Bayesian Data Analysis Chapter 2

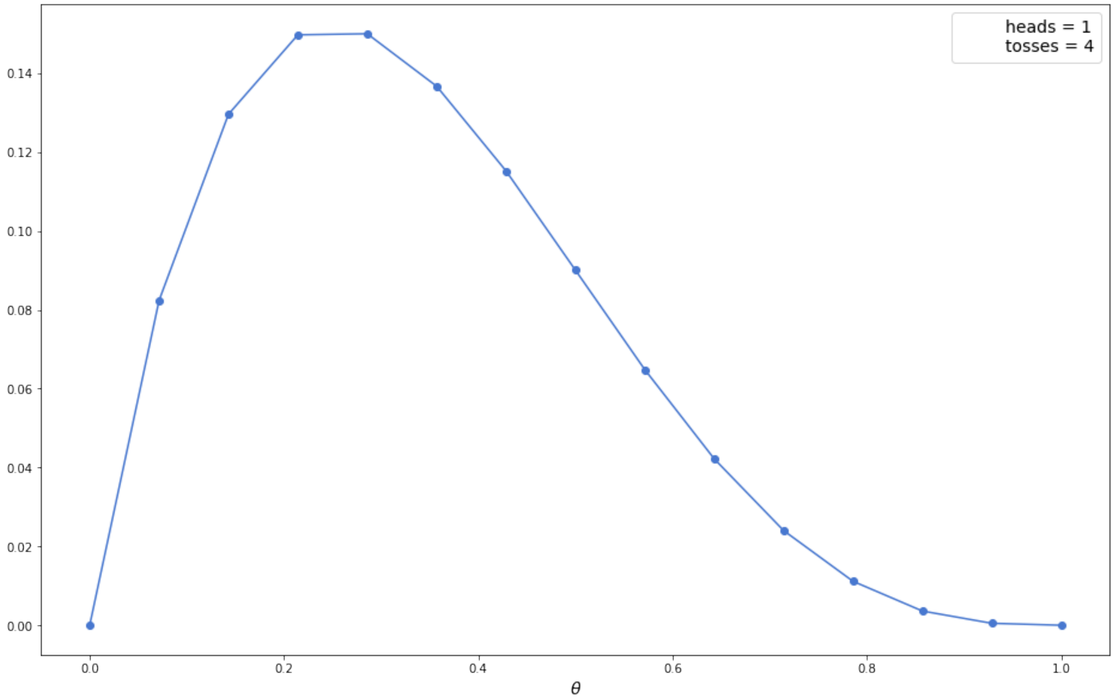
January 19, 2019

In [1]:
```
%matplotlib inline
import pymc3 as pm
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib
palette = 'muted'
sns.set_palette(palette); sns.set_color_codes(palette)

figureSize=(16,10)
```

In [2]:
```
def posterior_grid(grid_points=100, heads=6, tosses=9):
    """
    A grid implementation for the coin-flip problem
    """
    # define a grid
    grid = np.linspace(0, 1, grid_points)

    # define prior
    prior = np.repeat(5, grid_points)  # uniform
    #prior = (grid  <= 0.4).astype(int)  # truncated
    #prior = abs(grid - 0.5)  # "M" prior

    # compute likelihood at each point in the grid
    likelihood = stats.binom.pmf(heads, tosses, grid)

    # compute product of likelihood and prior
    unstd_posterior = likelihood * prior

    # standardize the posterior, so it sums to 1
    posterior = unstd_posterior / unstd_posterior.sum()
    return grid, posterior
```

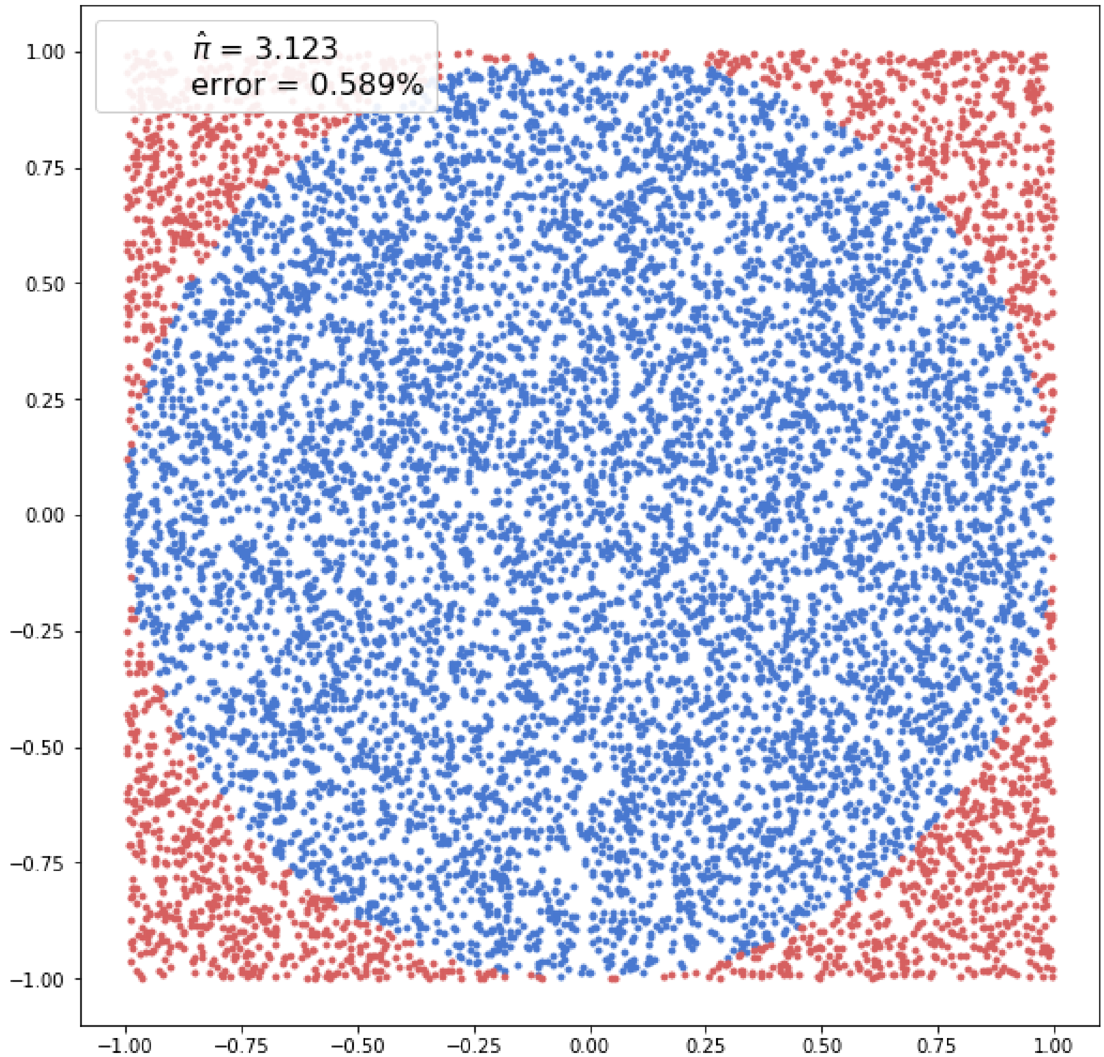
In [3]:
```
points = 15
h, n = 1, 4
grid, posterior = posterior_grid(points, h, n)
```

```
plt.plot(grid, posterior, 'o-')
plt.plot(0, 0, label='heads = {}\ntosses = {}'.format(h, n), alpha=0)
plt.xlabel(r'$\theta$', fontsize=14)
plt.legend(loc=0, fontsize=14)
fig = matplotlib.pyplot.gcf()
fig.set_size_inches(figureSize[0],figureSize[1], forward=True)
```



0.1 Monte Carlo

In [4]:
```
N = 10000

x, y = np.random.uniform(-1, 1, size=(2, N))
inside = (x**2 + y**2)  <= 1
pi = inside.sum()*4/N
error = abs((pi - np.pi)/pi)* 100

outside = np.invert(inside)

plt.plot(x[inside], y[inside], 'b.')
plt.plot(x[outside], y[outside], 'r.')
plt.plot(0, 0, label='$\hat \pi$ = {:4.3f}\nerror = {:4.3f}%'.format(pi, error), alpha=
plt.axis('square')
plt.legend(frameon=True, framealpha=0.9, fontsize=16);
fig = matplotlib.pyplot.gcf()
fig.set_size_inches(figureSize[0],figureSize[1])
```



0.2 Metropolis

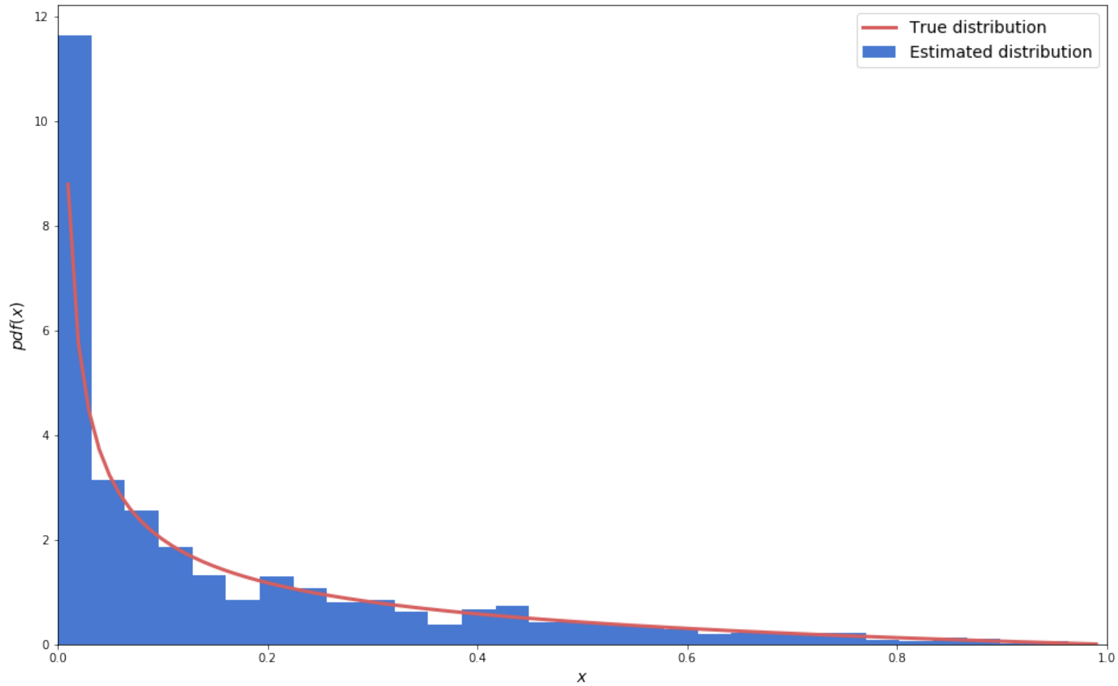
0.2.1 this is not a good implementation see https://www.youtube.com/watch?v=OTO1DygELpY

In [5]:
```
def metropolis(func, steps=10000):
    """A very simple Metropolis implementation"""
    samples = np.zeros(steps)
    old_x = func.mean()
    old_prob = func.pdf(old_x)

    for i in range(steps):
        new_x = old_x + np.random.normal(0, 1)
        new_prob = func.pdf(new_x)
        acceptance = new_prob/old_prob
```

```
                if acceptance >= np.random.random():
                    samples[i] = new_x
                    old_x = new_x
                    old_prob = new_prob
                else:
                    samples[i] = old_x
            return samples
```

```
In [6]: np.random.seed(345)
        func = stats.beta(0.4, 2)
        samples = metropolis(func=func)
        x = np.linspace(0.01, .99, 100)
        y = func.pdf(x)
        plt.xlim(0, 1)
        plt.plot(x, y, 'r-', lw=3, label='True distribution')
        plt.hist(samples, bins=30, normed=True, label='Estimated distribution')
        plt.xlabel('$x$', fontsize=14)
        plt.ylabel('$pdf(x)$', fontsize=14)
        plt.legend(fontsize=14)
        fig = matplotlib.pyplot.gcf()
        fig.set_size_inches(figureSize[0],figureSize[1])
```

```
/home/damianos/miniconda3/envs/pymc3/lib/python3.7/site-packages/matplotlib/axes/_axes.py:6521
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be removed in 3.1. Use 'density' i
  alternative="'density'", removal="3.1")
```



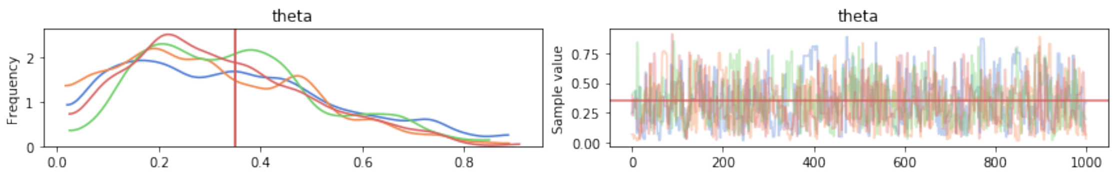
0.3 PyMC3 primer

In [7]:
```
np.random.seed(123)
n_experiments = 4
theta_real = 0.35  # unkwon value in a real experiment
data = stats.bernoulli.rvs(p=theta_real, size=n_experiments)
```

In [8]:
```
with pm.Model() as our_first_model:
    # a priori
    theta = pm.Beta('theta', alpha=1, beta=1)
    # likelihood
    y = pm.Bernoulli('y', p=theta, observed=data)
    # start at the maximum a posteriori probability (MAP) estimate
    # see https://en.wikipedia.org/wiki/Maximum_a_posteriori_estimation
    start = pm.find_MAP()
    # use M-H algorithm to sample
    step = pm.Metropolis()
    trace = pm.sample(1000, step=step, start=start)
```

```
/home/damianos/miniconda3/envs/pymc3/lib/python3.7/site-packages/pymc3/tuning/starting.py:61: U
  warnings.warn('find_MAP should not be used to initialize the NUTS sampler, simply call pymc3
logp = -2.7726, ||grad|| = 1: 100%|| 6/6 [00:00<00:00, 3584.36it/s]
Multiprocess sampling (4 chains in 4 jobs)
Metropolis: [theta]
Sampling 4 chains: 100%|| 6000/6000 [00:00<00:00, 12985.97draws/s]
The number of effective samples is smaller than 25% for some parameters.
```

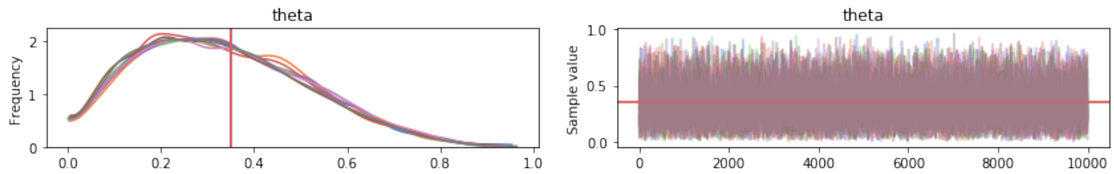
In [9]:
```
burnin = 0  # no burnin
chain = trace[burnin:]
pm.traceplot(chain, lines={'theta':theta_real});
```



In [10]:
```
with our_first_model:
    step = pm.Metropolis()
    multi_trace = pm.sample(10000, step=step, njobs=8)
```

```
Multiprocess sampling (8 chains in 8 jobs)
Metropolis: [theta]
Sampling 8 chains: 100%|| 84000/84000 [00:06<00:00, 12045.26draws/s]
The number of effective samples is smaller than 25% for some parameters.
```

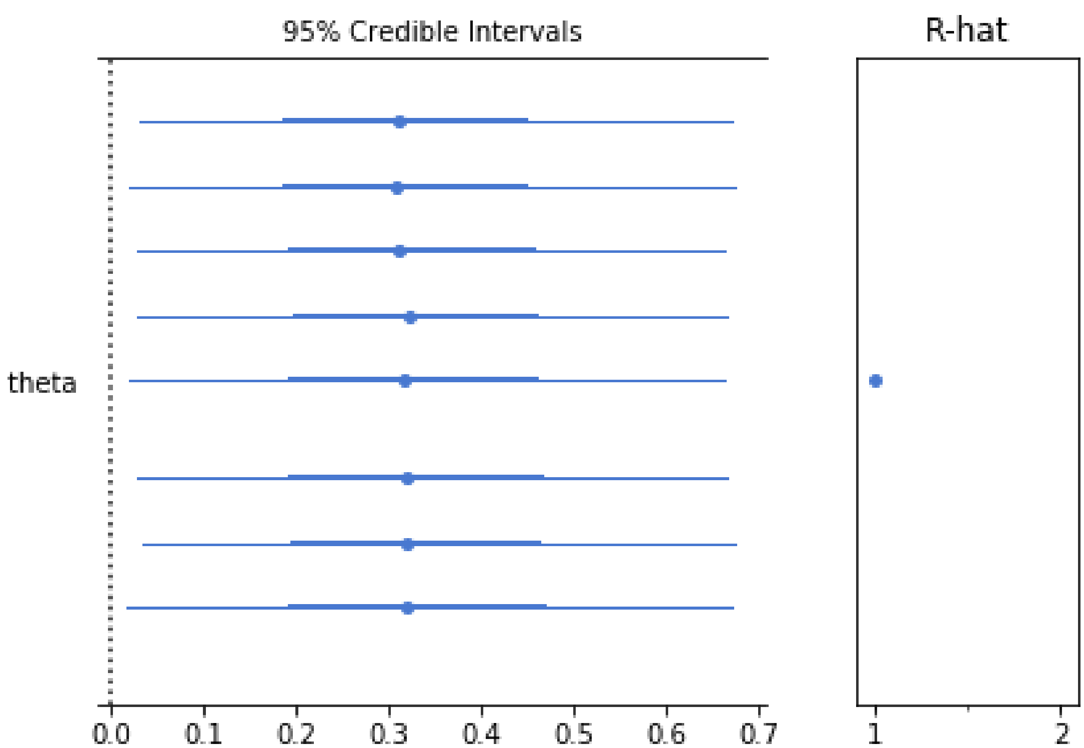
In [11]:
```
burnin = 0   # no burnin
multi_chain = multi_trace[burnin:]
pm.traceplot(multi_chain, lines={'theta':theta_real});
```



In [12]:
```
pm.gelman_rubin(multi_chain)
```

Out[12]:
```
{'theta': 1.000235781648356}
```

In [13]:
```
pm.forestplot(multi_chain, varnames=['theta']);
```



In [14]:
```
pm.summary(multi_chain)
```

Out[14]:

	mean	sd	mc_error	hpd_2.5	hpd_97.5	n_eff	Rhat
theta	0.335045	0.17943	0.001399	0.023004	0.668089	16414.812056	1.000236

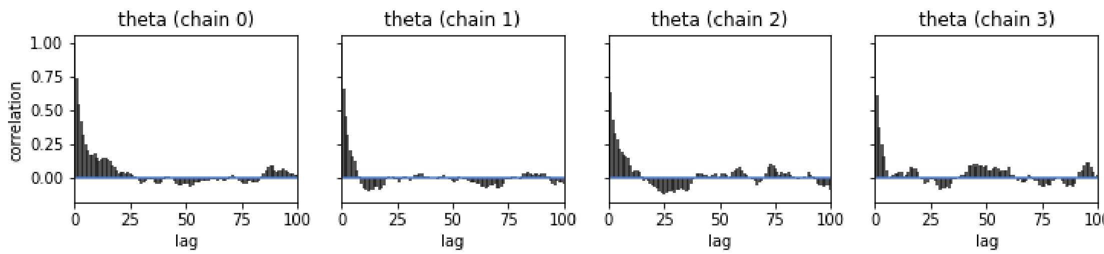
In [15]:
```
pm.summary(multi_chain)
```

Out[15]:

	mean	sd	mc_error	hpd_2.5	hpd_97.5	n_eff	Rhat
theta	0.335045	0.17943	0.001399	0.023004	0.668089	16414.812056	1.000236

In [16]:
```
pm.autocorrplot(chain)
```

Out[16]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f3b21839c50>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f3b2171d710>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f3b216486a0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f3b21671630>]],
      dtype=object)
```

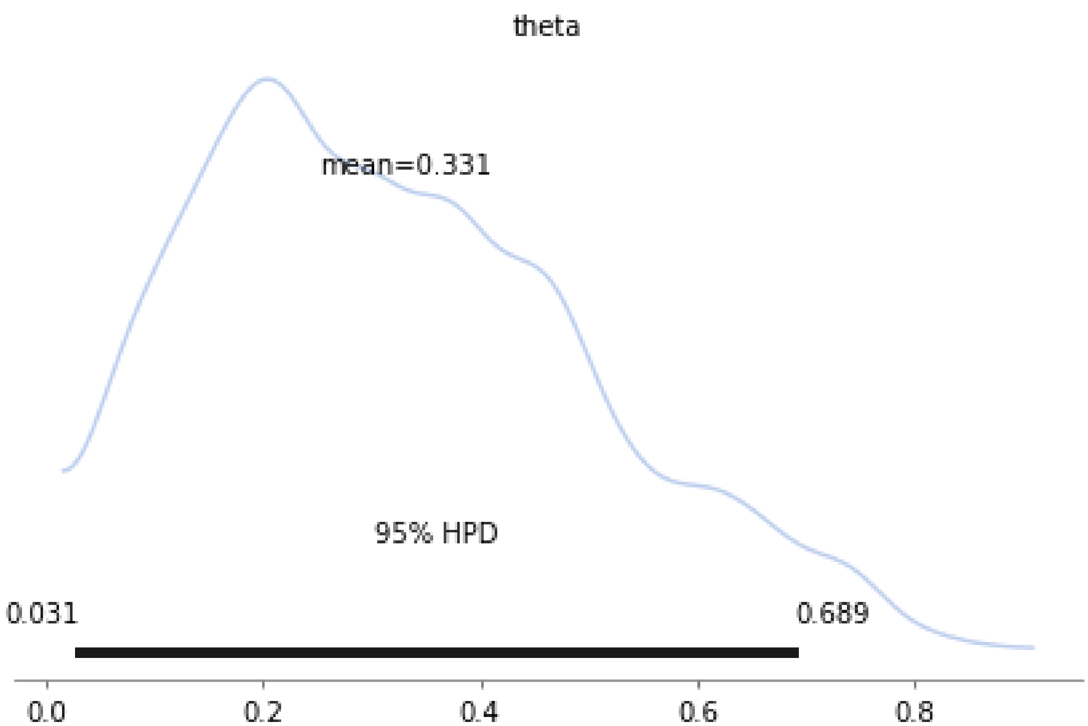


In [17]:
```
pm.effective_n(multi_chain)['theta']
```

Out[17]:
```
16414.812055918923
```

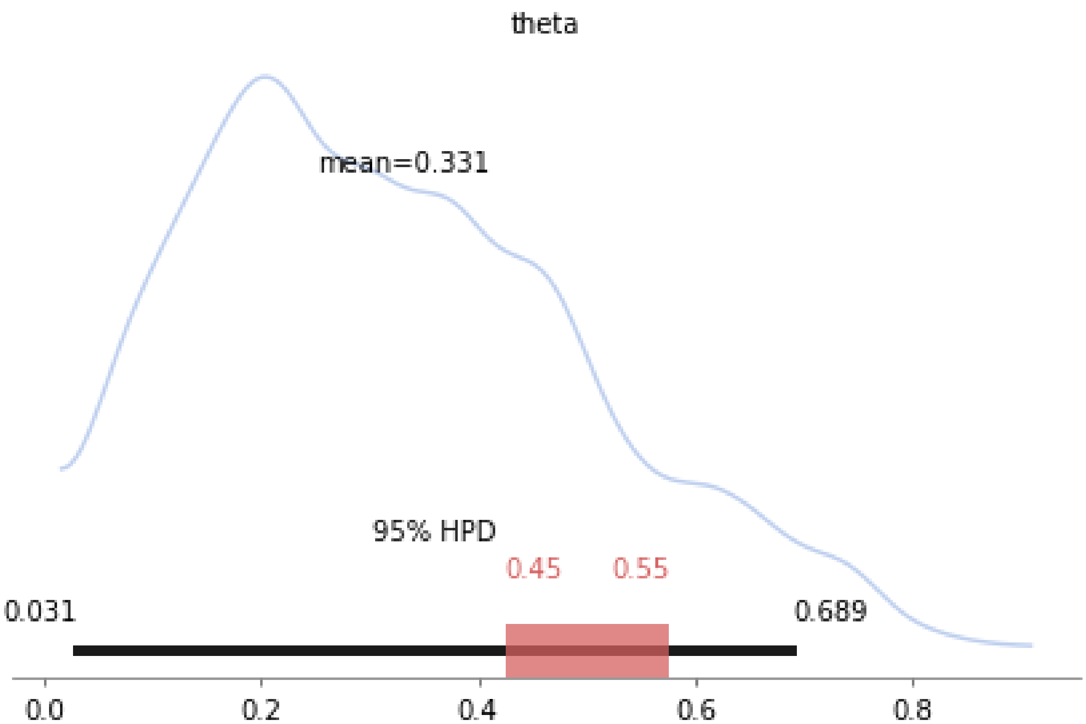
In [18]:
```
pm.plot_posterior(chain, kde_plot=True)
```

Out[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3b218d6d68>
```



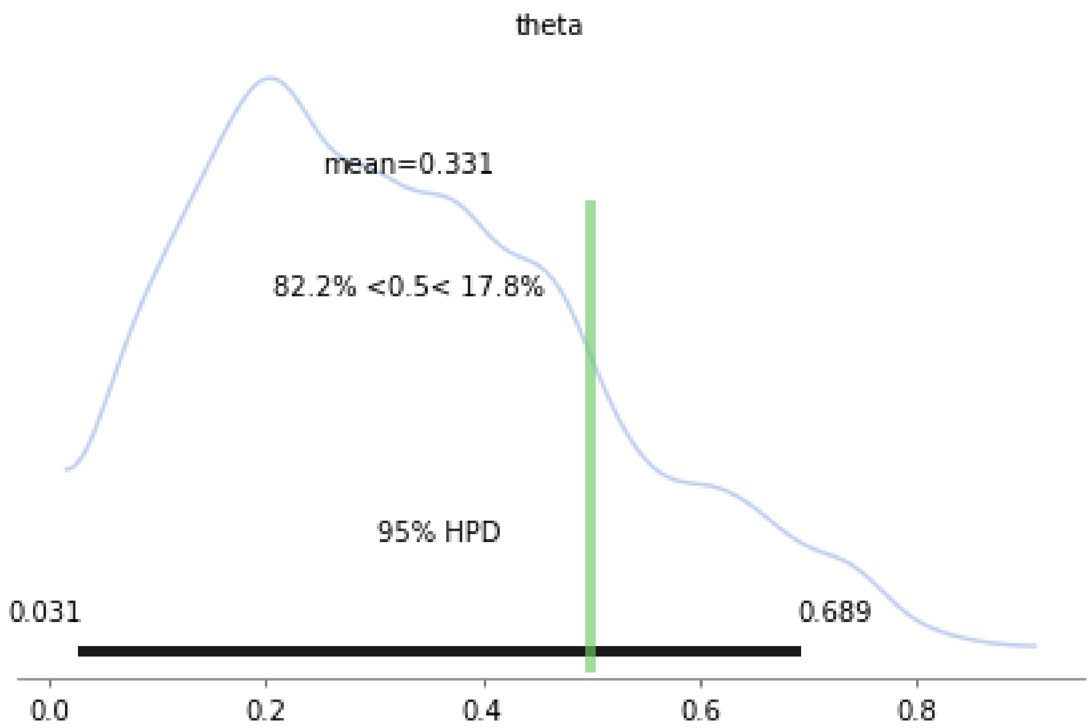
```
In [19]: pm.plot_posterior(chain, kde_plot=True, rope=[0.45, .55])
```

Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x7f3b229dd0f0>




```
In [20]: pm.plot_posterior(chain, kde_plot=True, ref_val=0.5)
```

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x7f3b22acdc50>




```
In [21]: import sys, IPython, scipy, matplotlib, platform
         print("This notebook was created on a %s computer running %s and using:\nPython %s\nIP
```

```
This notebook was created on a x86_64 computer running debian buster/sid and using:
Python 3.7.2
IPython 7.2.0
PyMC3 3.6
NumPy 1.16.0
SciPy 1.2.0
Matplotlib 3.0.2
Seaborn 0.9.0


/home/damianos/miniconda3/envs/pymc3/lib/python3.7/site-packages/ipykernel_launcher.py:2: Depre
```