In [1]:

```
# Бондарев И. М., гр. 853501
```

In [2]:

```
import numpy as np
from matplotlib import pyplot as plt

from scipy.special import kolmogorov

from scipy import stats
```

In [3]:

```
n = 3
m = 3
X = [1, 2, 3]
Y = [1, 2, 3]
P = [[0.05, 0.1, 0.05],
     [0.05, 0.2, 0.05],
     [0.1, 0.3, 0.1]]
```

In [4]:

```
q = np.sum(P, axis=1)
l = np.cumsum(q)

def rand():
    rv_X = np.random.uniform()
    k = np.searchsorted(l, rv_X)
    r = np.cumsum(P[k]) / q[k]
    rv_Y = np.random.uniform()
    s = np.searchsorted(r, rv_Y)
    return X[k], Y[s]
```

In [5]:

```
rand()
```

Out[5]:

```
(1, 1)
```

In [6]:

```
N = 10000
samples = [rand() for i in range(N)]
```

In [7]:

```
P_empirical = np.zeros((n, m))


for item in samples:
    P_empirical[item[0] - 1][item[1] - 1] += 1

P_empirical /= N

P_empirical
```

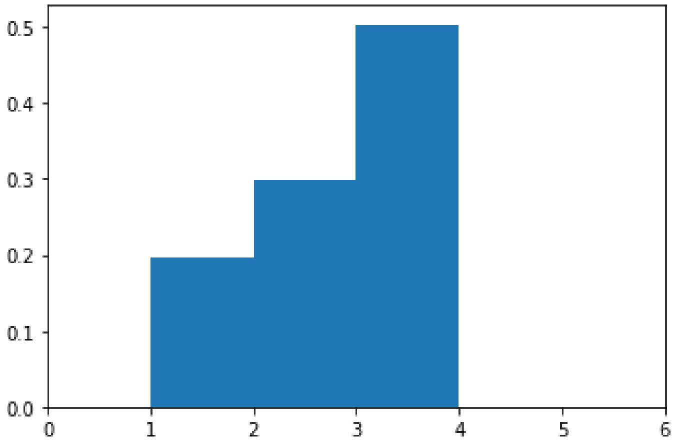
Out[7]:

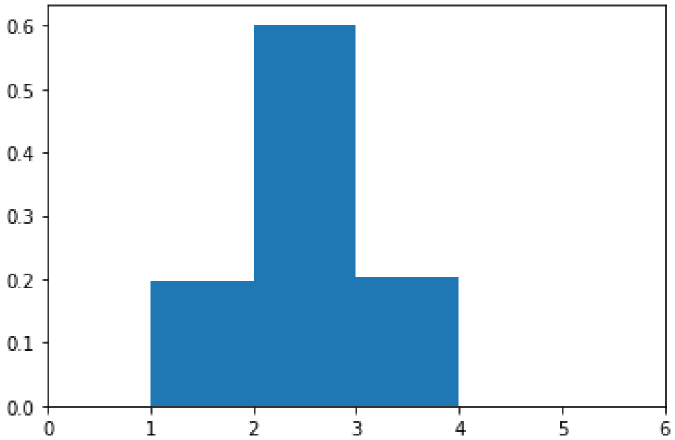
```
array([[0.051 , 0.0956, 0.0509],
       [0.0484, 0.2031, 0.0472],
       [0.0967, 0.3033, 0.1038]])
```

In [8]:

```python
def plot_hist(points, weights):
    plt.hist(points, bins=list(points) + [points[-1] + 1], weights=weights, align='mid',
histtype='stepfilled')
    plt.xlim(0, n + m)
    plt.show()


plot_hist(X, P_empirical.sum(axis=1))
plot_hist(Y, P_empirical.sum(axis=0))
```





In [9]:

```python
P_x_t = np.sum(P, axis=1)
P_y_t = np.sum(P, axis=0)

M_x = 0
M_y = 0

for i in range(n):
    M_x += P_x_t[i] * X[i]

for j in range(m):
    M_y += P_y_t[j] * Y[j]


print(f'Theoretical M_x = {M_x}, M_y = {M_y}')
```

```
Theoretical M_x = 2.3, M_y = 2.0
```

In [10]:

```python
def get_M_Emp(X):
    return X.mean()

def get_D_Emp(X):
    m_x = X.mean()
    return ((X - m_x) ** 2 / (len(X) - 1)).sum()
```

In [11]:

```python
samples_X = np.array(np.squeeze(np.matrix(samples)[:, 0]))[0]
```

```
samples_X = np.array(np.squeeze(np.matrix(samples)[:, 0]))[0]
samples_Y = np.array(np.squeeze(np.matrix(samples)[:, 1]))[0]

M_x_e = get_M_Emp(samples_X)
M_y_e = get_M_Emp(samples_Y)

print(f'Empirical M_x = {M_x_e}, M_y = {M_y_e}')
```

```
Empirical M_x = 2.3063, M_y = 2.0058
```

In [12]:

```
D_x = 0
D_y = 0

for i in range(n):
    D_x += P_x_t[i] * (X[i] - M_x)**2

for j in range(m):
    D_y += P_y_t[j] * (Y[j] - M_y)**2


print(f'Theoretical D_x = {D_x}, D_y = {D_y}')
```

```
Theoretical D_x = 0.61, D_y = 0.4
```

In [13]:

```
D_x_e = get_D_Emp(samples_X)
D_y_e = get_D_Emp(samples_Y)

print(f'Empirical D_x = {D_x_e}, D_y = {D_y_e}')
```

```
Empirical D_x = 0.6075410641064106, D_y = 0.39800616061606164
```

In [14]:

```
def M_interval(M, D, n, p):
    alpha = 1 - p
    return M - stats.norm.ppf(1 - alpha / 2) * np.sqrt(D) / np.sqrt(n), M + stats.norm.p
pf(1 - alpha / 2) * np.sqrt(D) / np.sqrt(n)

def D_interval(M, D, n, p):
    alpha = 1 - p
    return (n - 1) * D / stats.chi2(n - 1).ppf(1 - alpha / 2), (n - 1) * D / stats.chi2
(n - 1).ppf(alpha / 2)
```

In [15]:

```
M_x_interval = M_interval(M_x_e, D_x_e, N, 0.95)
M_y_interval = M_interval(M_y_e, D_y_e, N, 0.95)

print(f'Interval: M_x = {M_x_interval}, Interval: M_y = {M_y_interval}')
```

```
Interval: M_x = (2.2910230762925075, 2.321576923707492), Interval: M_y = (1.9934350322427
06, 2.0181649677572935)
```

In [16]:

```
D_x_interval = D_interval(M_x_e, D_x_e, N, 0.95)
D_y_interval = D_interval(M_y_e, D_y_e, N, 0.95)

print(f'Interval: D_x = {D_x_interval}, Interval: D_y = {D_y_interval}')
```

```
Interval: D_x = (0.5910459050537809, 0.6247398583102958), Interval: D_y = (0.387200018758
071, 0.4092732608216786)
```

In [17]:

```
def get_R(X, M_x, Y, M_y, D_x, D_y):
    numerator = 0
    for i in range(N):
```

```python
        numerator += (X[i] - M_x) * (Y[i] - M_y)

    return numerator / (np.sqrt(D_x * D_y * ((N - 1) ** 2)))

R = get_R(samples_X, M_x, samples_Y, M_y, D_x, D_y)
print(f'R = {R}')
```

```
R = 0.011054552362178428
```

In [18]:

```python
def test_chi_square(alpha=0.05):
    statistics = len(P) * np.sum(np.nan_to_num((P - P_empirical) ** 2 / P))
    p_value = 1 - stats.chi2.cdf(statistics, len(P) - 1)
    return statistics, p_value, p_value > alpha


test_chi_square()
```

Out[18]:

```
(0.002326350000000004, 0.9988375012258253, True)
```