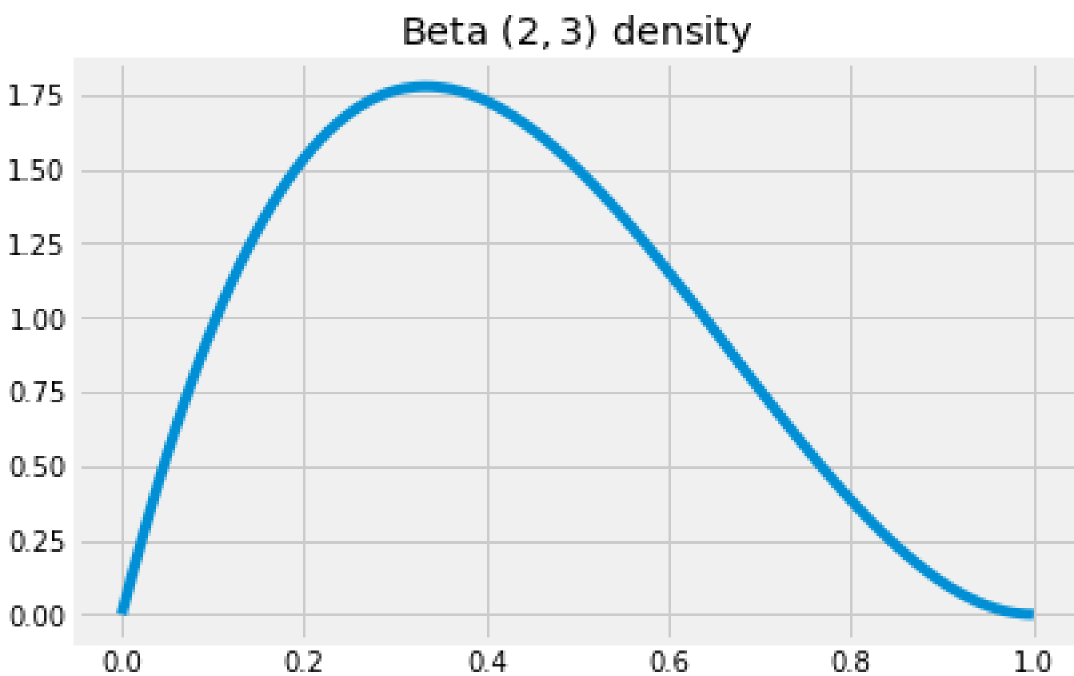
Notebook

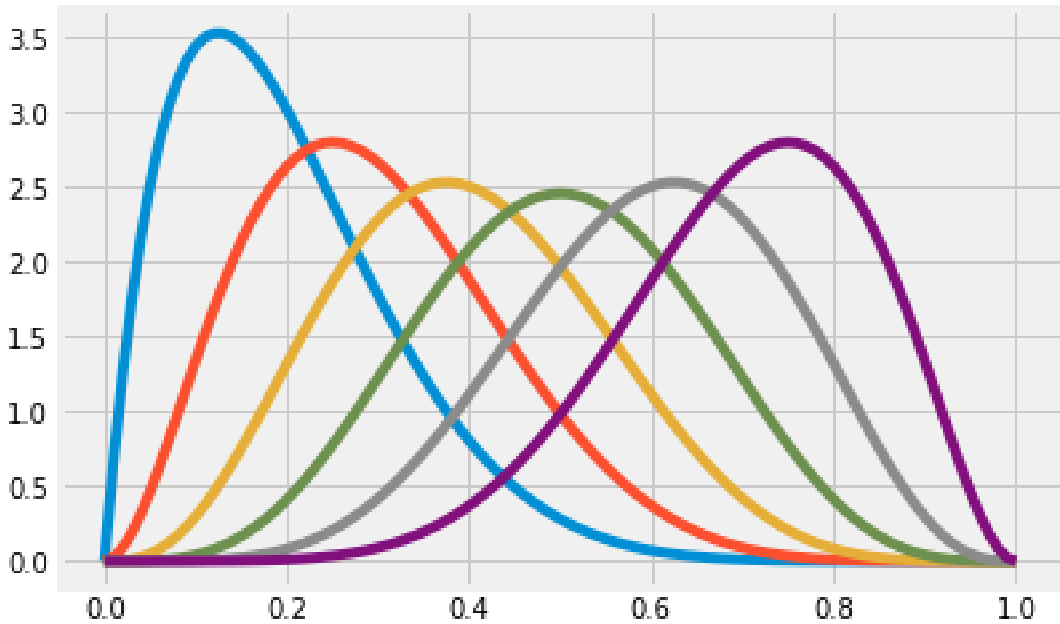
November 13, 2018

Local date & time is : 11/13/2018 13:46:29 PST

In [6]:
```
# Your answer to 1a

x = np.arange(0, 1.01, 0.01)

plt.plot(x,stats.beta.pdf(x, 2, 3))

plt.title('Beta $(2, 3)$ density');
```



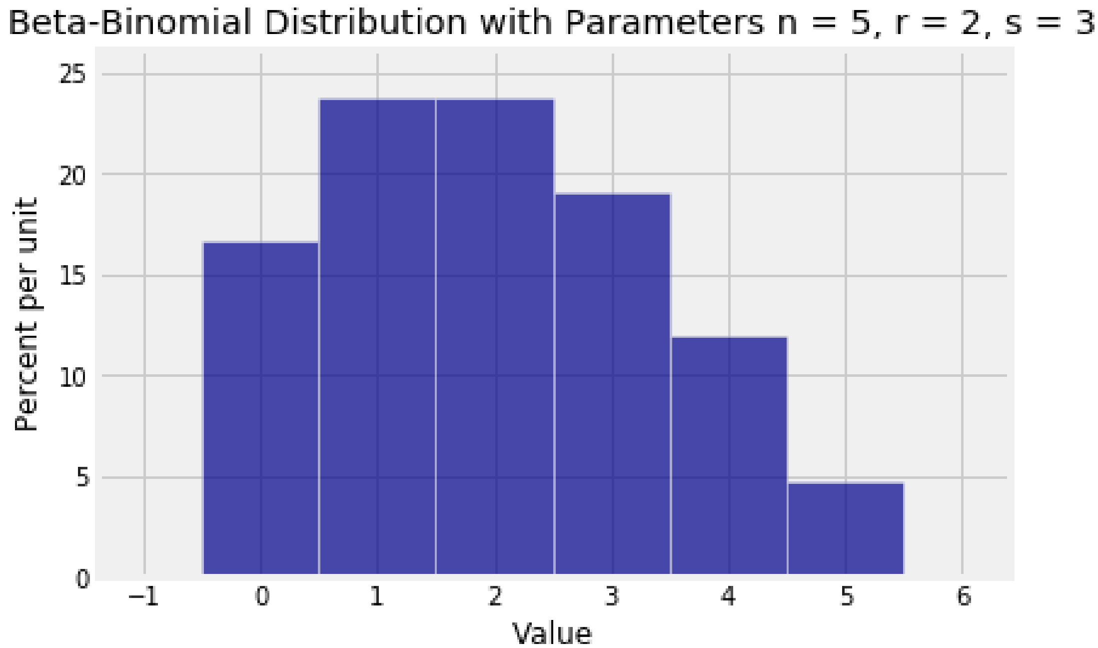
In [7]:
```
# Your answer to 1b
for i in range(0,6):
    plt.plot(x,stats.beta.pdf(x, 2 + i, 8-i))
```



In [8]:
```
# Your answer to 1c

r = 2
s = 3
n = 5
k = np.arange(0, n+1)

map_estimates = (r + k - 1) / (r + s + n - 2)

map_estimates
```

Out[8]:
```
array([0.125, 0.25 , 0.375, 0.5  , 0.625, 0.75 ])
```

In [9]:
```
# Your answer to 1d

def probs_N(n):
    def C(r, s):
        return special.gamma(r + s) / (special.gamma(r) * special.gamma(s))
    return special.comb(5, n) * (C(2, 3) / C(2 + n, 3 + 5 - n))

dist = Table().values(np.arange(6)).probability_function(probs_N)
Plot(dist)
plt.title('Beta-Binomial Distribution with Parameters n = 5, r = 2, s = 3');
```


Beta-Binomial Distribution with Parameters n = 5, r = 2, s = 3
Percent per unit
Value
0
5
10
15
20
25
−1
0
1
2
3
4
5
6

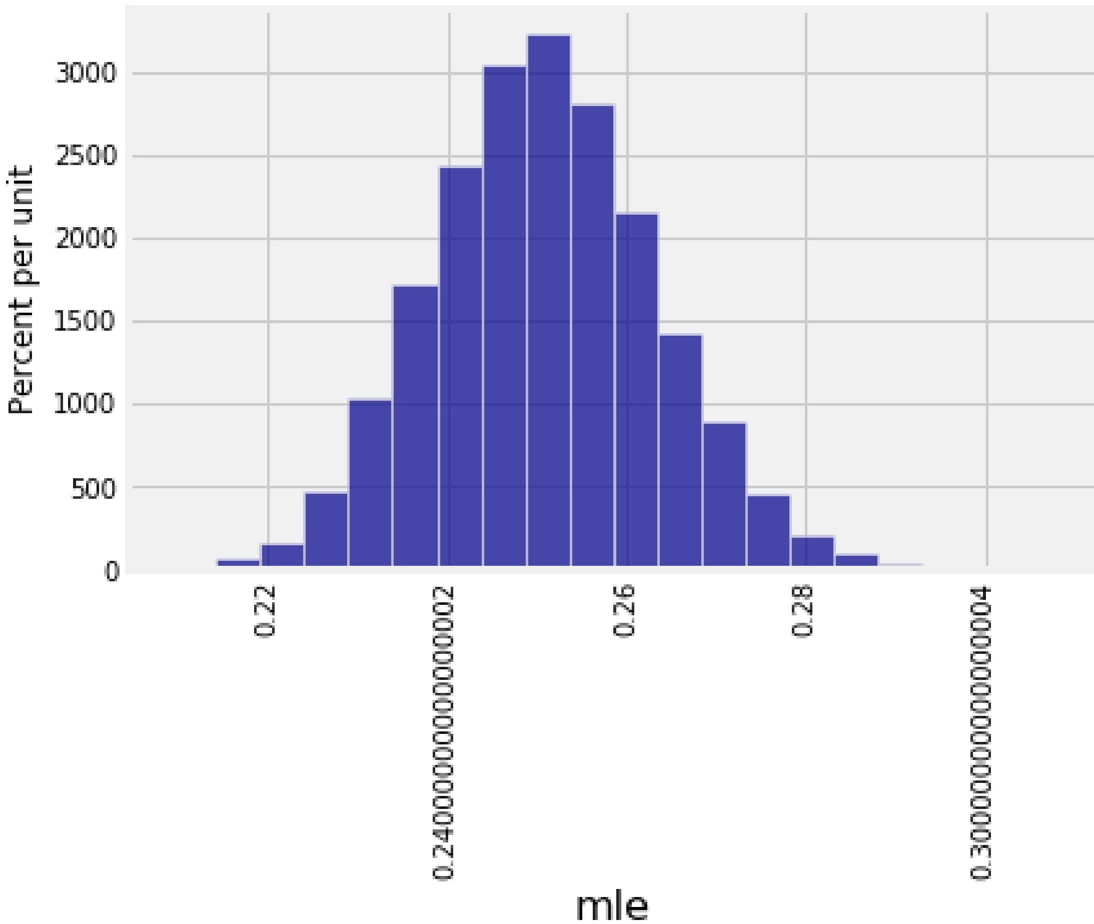
1 newpage

2 newpage

3 newpage

4 newpage

In [10]:
```
# Your answer to 5b
lam = 0.25

mle_400 = make_array()
for i in np.arange(0,10000):
    rvs = stats.expon.rvs(scale = 1/lam, size=400)
    mle_400 = np.append(mle_400, 1 / np.mean(rvs))
Table().with_column('mle', mle_400).hist(bins=20)
```



In [11]:
```
# The mean of your 10000 mle's
np.mean(mle_400)
```

Out[11]: 0.25074372463559136

5 newpage

In [14]:
```
#your solution to 6a
np.mean(original_sample)
```

Out[14]: 11.843044976596191

In [15]:
```
3 * 625 / sum(original_sample)
```

Out[15]: 0.2533132320217051

In [46]:
```
#your solution to 6d

def log_likelihood(r, lam, data):
    sample_sum =  sum(data)
    sum_of_logs = sum([np.log(i) for i in data])
    ans = 625 * r * np.log(lam) - 625 * np.log(special.gamma(r)) - lam * sample_sum + (r - 1) *
    return ans
```

In [47]:
```
log_likelihood(3,0.25, original_sample)
```

Out[47]: -2059.282424391444

In [49]:
```
#your solution to 6d (continued)

def function_to_minimize(r, lam):
    return -log_likelihood(r, lam, original_sample)
```

In [51]:
```
#your solution to 6e

r_mles = make_array()

for i in range(0,2500):
    new_table = original_tbl.sample(625)
    #print(new_table)
    original_sample = np.random.choice(new_table.column(0), 625)
    #print(original_sample)
    predictions = minimize(function_to_minimize, method = 'Nelder-Mead')
    #print(predictions)
    r_mles = np.append(r_mles, predictions[0])
    #log_likelihood(r, lam, data)
```

In [53]:
```
#6e continued

left_end = percentile(2.5, r_mles)
right_end = percentile(97.5, r_mles)

good = left_end < true_r and right_end > true_r

[left_end, right_end], good
```

Out[53]: ([2.3866311866317296, 3.2488644130709075], True)