.060226	0.457795	0.131288
4	0.100034	0.829839	0.949033	0.252547	0.976440	0.555777	0.202846	0.336933	0.072127	0.103741
5	0.798860	0.230012	0.771341	0.385104	0.580586	0.578666	0.538413	0.136657	0.728548	0.271262
6	0.814118	0.844419	0.666889	0.012693	0.026513	0.016331	0.206016	0.976696	0.192592	0.976684
7	0.487722	0.437924	0.251286	0.205329	0.073067	0.910053	0.468173	0.969283	0.227627	0.784670
8	0.195025	0.528089	0.433629	0.353929	0.527884	0.817469	0.579965	0.042813	0.478103	0.369838
9	0.149255	0.456746	0.137945	0.578408	0.715358	0.654252	0.019916	0.611223	0.672685	0.160349

In [5]:
```
#printing out the type and lenth of the dataset
print(type(len(random_pd)))
print(len(random_pd))
```

```
&lt;class 'int'&gt;
10
```

Statistics on random Numpy irrational numbers - decimals

In [7]:
```
random_pd.describe()
```

Out[7]:

	0	1	2	3	4	5	6	7	8	9
count	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000
mean	0.428314	0.629577	0.487280	0.391270	0.499919	0.528736	0.298759	0.409147	0.355788	0.389294
std	0.312038	0.234666	0.320410	0.273416	0.304651	0.259139	0.191538	0.354165	0.253276	0.320131
min	0.100034	0.230012	0.033231	0.012693	0.026513	0.016331	0.019916	0.042813	0.016411	0.036564
25%	0.162212	0.465584	0.271049	0.217134	0.335279	0.415261	0.203639	0.104146	0.170977	0.138553
50%	0.341374	0.632997	0.412302	0.369516	0.536365	0.567222	0.287581	0.339869	0.342711	0.320550
75%	0.726068	0.810591	0.745228	0.535047	0.681665	0.638669	0.439182	0.588798	0.530688	0.602213
max	0.885118	0.985897	0.949033	0.968689	0.976440	0.910053	0.579965	0.976696	0.728548	0.976684

In [8]:
```
random_pd.sum()
```

Out[8]:
```
0    4.283136
1    6.295773
2    4.872804
3    3.912698
4    4.999192
5    5.287358
6    2.987591
7    4.091472
8    3.557877
9    3.892936
dtype: float64
```

In [9]:
```
random_pd.max()
```

Out[9]:
```
0    0.885118
1    0.985897
2    0.949033
3    0.968689
4    0.976440
5    0.910053
6    0.579965
7    0.976696
8    0.728548
9    0.976684
dtype: float64
```

In [10]:
```
random_pd.min()
```

Out[10]:
```
0    0.100034
1    0.230012
2    0.033231
3    0.012693
4    0.026513
5    0.016331
6    0.019916
7    0.042813
8    0.016411
9    0.036564
dtype: float64
```

In [11]:
```
random_pd.std()
```

Out[11]:
```
0    0.312038
1    0.234666
2    0.320410
3    0.273416
4    0.304651
5    0.259139
6    0.191538
7    0.354165
8    0.253276
9    0.320131
dtype: float64
```

In [12]:
```
random_pd.count()
```

Out[12]:
```
0    10
1    10
2    10
3    10
4    10
5    10
6    10
7    10
8    10
9    10
dtype: int64
```

In [13]:
```
random_pd.corr()
```

Out[13]:

	0	1	2	3	4	5	6	7	8	9
0	1.000000	-0.086122	-0.067745	-0.385776	-0.533824	-0.230855	0.359844	0.200589	-0.286429	0.325134
1	-0.086122	1.000000	0.323389	-0.019933	0.073136	-0.576911	-0.260221	-0.099795	-0.624346	-0.017976
2	-0.067745	0.323389	1.000000	-0.397555	0.105548	-0.450983	0.167630	-0.191706	0.032066	-0.275813
3	-0.385776	-0.019933	-0.397555	1.000000	0.627221	0.183548	-0.222118	-0.536209	0.306189	-0.102304
4	-0.533824	0.073136	0.105548	0.627221	1.000000	0.216110	-0.213867	-0.661767	0.136523	-0.561173
5	-0.230855	-0.576911	-0.450983	0.183548	0.216110	1.000000	0.439364	-0.141929	0.100043	-0.142346
6	0.359844	-0.260221	0.167630	-0.222118	-0.213867	0.439364	1.000000	-0.329444	-0.002343	0.064734
7	0.200589	-0.099795	-0.191706	-0.536209	-0.661767	-0.141929	-0.329444	1.000000	-0.194567	0.579138
8	-0.286429	-0.624346	0.032066	0.306189	0.136523	0.100043	-0.002343	-0.194567	1.000000	-0.201578
9	0.325134	-0.017976	-0.275813	-0.102304	-0.561173	-0.142346	0.064734	0.579138	-0.201578	1.000000

This syntax allows me to see the first column [0] in an array

In [14]:
```
random_pd[0].unique()
```

Out[14]:
```
array([0.15176844, 0.50769295, 0.8851185 , 0.19354353, 0.10003412,
       0.79885957, 0.81411754, 0.48772189, 0.1950253 , 0.14925452])
```

This syntax allows me to see how many rows are in column [0]

In [15]:
```
random_pd[0].nunique()
```

Out[15]:
```
10
```

This syntax is adding all the columns together and placing the total in the sum_all column..

In [17]:
```
random_pd['sum_all'] = random_pd[0]  + random_pd[1] +  random_pd[2] + random_pd[3] + random_pd[4] + random_pd[5
+ random_pd[6] + random_pd[7] + random_pd[8] + random_pd[9]
random_pd
```

Out[17]:

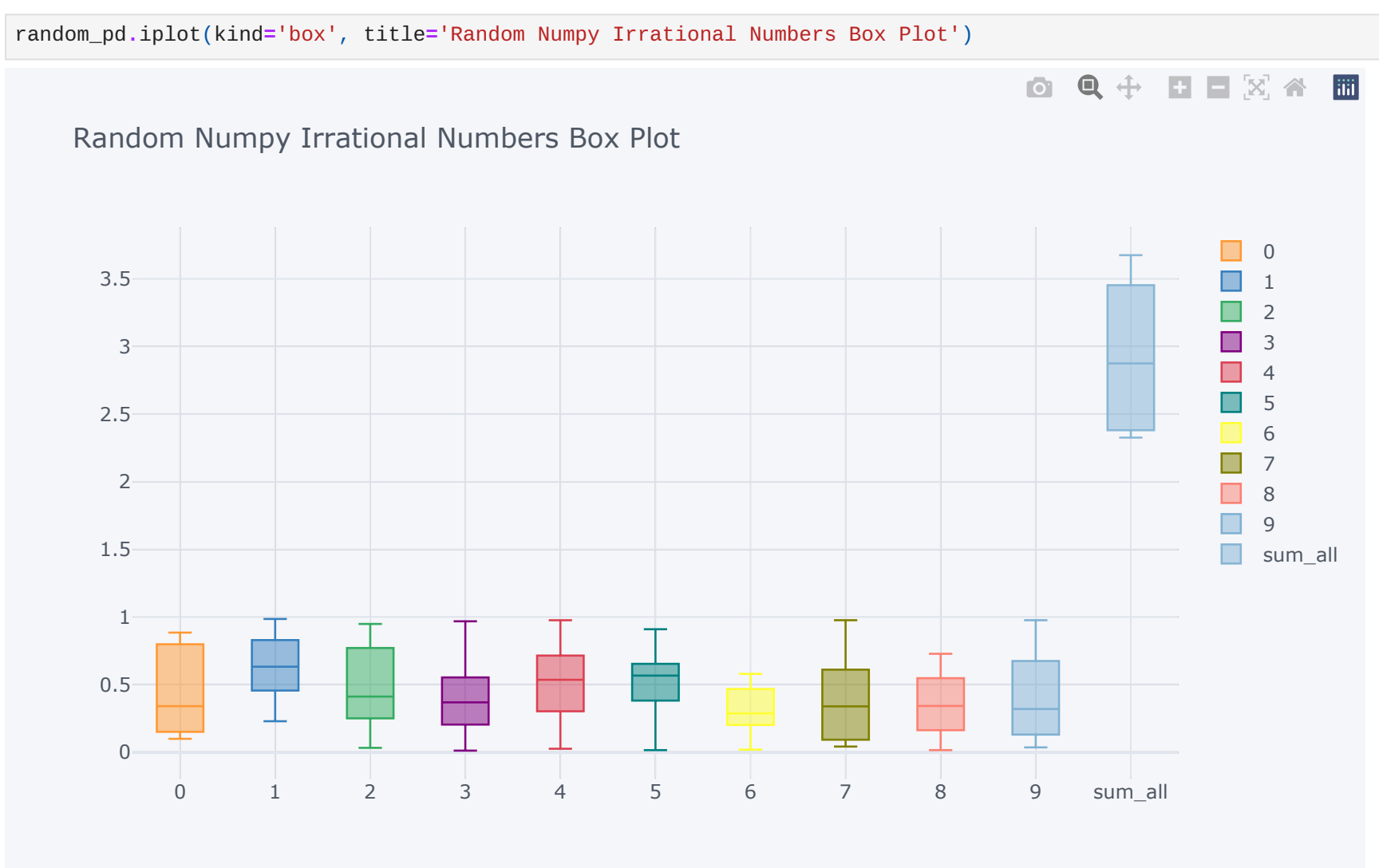
	0	1	2	3	4	5	6	7	8	9	sum_all
0	0.151768	0.492096	0.390975	0.478734	0.430338	0.382600	0.044891	0.521525	0.548217	0.383386	2.326512
1	0.507693	0.752845	0.033231	0.968689	0.820568	0.591919	0.257081	0.093309	0.163772	0.675155	3.674944
2	0.885118	0.737906	0.330337	0.123447	0.303592	0.513245	0.352208	0.342806	0.016411	0.036564	2.893645
3	0.193544	0.985897	0.908139	0.553818	0.544846	0.267044	0.318080	0.060226	0.457795	0.131288	3.453288
4	0.100034	0.829839	0.949033	0.252547	0.976440	0.555777	0.202846	0.336933	0.072127	0.103741	3.663671
5	0.798860	0.230012	0.771341	0.385104	0.580586	0.578666	0.538413	0.136657	0.728548	0.271262	3.344569
6	0.814118	0.844419	0.666889	0.012693	0.026513	0.016331	0.206016	0.976696	0.192592	0.976684	2.380962
7	0.487722	0.437924	0.251286	0.205329	0.073067	0.910053	0.468173	0.969283	0.227627	0.784670	2.365382
8	0.195025	0.528089	0.433629	0.353929	0.527884	0.817469	0.579965	0.042813	0.478103	0.369838	2.856025
9	0.149255	0.456746	0.137945	0.578408	0.715358	0.654252	0.019916	0.611223	0.672685	0.160349	2.691964

This syntax is so I may use Cufflinks in my jupyter_notebook

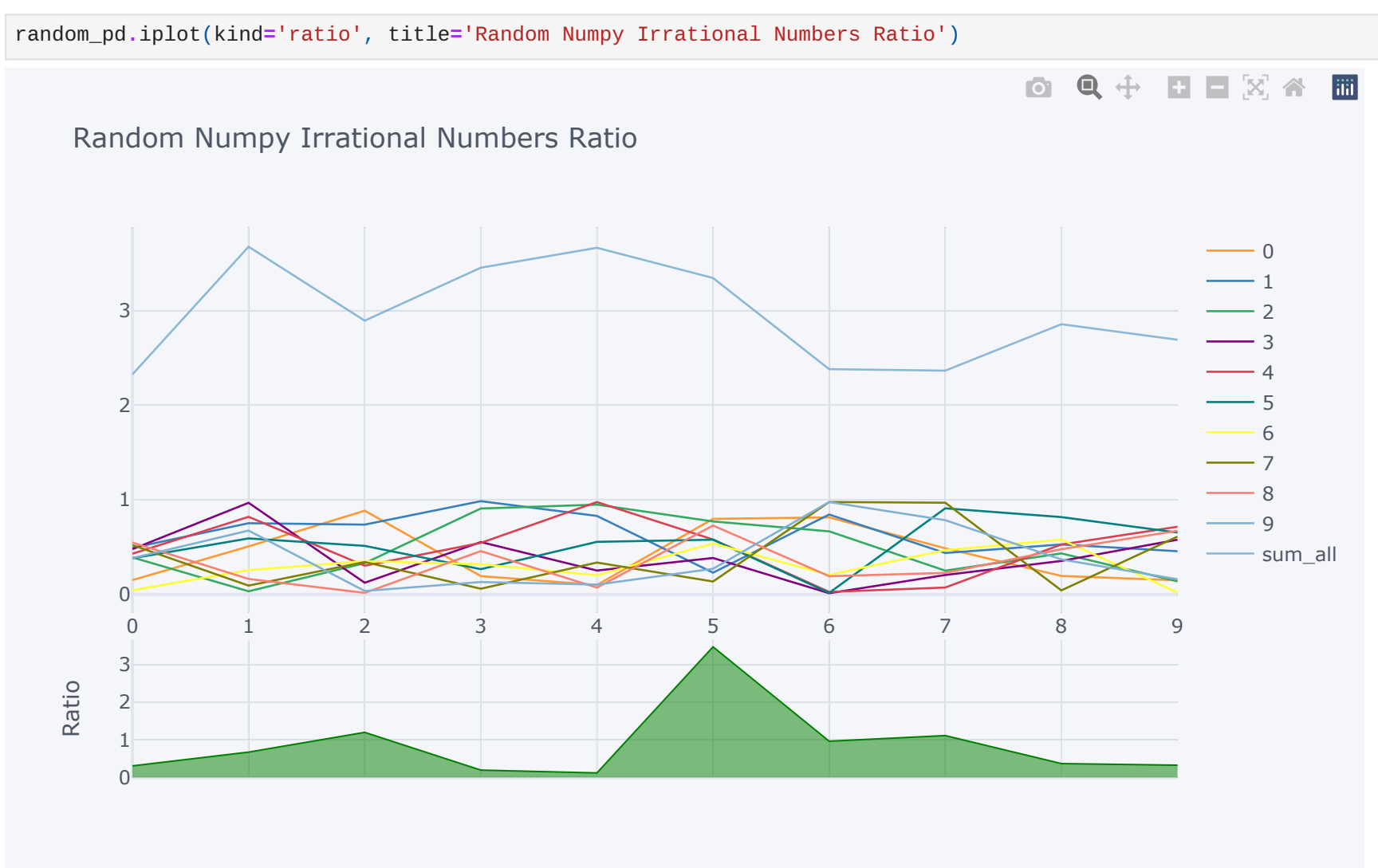
In [18]:
```
from plotly import __version__
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot

print(__version__)
import cufflinks as cf
cf.go_offline()
```

```
5.7.0
```

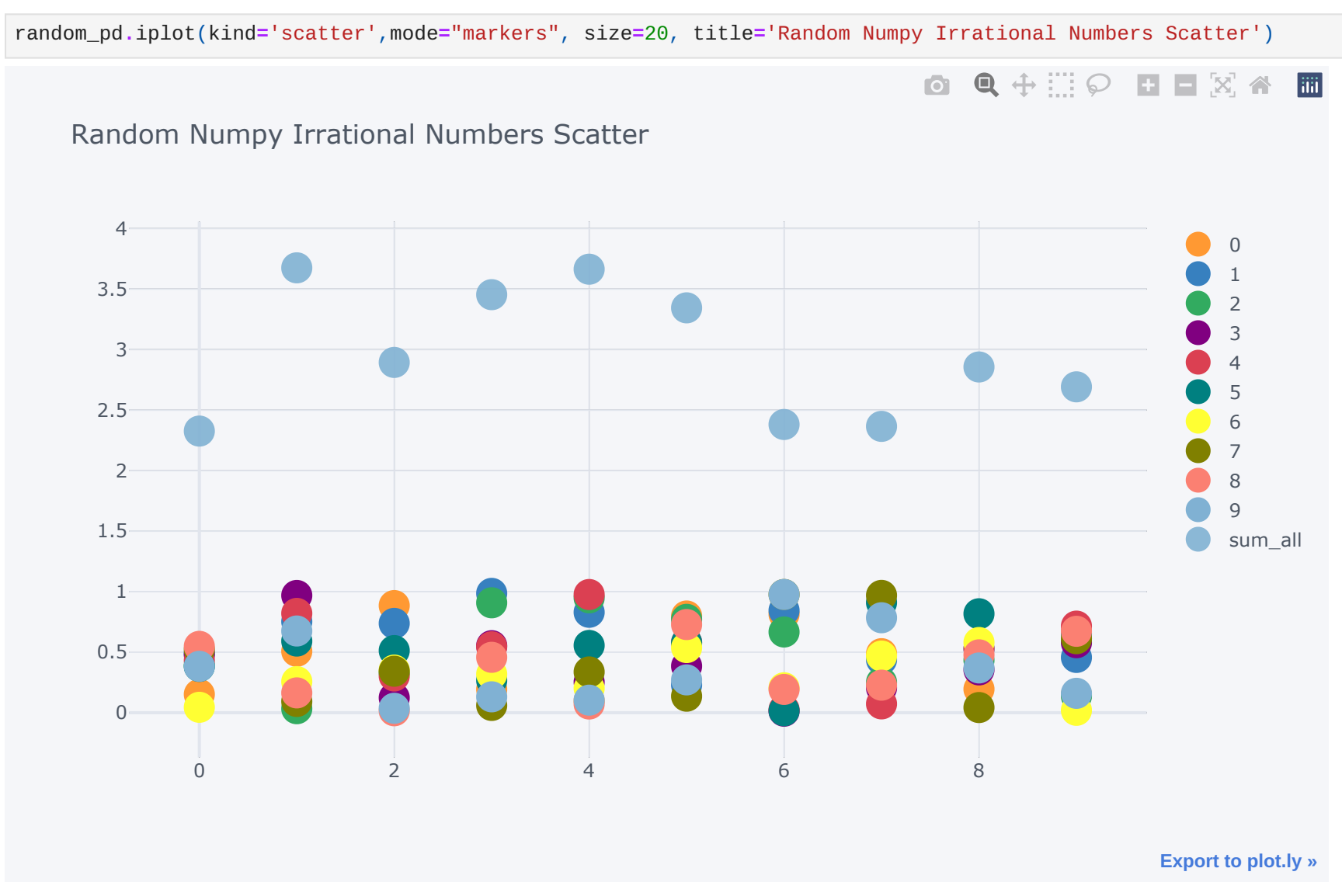
In [19]:
```
random_pd.iplot(kind='box', title='Random Numpy Irrational Numbers Box Plot')
```



In [21]:
```
random_pd.iplot(kind='ratio', title='Random Numpy Irrational Numbers Ratio')
```



In [22]:
```
random_pd.iplot(kind='scatter',mode="markers", size=20, title='Random Numpy Irrational Numbers Scatter')
```



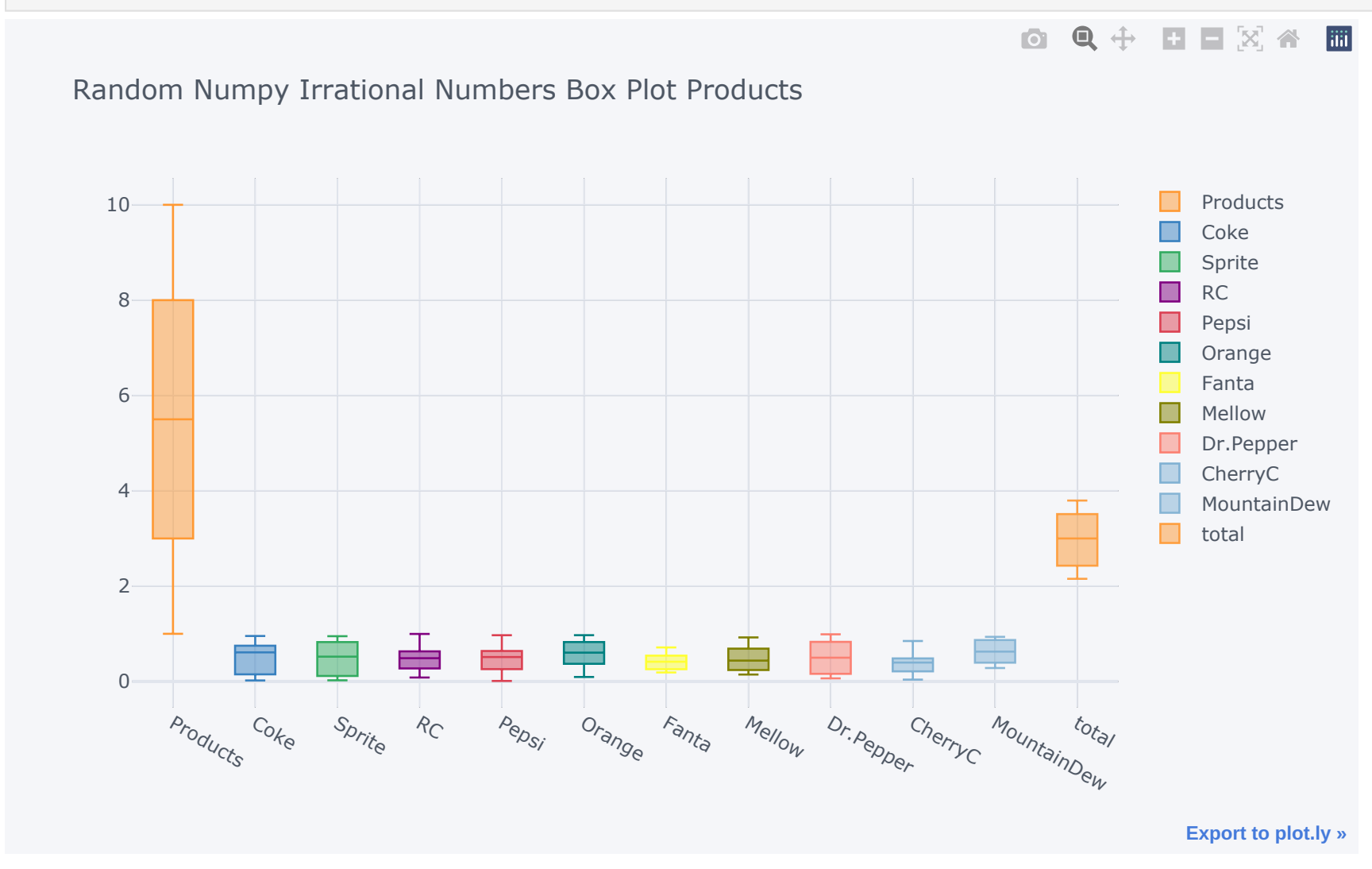
I made the random numbers into a csv - I implemented ficticious products for column names

In [23]:
```
ran = pd.read_csv('/home/harlohutch77/random_pd.csv')
ran
```

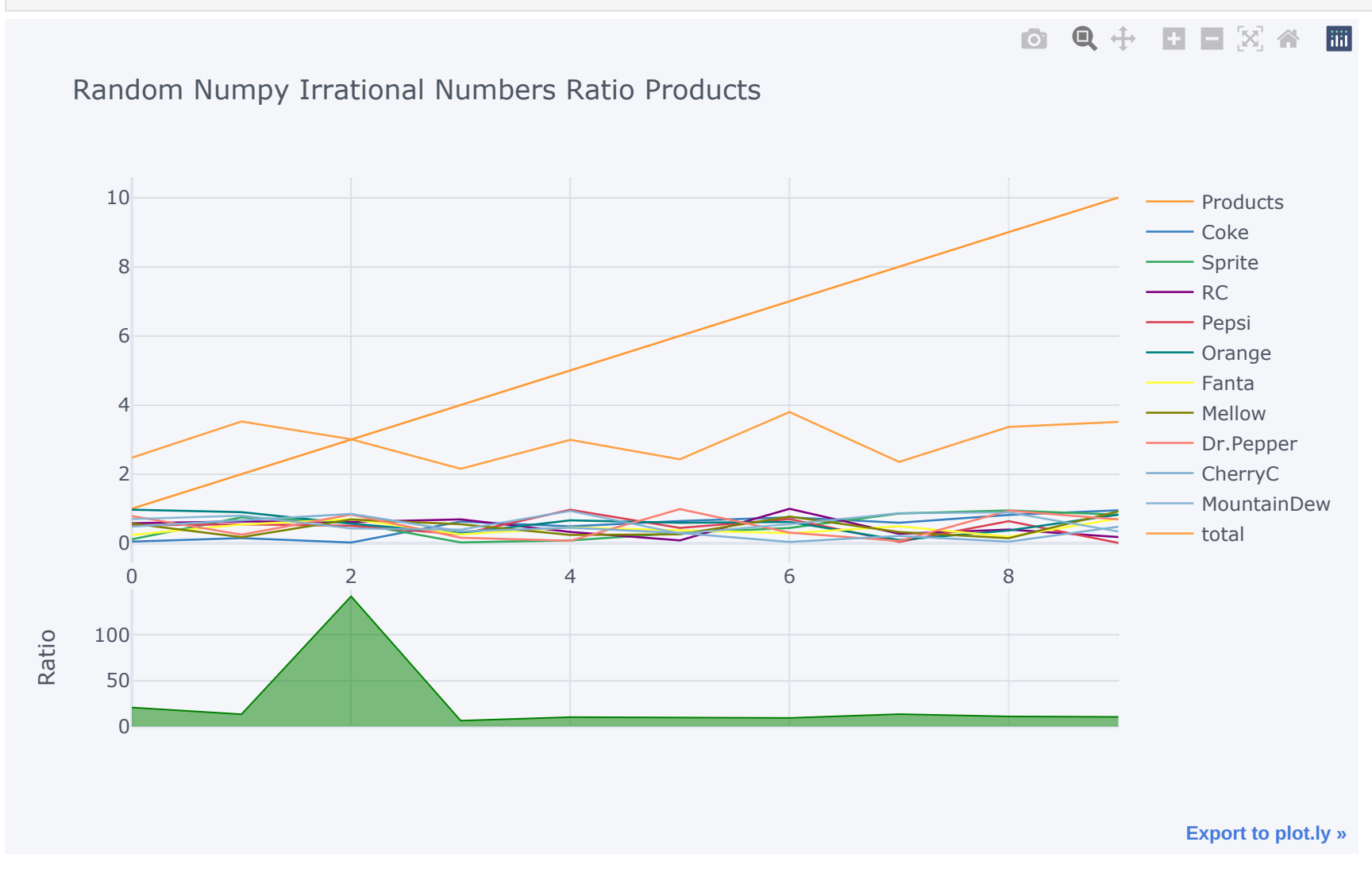
Out[23]:

	Products	Coke	Sprite	RC	Pepsi	Orange	Fanta	Mellow	Dr.Pepper	CherryC	MountainDew	total
0	1	0.048277	0.115401	0.579283	0.519849	0.971133	0.242123	0.570446	0.786197	0.483752	0.699466	2.476064
1	2	0.149523	0.751276	0.631731	0.547993	0.900539	0.541722	0.176662	0.248823	0.664372	0.799924	3.522784
2	3	0.021145	0.599164	0.615859	0.505455	0.567262	0.699447	0.688694	0.830987	0.848947	0.429452	3.008332
3	4	0.627663	0.024364	0.690763	0.258879	0.294591	0.257521	0.550780	0.160410	0.344160	0.394360	2.153782
4	5	0.489216	0.082435	0.326978	0.970392	0.663406	0.459601	0.240440	0.071870	0.449891	0.936301	2.992026
5	6	0.643989	0.296370	0.082906	0.444029	0.592958	0.368867	0.258838	0.989333	0.297146	0.282153	2.429120
6	7	0.750233	0.442865	0.997067	0.698682	0.616461	0.290601	0.773342	0.307834	0.038450	0.553325	3.795909
7	8	0.593524	0.861231	0.273619	0.039832	0.094168	0.490929	0.325019	0.065362	0.212536	0.869822	2.353302
8	9	0.819742	0.949817	0.396920	0.639358	0.370415	0.189395	0.145672	0.943339	0.045729	0.894330	3.365645
9	10	0.954198	0.826348	0.179130	0.010900	0.827295	0.714059	0.923986	0.689863	0.478951	0.339999	3.511930

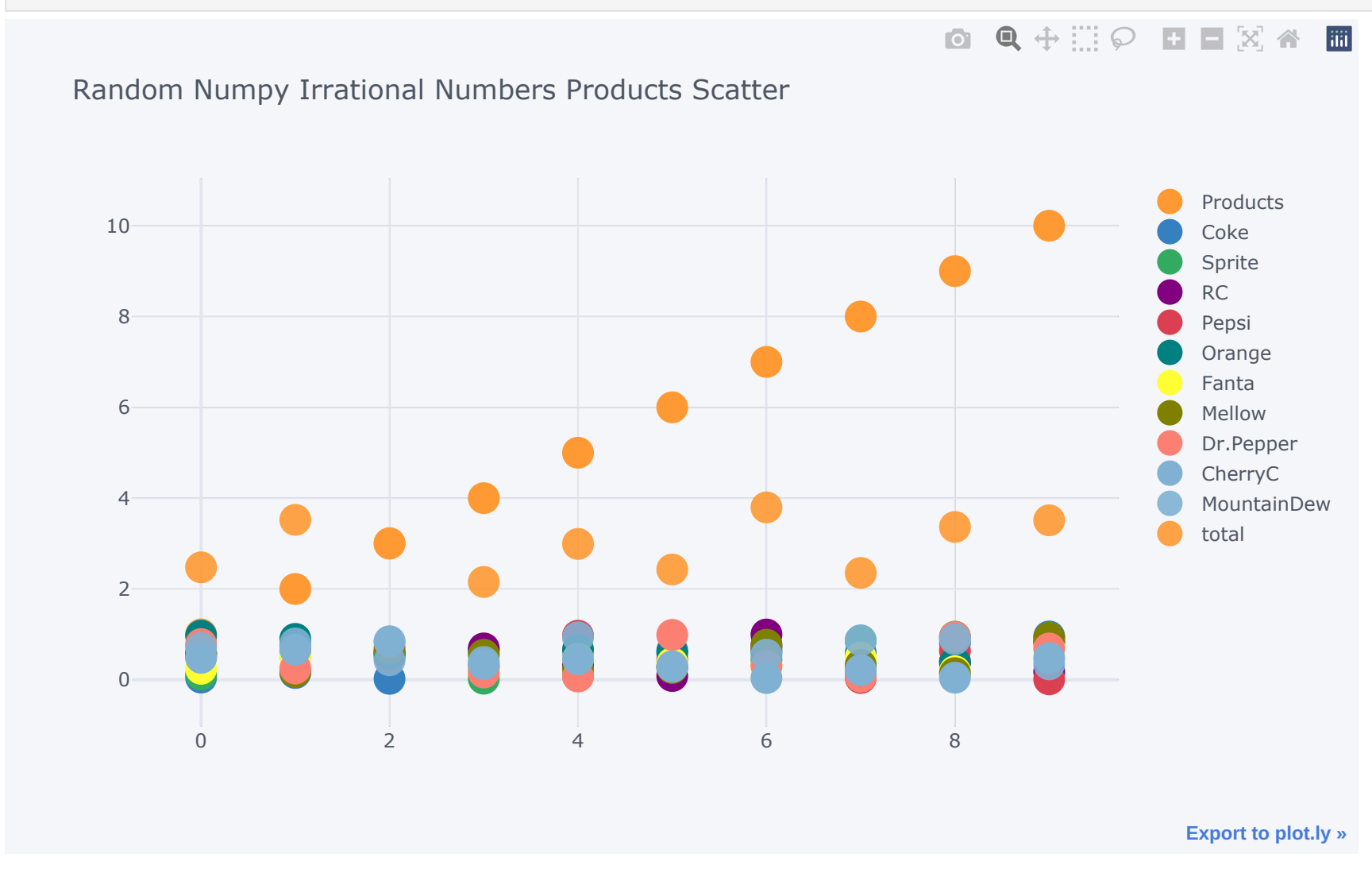
In [24]:
```
ran.iplot(kind='box', title='Random Numpy Irrational Numbers Box Plot Products')
```



In [25]:
```
ran.iplot(kind='ratio', title='Random Numpy Irrational Numbers Ratio Products')
```



In [26]:
```
ran.iplot(kind='scatter',mode="markers", size=20, title='Random Numpy Irrational Numbers Products Scatter ')
```



In [86]:
```
from IPython import display
display.Image('/home/harlohutch77/pictures/python-powered.gif', height='400', width='800')
```

Out[86]:
```
<IPython.core.display.Image object>
```