PART 1:

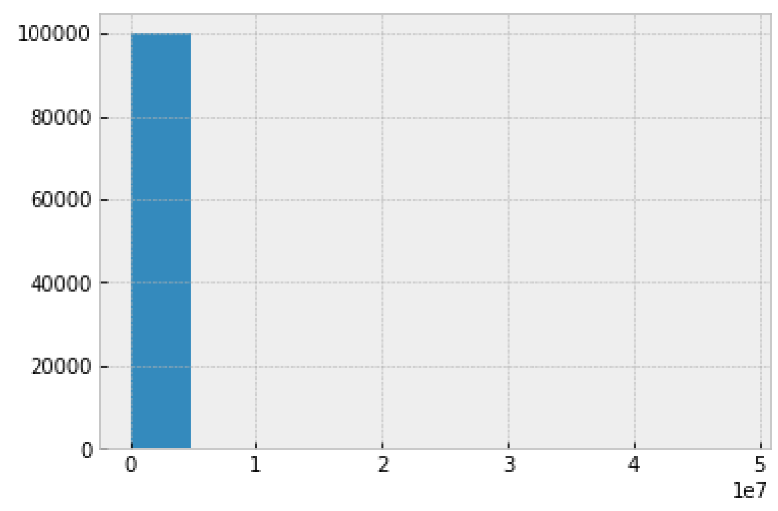
In [3]:
```
import pandas as pd
data = pd.read_csv('Houses.csv',names=['TUI','price','date_of_transfer','postcode','property_type',
'O/N','duration','paon','saon','street','locality','town','district','country','Category_type','reco
rd_status'])
```

In [4]:
```
data.price.describe()
```

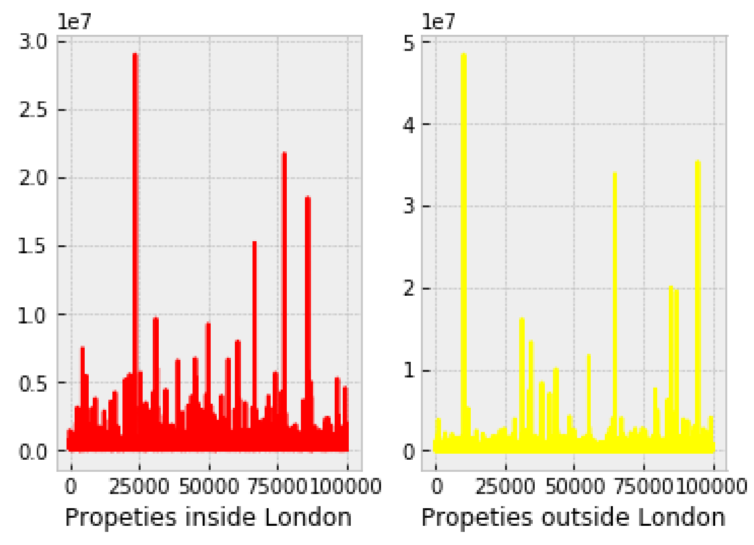
Out[4]:
```
count    1.000000e+05
mean     1.743868e+05
std      3.514634e+05
min      1.500000e+02
25%      7.400000e+04
50%      1.290000e+05
75%      2.070000e+05
max      4.846572e+07
Name: price, dtype: float64
```

Histogram

In [6]:
```
%matplotlib inline
from matplotlib import pyplot as plt
import numpy as np
import scipy.stats as stats
plt.style.use('bmh')
plt.hist(data.price)
plt.show()
```

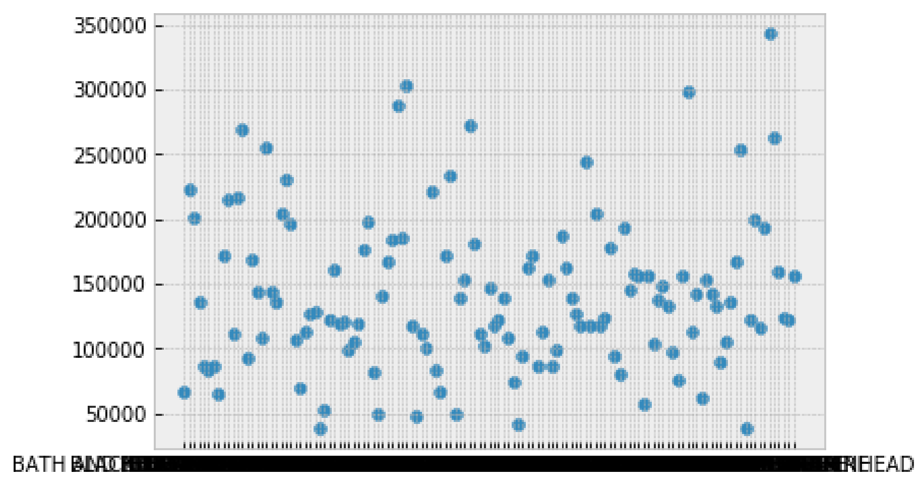


In [10]:
```
inside_london = data[data.town == 'LONDON'].price
outside_london = data[data.town != 'LONDON'].price

plt.subplot(1,2,1)
plt.plot(inside_london, color = 'red')
plt.xlabel('Propeties inside London')

plt.subplot(1,2,2)
plt.plot(outside_london, color = 'yellow')
plt.xlabel('Propeties outside London')

plt.show()
```



In [13]:
```
mean_estate_prices = data.groupby('country').price.mean()
```

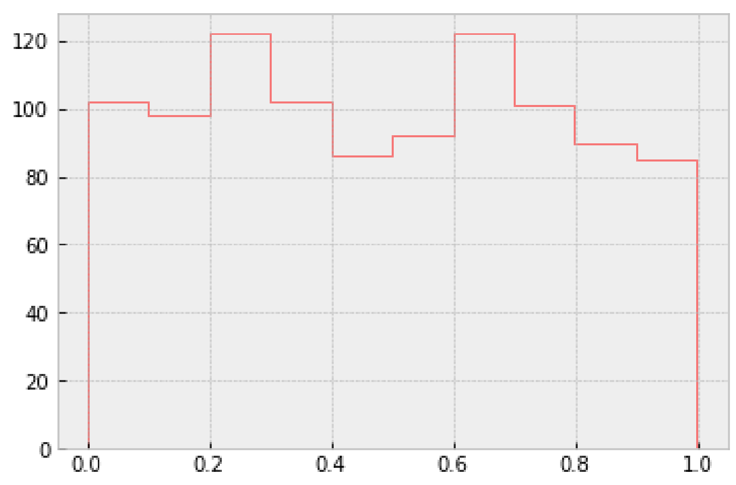
In [14]:
```
# Using scatterplot filtering by countries:
plt.scatter(mean_estate_prices.index,mean_estate_prices)
plt.show()
```



Part 2:

In [23]:
```
import numpy as np
sample = list(np.random.rand() for _ in range(1000))
plt.hist(sample, color = 'red',histtype = 'step')
```

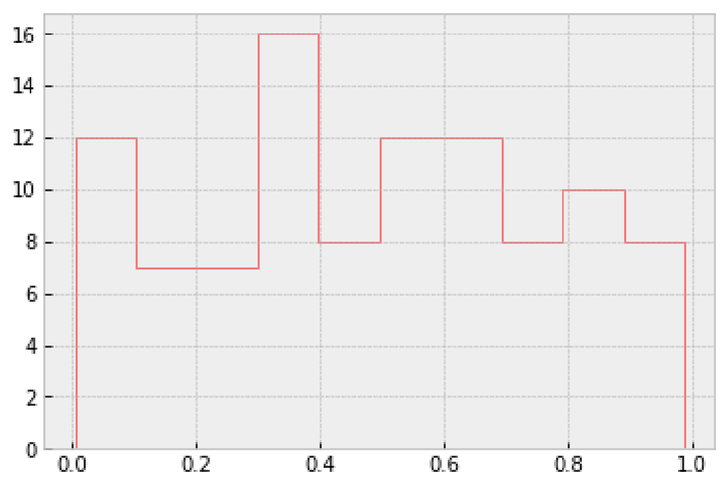
Out[23]:
```
(array([102.,  98., 122., 102.,  86.,  92., 122., 101.,  90.,  85.]),
 array([9.73351802e-04, 1.00797353e-01, 2.00621355e-01, 3.00445356e-01,
        4.00269357e-01, 5.00093359e-01, 5.99917360e-01, 6.99741361e-01,
        7.99565363e-01, 8.99389364e-01, 9.99213365e-01]),
 <a list of 1 Patch objects>)
```



Shape of histogram is quite what we expect it to be, i.e. uniform and randomized but not normal

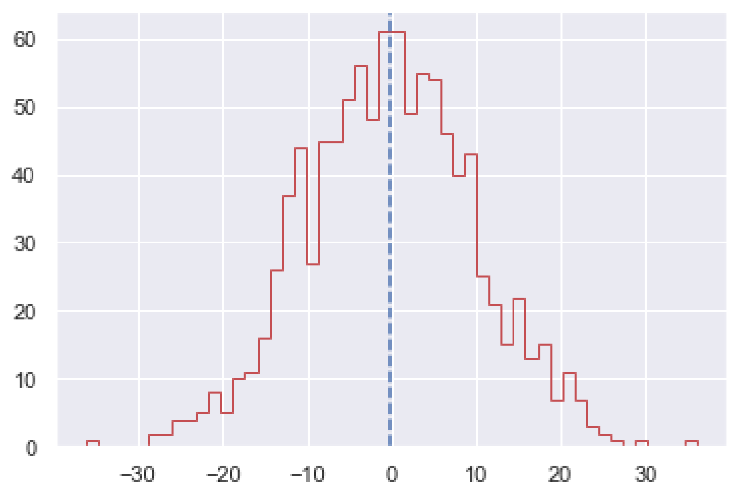
In [33]:
```
import numpy as np
np.random.seed(10)
sample = list(np.random.rand() for _ in range(100))
plt.hist(sample, color = 'red',histtype = 'step')
```

Out[33]:
```
(array([12.,  7.,  7., 16.,  8., 12., 12.,  8., 10.,  8.]),
 array([0.00394827, 0.10231599, 0.20068371, 0.29905143, 0.39741915,
        0.49578687, 0.59415459, 0.69252231, 0.79089003, 0.88925775,
        0.98762547]),
 <a list of 1 Patch objects>)
```



Shape is absolutely affected by varying the numbers that we have generated

In [55]:
```
# Using normal random variable
import numpy as np
import seaborn as sns
sns.set()
sample = np.random.normal(0,10,1000)
plt.hist(sample, color = 'r',histtype = 'step',bins=50)
plt.axvline(sample.mean(), linestyle = 'dashed')
plt.show()
```



In [56]:
```
## The best fit here is a bell shape
```