run_length

Unknown Author

January 09, 2014

Part I

Data Analysis with Pandas in IPython Notebook

`%matplotlib inline` is an IPython “magic” function that allows figures to be displayed in the notebook.

```
In [18]: %matplotlib inline
         import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
```

Reading csv data into a convienient structure with pandas. Pandas includes some intergration with the IPython notebook such that the output can be automatically formatted in HTML.

```
In [19]: data = pd.read_csv("../input/run_length_data.csv")
         data.head()
```

```
            208     132   209207   209144   2098582
Out [19]: 0   3.8790  1.0344  0.6465  5.9478   2.1981
          1   1.2930  1.0344  0.6465  1.8102   2.5860
          2   2.4567  1.0344  0.6465  0.7758   2.3274
          3   1.9395  0.5172  0.7758  3.4911   1.1637
          4   1.4223  2.3274  0.9051  7.1115   4.9134
```

Summary statistics:

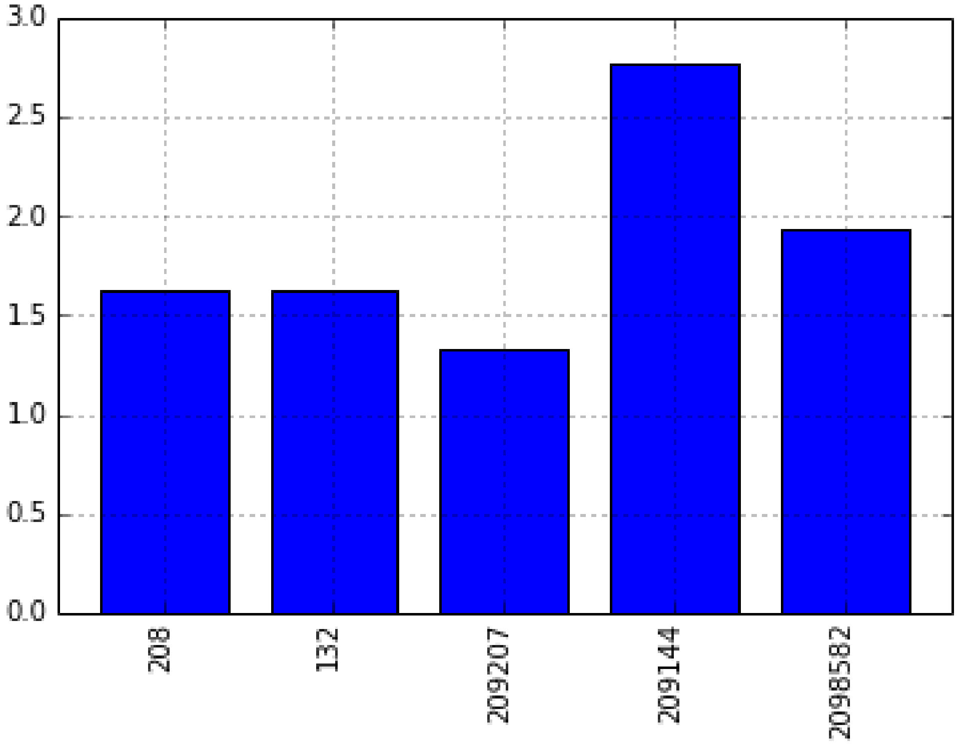
```
In [20]: data.describe()
```

```
                     208         132      209207      209144     2098582
Out [20]: count  129.000000  129.000000  122.000000  123.000000  133.000000
          mean     1.629781    1.627777    1.325855    2.771015    1.930750
          std      1.083059    1.134904    0.909222    2.411553    1.196032
          min      0.517200    0.517200    0.517200    0.517200    0.517200
          25%      0.775800    0.775800    0.775800    1.034400    1.034400
          50%      1.163700    1.293000    1.034400    1.939500    1.680900
          75%      2.327400    1.939500    1.551600    3.685050    2.456700
          max      5.172000    6.982200    5.947800   10.214700    5.818500
```

The plotting tools built into pandas allow for quick graphical views of the data.

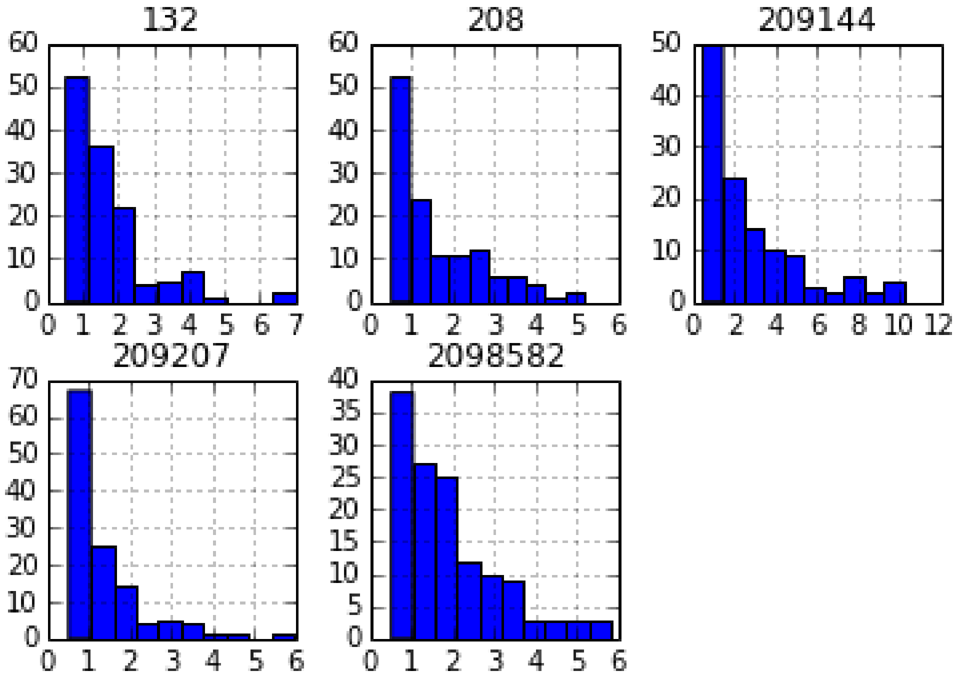
In [21]:

```
data.mean().plot(kind='bar');
```



In [22]:

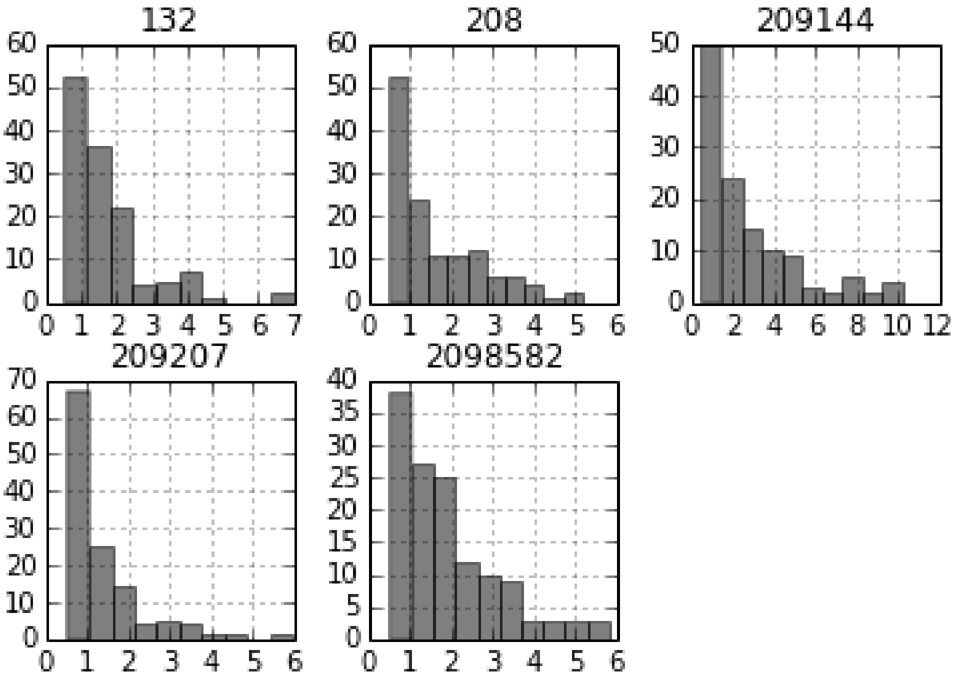
```
data.hist();
```



The plots are customizable.

In [23]:

```
data.hist(color='k', alpha=0.5, bins=10);
```

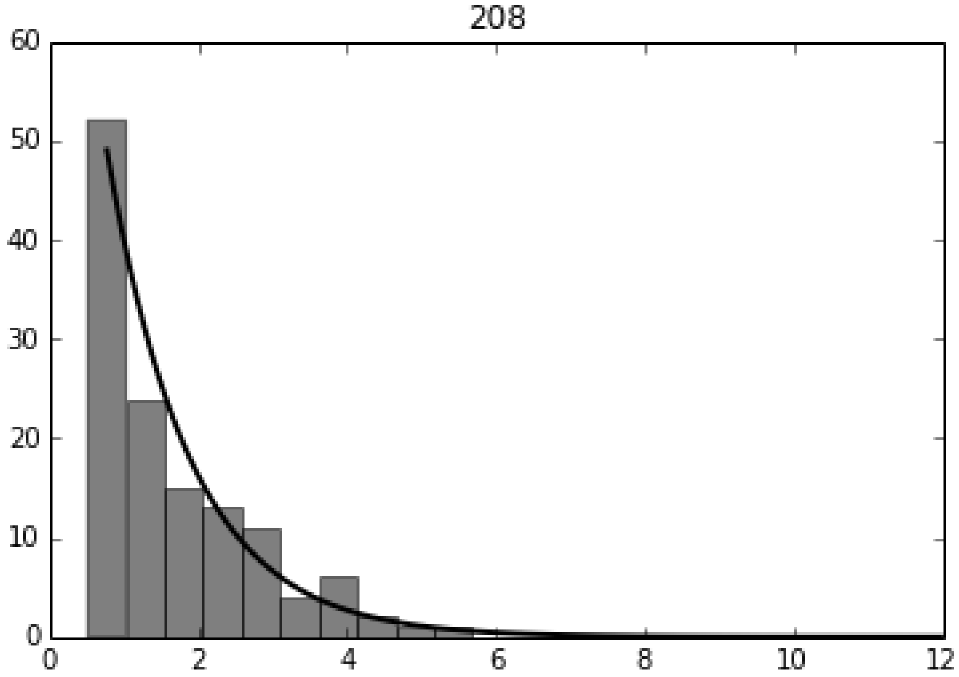


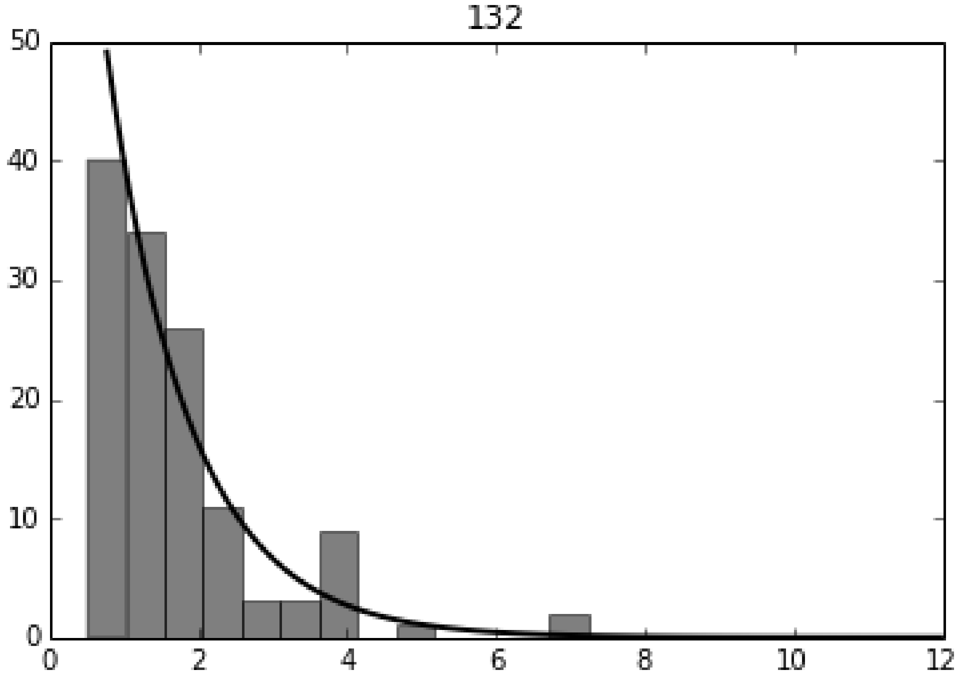
Figures can be generated and saved in a variety of formats. The code block below calculates fits to the data for each experiment and outputs the resulting graph to the notebook and saves it as an eps file.

In [24]:

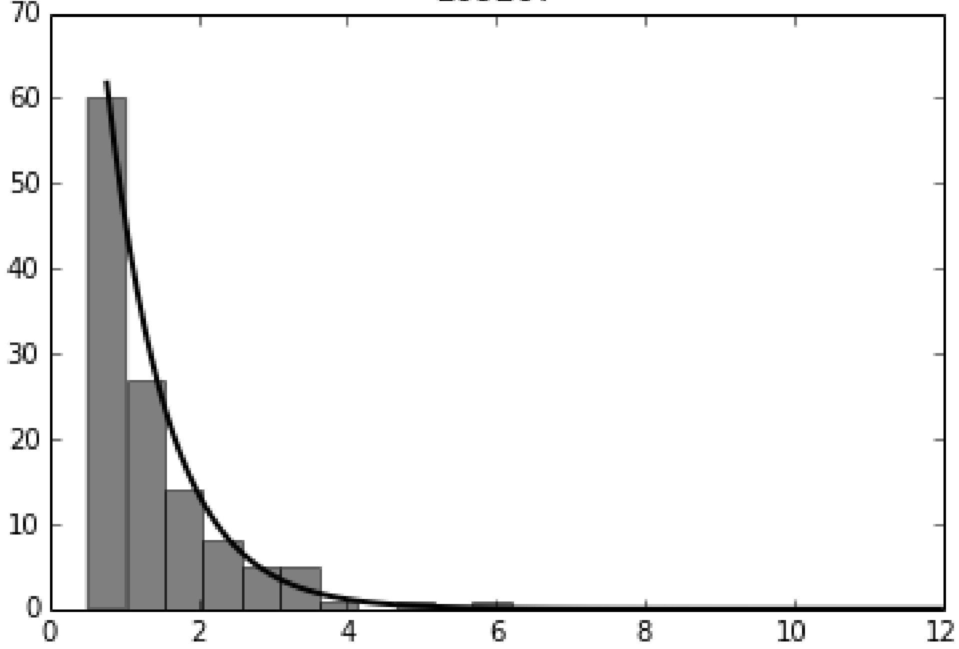
```
def fitline(x, mean, binsize, n, start):
    """Return y = f(x), where f(x) is the fit line a histogram from
    n data points with given mean and binsize
    """
    y = 1/mean*np.exp(-(x-start)/mean)*n*binsize
    return y

def rlplot(col, start, stop, binsize):
    """Return the mean of the data in col and produce a histogram plot
    running from start to stop with binsize=binsize, with fit line.

    Plots are saved as eps files.
    """
    x = np.arange(start + binsize/2, stop, .1) # xvals for fit line
    decayconst = np.mean(col) - start
    y = fitline(x, decayconst, binsize, len(col), start) # compute fit line

    plt.figure() # make a new figure
    plt.hist(col.values, bins=np.arange(start, stop, binsize), color='k', alpha=0.5) #
    plt.plot(x, y, color='k', linewidth=2) # add the fit lines
    plt.title(col.name)
    plt.savefig("../figures/" + col.name + "_runlength.eps") # save as .eps
    return decayconst

start = .5172
stop = 12
binsize = .5172
data.apply(rlplot, args=(start, stop, binsize)) # apply the plotting function to each
```

```
          208        1.112581
Out [24]: 132        1.110577
          209207     0.808655
          209144     2.253815
          2098582    1.413550
          dtype: float64
```

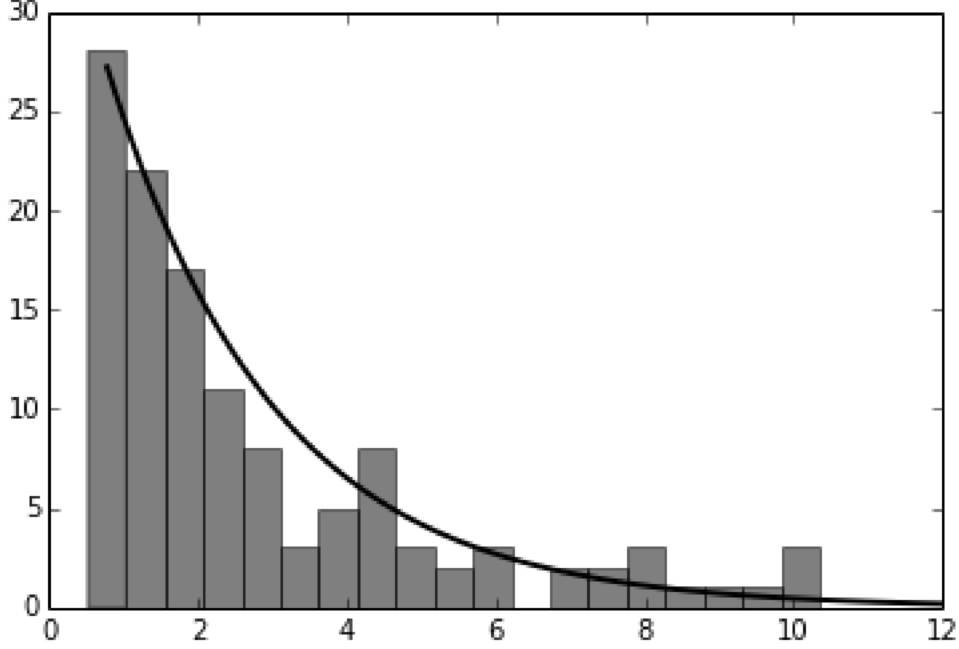


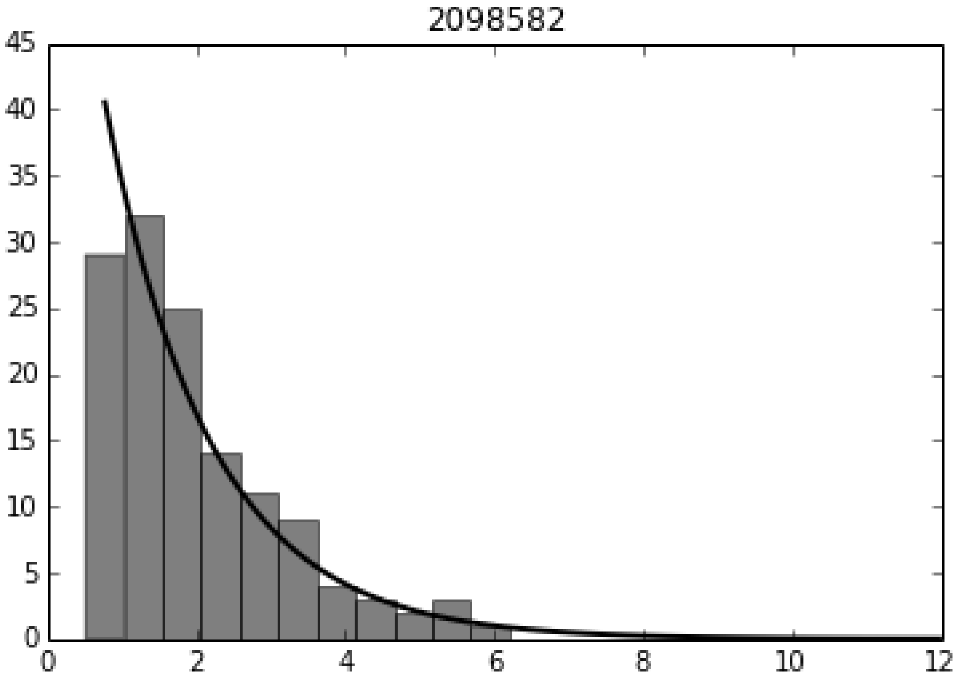


209207



209144




2098582
45
40
35
30
25
20
15
10
5
0
0
2
4
6
8
10
12