In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

標本平均：コインを n 回投げた時の表の出る相対頻度

In [2]:
```
# コインを n 回投げた時、表の出る相対頻度 F を調べる(F は n 個の無作為標本の標本平均)
n=12
nCoin=[]
for i in range(0,n):
    if np.random.rand(1)<0.5:
        Z=0
    else:
        Z=1
    nCoin.append(Z)
nCoin=np.array(nCoin)    # コイン投げの結果をアレイにしておく
SM=nCoin.mean()            # n 回中の表の相対頻度(標本平均)
SM
```

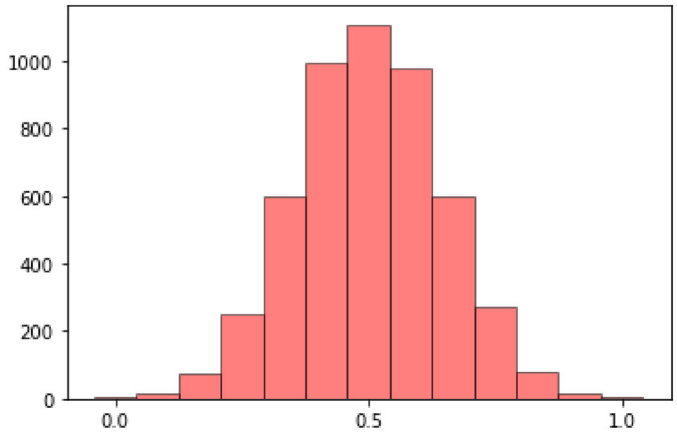
Out[2]:
```
0.3333333333333333
```

標本平均の分布をシミュレーションで調べる

In [3]:
```
# Sample Mean がどのように分布するかを調べる
# n 回のコイン投げを多数回繰り返して SM の値を収集する
# n= 16     # 1試行で振るコインの回数（すでに定義している）
T= 5000   # トライアル数
SM_list=[]
for i in range(T):
    nCoin=[]
    for j in range(n):
        if np.random.rand(1)<0.5:
            Z=0
        else:
            Z=1
        nCoin.append(Z)
    nCoin=np.array(nCoin)      # コイン投げの結果をアレイにしておく
    SM=nCoin.mean()            # n 回中の表の相対頻度(標本平均)
    SM_list=SM_list +[SM]
SM_list=np.array(SM_list) # アレイにしておくと便利
```

In [4]:
```
# SM_list # 中身を覗く
```

In [5]:
```
# SM_list の分析（１）ヒストグラム
F,x,_ = plt.hist(SM_list,
         range=(-1/(2*n),1+1/(2*n)),  # 幅を指定
         bins=n+1,                    # 階級数を指定
         color='red',                 # 色を指定
         alpha=0.5,                   # 色の濃さ（0～1）
         ec='k')                      # 棒に枠線つける（k=black）
plt.xticks(np.arange(0,1.1,0.5))   # x軸の目盛
plt.show()
```

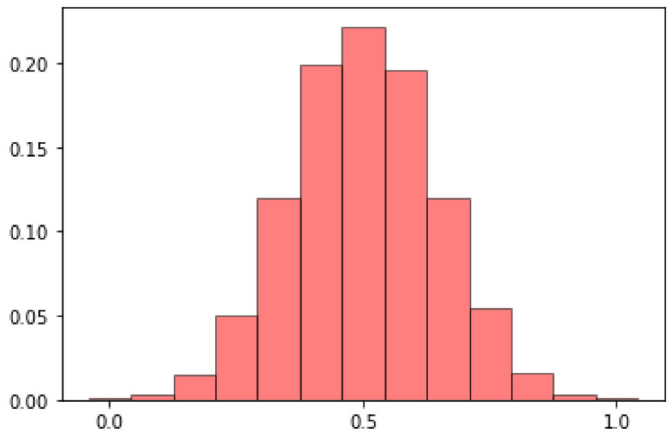


In [6]:
```
# 相対度数(relative frequency)を求める
RF=F/T
RF
```

Out[6]:
```
array([0.001 , 0.003 , 0.0144, 0.05  , 0.1202, 0.1994, 0.2218, 0.1958,
       0.12  , 0.0548, 0.0158, 0.0034, 0.0004])
```

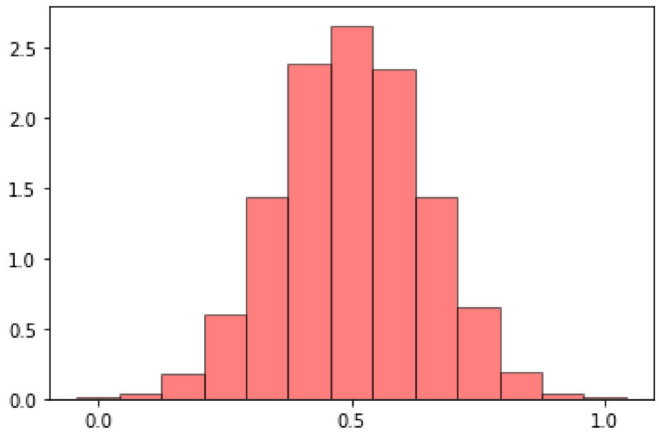
In [7]:
```
# 相対度数のヒストグラム
# plt.figure(figsize=(6,4))
xs=np.arange(0,1+1/(2*n),1/n)   # x 軸の目盛刻み
plt.bar(xs,RF,width=1/n,color='red', alpha=0.5,ec='black')
plt.xticks(np.arange(0,1.1,0.5))   # x軸の目盛表示
```

Out[7]:
```
([<matplotlib.axis.XTick at 0x21b2cf08970>,
  <matplotlib.axis.XTick at 0x21b2cf08940>,
  <matplotlib.axis.XTick at 0x21b2cf044c0>],
 [Text(0, 0, ''), Text(0, 0, ''), Text(0, 0, '')])
```



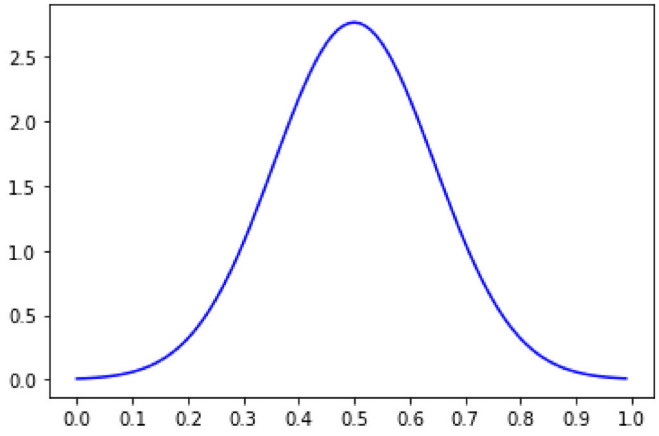
In [8]:
```
# 密度関数として扱ったヒストグラム（全面積＝１）
# plt.figure(figsize=(6,4))
# xs=np.arange(0,1+1/(2*n),1/n)    # x 軸の目盛刻み（すでに定義したものと同じ）
plt.bar(xs,RF*n,width=1/n,color='red', alpha=0.5,ec='black')  # 面積が1になるように縦軸を変更
plt.xticks(np.arange(0,1.1,0.5))    # x軸の目盛表示
```

Out[8]:
```
([<matplotlib.axis.XTick at 0x21b2cf76a90>,
  <matplotlib.axis.XTick at 0x21b2cf76a60>,
  <matplotlib.axis.XTick at 0x21b2cf715e0>],
 [Text(0, 0, ''), Text(0, 0, ''), Text(0, 0, '')])
```

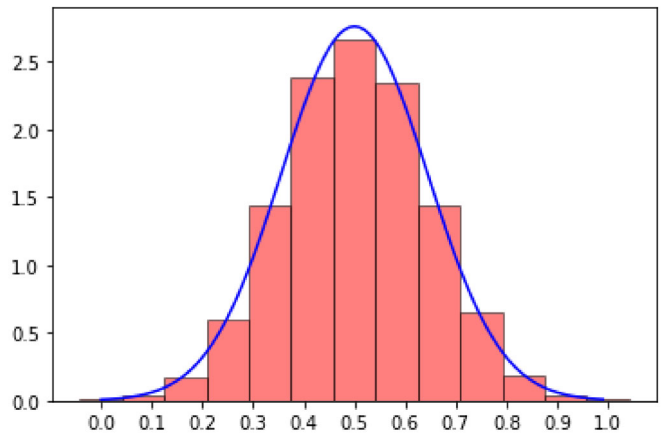


In [9]:
```
from scipy import stats # 統計ライブラリを使用
# 一般論から標本平均の平均値 = m（母平均), 分散 = s^2/n（母分散/n)
# その平均値と分散をもつ正規分布と比較する
# Ber(0.5)=B(1,0.5) の平均値と分散は m=0.5, s^2=0.25 =0.5^2
# 正規分布 N(m,s^2/n)
m=0.5                # 平均値の設定
s=0.5/np.sqrt(n)    # 標準偏差の設定
X=stats.norm(m,s)   # 通常の記号 N(m,s^2) とパラメータの使い方が異なる
```

In [10]:
```
# plt.figure(figsize=(6,4))
x=np.arange(0,1,0.01)                # x の範囲の指定、始点、終点、刻み幅
plt.plot(x,X.pdf(x),color='blue')
plt.xticks(np.arange(0,1.1,0.1))    # x軸の目盛
plt.show()
```



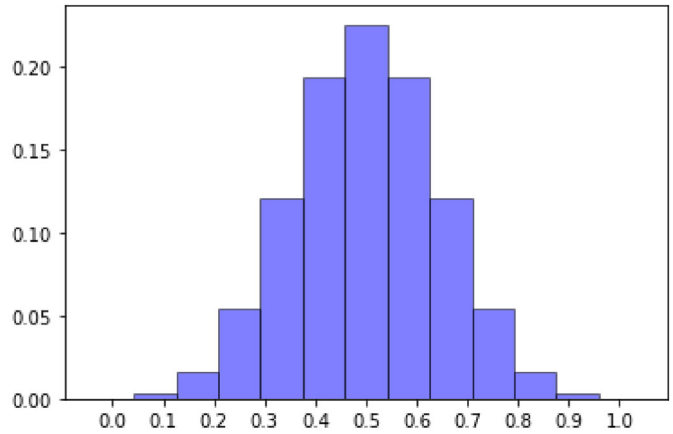
In [11]:
```
# Fitting
# plt.figure(figsize=(6,4))
plt.bar(xs,RF*n,width=1/n,color='red', alpha=0.5,ec='black')  # シミュレーション
plt.plot(x,X.pdf(x),color='blue')
plt.xticks(np.arange(0,1.1,0.1))    # x軸の目盛
plt.show()
```



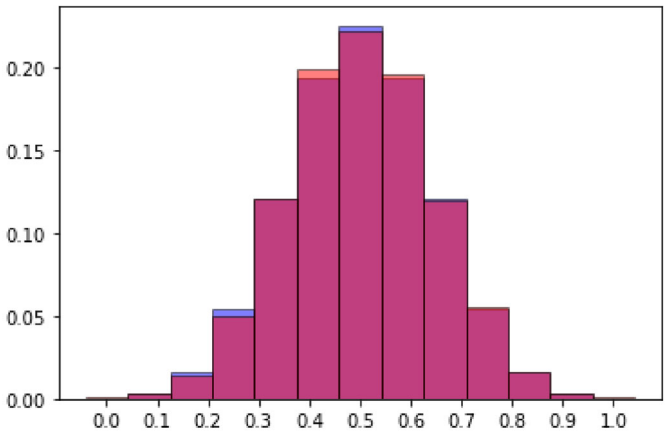
SM の理論分布とシミュレーションとの比較

In [12]:
```
# SM の理論分布は基本的に二項分布（横軸を変更するだけ）
# 二項分布 B(n,p) の定義
from scipy.special import comb    # 組合せ数を使う
p=0.5                       # パラメータの設定
x_range=np.arange(n+1)      # （本来）X の範囲は 0 から n までの n+1 個の整数
def bin(x):
    if x in x_range:
        return comb(n,x)*p**x*(1-p)**(n-x)
    else:
        return 0
BinProb = np.array([bin(x) for x in x_range])    # x_range(定義域)の各 x に対して確率を計算
```

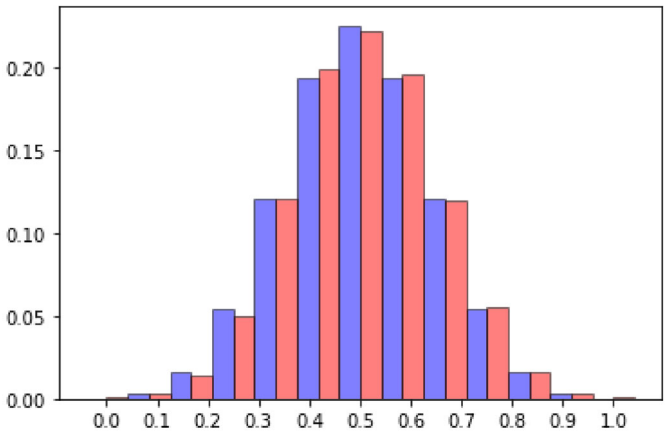
In [13]:
```
plt.figure(figsize=(6,4))
plt.bar(xs,BinProb,width=1/n, color='blue', alpha=0.5,ec='black')
plt.xticks(np.arange(0,1.1,0.1))    # x軸の目盛
plt.show()
```



In [14]:
```
# 理論とシミュレーションの比較
# plt.figure(figsize=(6,4))
plt.bar(xs,BinProb,width=1/n, color='blue', alpha=0.5,ec='black')
plt.bar(xs,RF,width=1/n,color='red', alpha=0.5, ec='black')
plt.xticks(np.arange(0,1.1,0.1))    # x軸の目盛
plt.show()
```

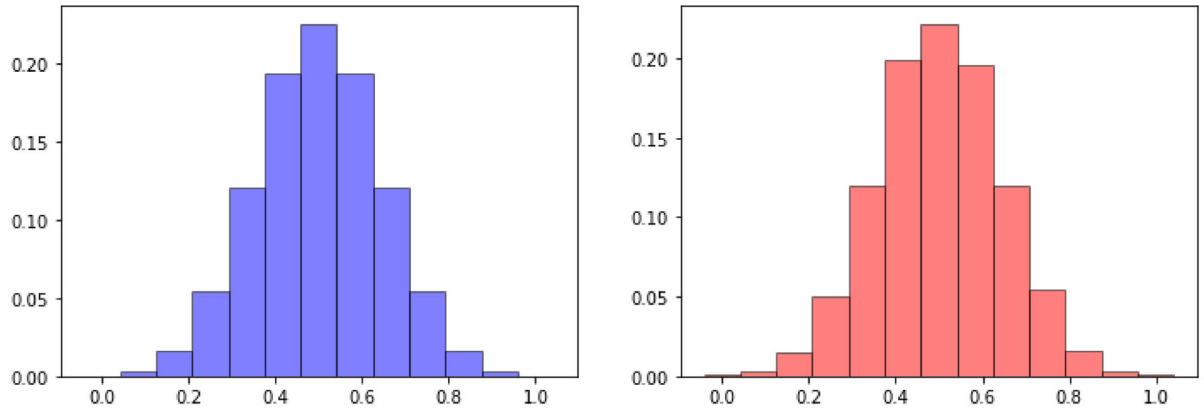


In [15]:
```
# 理論とシミュレーションの比較（棒グラフの幅を半分にして2つをずらして表示）
# plt.figure(figsize=(6,4))
plt.bar(xs-1/(4*n),BinProb,width=1/(2*n), color='blue', alpha=0.5,ec='black')
plt.bar(xs+1/(4*n),RF,width=1/(2*n),color='red', alpha=0.5, ec='black')
plt.xticks(np.arange(0,1.1,0.1))    # x軸の目盛
plt.show()
```



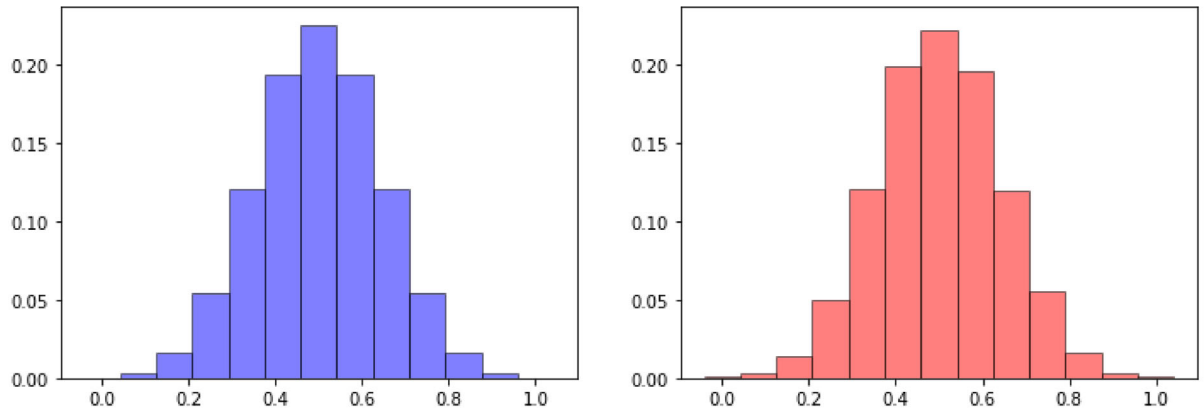
In [16]:
```
# 理論とシミュレーションの比較（グラフを並記）
fig = plt.figure(figsize=(12,4))
ax1=fig.add_subplot(1,2,1)    # 描画エリアを 1x2 に分割して1番のエリア
ax2=fig.add_subplot(1,2,2)    # 描画エリアを 1x2 に分割して2番のエリア
ax1.bar(xs,BinProb,width=1/n, color='blue', alpha=0.5,ec='black')
ax2.bar(xs,RF,width=1/n,color='red', alpha=0.5, ec='black')
# ax1.set_ylim([0,0.25])  y 軸を揃える
# ax2.set_ylim([0,0.25])
```

Out[16]: <BarContainer object of 13 artists>



In [17]:
```
# 理論とシミュレーションの比較（グラフを並記、初めから y 軸を揃える）
fig = plt.figure(figsize=(12,4))
ax1=fig.add_subplot(1,2,1)    # 描画エリアを 1x2 に分割して1番のエリア
ax2=fig.add_subplot(1,2,2, sharey=ax1)    # 描画エリアを 1x2 に分割して2番のエリア。y 軸は ax1 同じ
ax1.bar(xs,BinProb,width=1/n, color='blue', alpha=0.5,ec='black')
ax2.bar(xs,RF,width=1/n,color='red', alpha=0.5, ec='black')
```

Out[17]: <BarContainer object of 13 artists>



In [18]:
```
# シミュレーションと理論値との比較
RF, BinProb
```

Out[18]:
```
(array([0.001 , 0.003 , 0.0144, 0.05  , 0.1202, 0.1994, 0.2218, 0.1958,
        0.12  , 0.0548, 0.0158, 0.0034, 0.0004]),
 array([0.00024414, 0.00292969, 0.01611328, 0.05371094, 0.12084961,
        0.19335938, 0.22558594, 0.19335938, 0.12084961, 0.05371094,
        0.01611328, 0.00292969, 0.00024414]))
```

In [19]:
```
# DataFrame にして表示
pd.Series(RF)    # pandas の配列に変換
```

Out[19]:
```
0     0.0010
1     0.0030
2     0.0144
3     0.0500
4     0.1202
5     0.1994
6     0.2218
7     0.1958
8     0.1200
9     0.0548
10    0.0158
11    0.0034
12    0.0004
dtype: float64
```

In [20]:
```
# DataFrame にして表示
RF_S=pd.Series(RF)    # pandas の配列に変換
BinProb_S=pd.Series(BinProb)
Comparison=pd.concat([RF_S, BinProb_S],axis=1)
Comparison
```

Out[20]:

	0	1
0	0.0010	0.000244
1	0.0030	0.002930
2	0.0144	0.016113
3	0.0500	0.053711
4	0.1202	0.120850
5	0.1994	0.193359
6	0.2218	0.225586
7	0.1958	0.193359
8	0.1200	0.120850
9	0.0548	0.053711
10	0.0158	0.016113
11	0.0034	0.002930
12	0.0004	0.000244

In [21]:
```
# DataFrame にして表示
RF_S=pd.Series(RF)    # pandas の配列に変換
BinProb_S=pd.Series(BinProb)
Comparison=pd.concat([RF_S, BinProb_S],axis=1)
Comparison.columns=['Simulation','Theory']
Comparison
```

Out[21]:

	Simulation	Theory
0	0.0010	0.000244
1	0.0030	0.002930
2	0.0144	0.016113
3	0.0500	0.053711
4	0.1202	0.120850
5	0.1994	0.193359
6	0.2218	0.225586
7	0.1958	0.193359
8	0.1200	0.120850
9	0.0548	0.053711
10	0.0158	0.016113
11	0.0034	0.002930
12	0.0004	0.000244

In [22]:
```
Comparison['Difference']=Comparison['Simulation']-Comparison['Theory']
```

In [23]:
```
Comparison
```

Out[23]:

	Simulation	Theory	Difference
0	0.0010	0.000244	0.000756
1	0.0030	0.002930	0.000070
2	0.0144	0.016113	-0.001713
3	0.0500	0.053711	-0.003711
4	0.1202	0.120850	-0.000650
5	0.1994	0.193359	0.006041
6	0.2218	0.225586	-0.003786
7	0.1958	0.193359	0.002441
8	0.1200	0.120850	-0.000850
9	0.0548	0.053711	0.001089
10	0.0158	0.016113	-0.000313
11	0.0034	0.002930	0.000470
12	0.0004	0.000244	0.000156

In [ ]:
```
```