Содержание

Основные понятия математической статистики

In [28]:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import maxwell
import scipy.stats as sts
from scipy.stats import geom
#from random import random
from collections import Counter
#import copy
#import math
#from math import *
from random import *
import pandas as pd
#import calendar
#import statsmodels.api as sm

plt.style.use('ggplot')  # Красивые графики
plt.rcParams['figure.figsize'] = (15, 5)  # Размер картинок
```

2.1 Геометрическое распределение

2.1.1 Моделирование выбранных случайных величин

In [3]:

```
# Создание случайной величины с геометрическим распределением, зависящим
# от параметра p
p = 0.5
geom_rv = sts.geom(p)
```

In [4]:

```
#Генерация выборки объема n = 5 с выводом
for n in [5]:
    means_5 = []
    for i in range(5):
        sample = geom_rv.rvs(n)
        means_5.append(sample)
        print(sample)
```

```
[1 4 1 2 1]
[1 2 2 1 2]
[1 1 1 1 3]
[3 1 1 1 3]
[4 3 2 3 1]
```

In [5]:

```
#Генерация выборки объема n = 10 с выводом
for n in [10]:
    means_10 = []
    for i in range(5):
        sample = geom_rv.rvs(n)
        means_10.append(sample)
        print(sample)
```

```
[1 7 4 2 1 2 1 2 2 3]
[2 3 2 1 3 3 1 3 1 2]
[1 1 5 2 1 4 1 2 2 1]
[1 1 1 1 1 1 2 3 1 2]
[ 2  2  1  1  4  3 10  2  3  7]
```

In [6]:

```
#Генерация выборки объема n = 100 ,без вывода
for n in [100]:
    means_100 = []
    for i in range(5):
        sample = geom_rv.rvs(n)
        means_100.append(sample)
```

In [7]:

```
#Генерация выборки объема n = 1000 ,без вывода
for n in [1000]:
    means_1000 = []
    for i in range(5):
        sample = geom_rv.rvs(n)
        means_1000.append(sample)
```

In [8]:

```
#Генерация выборки объема n = 100000 ,без вывода
for n in [100000]:
    means_100000 = []
    for i in range(5):
        sample = geom_rv.rvs(n)
        means_100000.append(sample)
```

2.1.2 Построение эмпирической функции распределения

In [9]:

```
#n=5
for a in range(5):
    b=means_5[a]
    b=sorted(b)
    print('Empirical distribution function F5(x) for sample',a+1,':')
    for i in range(4):
        if(i==0):
            n=0.
            g=1
            print(n,', x <=',b[i])
        if(b[i+1]==b[i]):
            g+=1
        else:
            n=round(n+0.2*g,1)
            g=1
            print(n,',',b[i],'< x <=',b[i+1])
        if(i==3):
            n=1.
            print(n,', x >',b[i+1])
```

```
Empirical distribution function F5(x) for sample 1 :
0.0 , x <= 1
0.6 , 1 < x <= 2
0.8 , 2 < x <= 4
1.0 , x > 4
Empirical distribution function F5(x) for sample 2 :
0.0 , x <= 1
0.4 , 1 < x <= 2
1.0 , x > 2
Empirical distribution function F5(x) for sample 3 :
0.0 , x <= 1
0.8 , 1 < x <= 3
1.0 , x > 3
Empirical distribution function F5(x) for sample 4 :
0.0 , x <= 1
0.6 , 1 < x <= 3
1.0 , x > 3
```

In [10]:

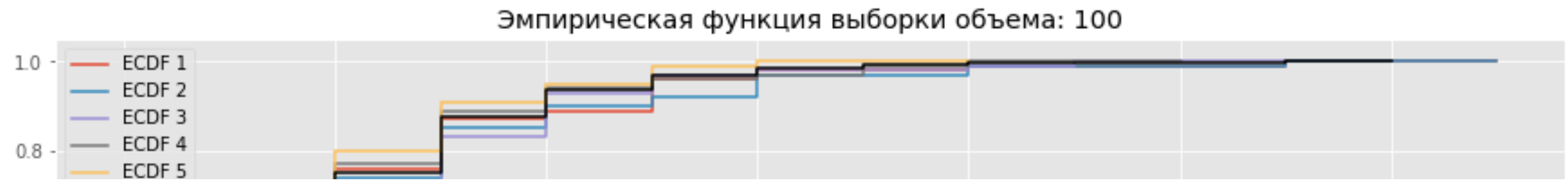
```
#n=5
for a in range(5):
    b=means_5[a]
    b=sorted(b)
    v=len(b)
    N=[]
    for i in range(b[v-1]):
        N.append(b.count(i))
        x1=[]
        y1=[]
        t=0
    for i in range(b[v-1]):
        t+=N[i]
        x1.append(i)
        y1.append(t/v)
        x1.append(i+1)
        y1.append(t/v)
    x1.append(b[v-1])
    y1.append(1)
    x1.append(b[v-1]+2)
    y1.append(1)
    plt.plot(x1,y1,label="ECDF "+str(a+1))
    plt.legend(loc='lower right')
plt.title("Эмпирическая функция выборки объема: "+str(v))
n=np.arange(1,8,1)#Построение
plt.step(n,1-(1-p)**(n-1),'k-', label='CDF')#теоретическойфункции
plt.legend()#распределения
plt.xlabel("numbers")
plt.ylabel("probability")
plt.show()
```

In [11]:

```
#n=10
for a in range(5):
    b=means_10[a]
    b=sorted(b)
    v=len(b)
    N=[]
    for i in range(b[v-1]):
        N.append(b.count(i))
        x2=[]
        y2=[]
        t=0
    for i in range(b[v-1]):
        t+=N[i]
        x2.append(i)
        y2.append(t/v)
        x2.append(i+1)
        y2.append(t/v)
    x2.append(b[v-1])
    y2.append(1)
    x2.append(b[v-1]+2)
    y2.append(1)
    plt.plot(x2,y2,label="ECDF "+str(a+1))
    plt.legend(loc='lower right')
plt.title("Эмпирическая функция выборки объема: "+str(v))
n=np.arange(1,8,1)#Построение
plt.step(n,1-(1-p)**(n-1),'k-', label='CDF')#теоретическойфункции
plt.legend()#распределения
plt.xlabel("numbers")
plt.ylabel("probability")
plt.show()
```

In [12]:

```
#n=100
for a in range(5):
    b=means_100[a]
    b=sorted(b)
    v=len(b)
    N=[]
    for i in range(b[v-1]):
        N.append(b.count(i))
        x3=[]
        y3=[]
        t=0
    for i in range(b[v-1]):
        t+=N[i]
        x3.append(i)
        y3.append(t/v)
        x3.append(i+1)
        y3.append(t/v)
    x3.append(b[v-1])
    y3.append(1)
    x3.append(b[v-1]+2)
    y3.append(1)
    plt.plot(x3,y3,label="ECDF "+str(a+1))
    plt.legend(loc='lower right')
plt.title("Эмпирическая функция выборки объема: "+str(v))
n=np.arange(1,13,1)#Построение
plt.step(n,1-(1-p)**(n-1),'k-', label='CDF')#теоретическойфункции
plt.legend()#распределения
plt.xlabel("numbers")
plt.ylabel("probability")
plt.show()
```



In [13]:

```
#n=1000
for a in range(5):
    b=means_1000[a]
    b=sorted(b)
    v=len(b)
    N=[]
    for i in range(b[v-1]):
        N.append(b.count(i))
        x4=[]
        y4=[]
        t=0
    for i in range(b[v-1]):
        t+=N[i]
        x4.append(i)
        y4.append(t/v)
        x4.append(i+1)
        y4.append(t/v)
    x4.append(b[v-1])
    y4.append(1)
    x4.append(b[v-1]+2)
    y4.append(1)
    plt.plot(x4,y4,label="ECDF "+str(a+1))
    plt.legend(loc='lower right')
plt.title("Эмпирическая функция выборки объема: "+str(v))
n=np.arange(1,18,1)#Построение
plt.step(n,1-(1-p)**(n-1), 'k-', label='CDF')#теоретическойфункции
plt.legend()#распределения
plt.xlabel("numbers")
plt.ylabel("probability")
plt.show()
```

In [14]:

```
#n=100000
for a in range(5):
    b=means_100000[a]
    b=sorted(b)
    v=len(b)
    N=[]
    for i in range(b[v-1]):
        N.append(b.count(i))
        x5=[]
        y5=[]
        t=0
    for i in range(b[v-1]):
        t+=N[i]
        x5.append(i)
        y5.append(t/v)
        x5.append(i+1)
        y5.append(t/v)
    x5.append(b[v-1])
    y5.append(1)
    x5.append(b[v-1]+2)
    y5.append(1)
    plt.plot(x5,y5,label="ECDF "+str(a+1))
    plt.legend(loc='lower right')
plt.title("Эмпирическая функция выборки объема: "+str(v))
n=np.arange(1,25,1)#Построение
plt.step(n,1-(1-p)**(n-1), 'k-', label='CDF')#теоретическойфункции
plt.legend()#распределения
plt.xlabel("numbers")
plt.ylabel("probability")
plt.show()
```

Пусть $X = (X_1, \dots, X_n)$- выборка из дискретного распределения $\sigma(\xi)$ Величина скачка в точке j есть

$$\Delta\hat{F}_n(j) = \hat{F}_n(j) - \hat{F}_n(j-0) = \frac{\nu}{n},$$

$j = 1, \dots, N$

Здесь $P\{\Delta\dot{F}_n(j) = 0\} = P(\nu_j = 0) = (1-p_j)^n$ что мало при больших n, т.е. в большой выборке скачок в точке j наверняка будет иметь место. Более того, так как $P\{\cup_{j=1}^N\{\Delta\hat{F}_n(j) = 0\}\} \le \sum_{j=1}^N(1-p_j)^n \to 0,$ при $n \to \infty$, то в больших выборках с вероятностью, близкой к 1, скачки э.ф.р. $F_n(x)$ будут иметь место во всех точках 1,2,...,N, а случайными будут лишь величины этих скачков.

Если же теоретическая функция распределения $F_\xi = F(x)$ непрерывна, то с вероятностью 1 все элементы выборки $X = (X_1, \dots, X_n)$ будут различны, и случайными теперь будут точки скачков, величины же скачков неслучайны и равны $\frac{1}{n}$

Таким образом, для выборок из дискретных и непрерывных распределений характер соответствующих эмпирических функций распределениябудет различным, что можно заметить на получившихся графиков для дискретного и непрерывного распределения. Тем не менее в любомслучае э.ф.р. $\hat{F}_n(x)$ с увеличением объема выборки n сближается в каждой точке x с теоретической функцией распределения F(x).

Максимальная точная верхняя граница разности пары эмпирических функций распределения - наибольшая разность между значениями вероятности двух функций в одной точке.

Верхняя граница разности э.ф.р. выборок размера n = 5 : 0.600
Верхняя граница разности э.ф.р. выборок размера n = 10 : 0.400
Верхняя граница разности э.ф.р. выборок размера n = 100 : 0.17
Верхняя граница разности э.ф.р. выборок размера n = 1000 : 0.061
Верхняя граница разности э.ф.р. выборок размера n = 100000 : 0.0049
С увеличением объема выборки верхняя граница разности уменьшается, что, впрочем, очевидно.

2.1.3 Построение вариационного ряда выборки

Определение:
Пусть $X = (X_1, \ldots, X_n)$- выборка из некоторого распределения $\sigma(\xi)$
Произвольной реализации $x = (x_1, \ldots, x_n)$ этой выборки можно поставить в соответствие упорядоченную последовательность

$$x_{(1)} \leq x_{(2)} \leq \ldots \leq x_{(n)}$$

располагая $x_1, \ldots, x_n$ в порядке их возрастания, так что $x_{(1)} = min\{x_1, \ldots, x_n\}, x_{(2)}$ - второе по величине значение, $x_{(n)} = max\{x_1, \ldots, x_n\}$
Обозначим через $X_{(k)}$ случайную величину, которая для каждой реализации выборки X принимает значение $x_{(k)}, k = 1, \ldots, n$. Так по выборке X определяют новую последовательность случайных величин $X_{(1)}, \ldots, X_{(n)}$, называемых $порядковыми\ статистиками$ выборки. Из определения порядковых статистик следует, что они упорядочены по возрастанию их значений, т.е. они образуют возрастающую последовательность

$$X_{(1)} \leq X_{(2)} \leq \ldots \leq X_{(n)},$$

которая называется $вариационным\ рядом$ выборки X.

In [15]:
```
#Вариационный ряд для выборки объема n=5 с выводом
for a in range(5):
    b=means_5[a]
    b=sorted(b)
    print(b)
```

```
[1, 1, 1, 2, 4]
[1, 1, 2, 2, 2]
[1, 1, 1, 1, 3]
[1, 1, 1, 3, 3]
[1, 2, 3, 3, 4]
```

In [16]:

```
#Вариационный ряд для выборки объема n=10 с выводом
for a in range(5):
    b=means_10[a]
    b=sorted(b)
    print(b)
```

```
[1, 1, 1, 2, 2, 2, 2, 3, 4, 7]
[1, 1, 1, 2, 2, 2, 3, 3, 3, 3]
[1, 1, 1, 1, 1, 2, 2, 2, 4, 5]
[1, 1, 1, 1, 1, 1, 1, 2, 2, 3]
[1, 1, 2, 2, 2, 3, 3, 4, 7, 10]
```

In [17]:

```
#Вариационный ряд для выборки объема n=100 без вывода
for a in range(5):
    b=means_100[a]
    b=sorted(b)
```

In [18]:

```
#Вариационный ряд для выборки объема n=1000 без вывода
for a in range(5):
    b=means_1000[a]
    b=sorted(b)
```

In [19]:

```
#Вариационный ряд для выборки объема n=100000 без вывода
for a in range(5):
    b=means_100000[a]
    b=sorted(b)
```

Определение:
α- квантиль случайной величины ξ с функцией распределения $F(x) = P\{\xi < x\}$ — это любое число x_α, удовлетворяющее двум условиям:
$1) F(x_\alpha) \leq \alpha \; 2) F(x_\alpha + 0) \geq \alpha.$
Исходя из того, что при больших выборках э.ф.р. стремится к теоритической функции распределения, эмпирические квантили так же стремятся к теоритическим по определению.
Пусть F(x) - функция распределения. Тогда квантильная функция:
$F^{-1}(r) = min\{x \in N_+ : F(x) \geq r\} forr \in (0; 1)$
$F^{-1}(r) = [\frac{ln(1-r)}{ln(1-p}]$

In [20]:

```
k = 1
for b in [means_5[a], means_10[a],means_100[a],means_1000[a],means_100000[a]]:
    if(k==1):
        print('n = 5')
    if(k==2):
        print('n = 10')
    if(k==3):
        print('n = 100')
    if(k==4):
        print('n = 1000')
    if(k==5):
        print('n = 100000')
    print(np.quantile(b, 0.1))
    k += 1
```

```
n = 5
1.4
n = 10
1.0
n = 100
1.0
n = 1000
1.0
n = 100000
1.0
```

In [21]:

```
#Сравнение
np.log(1-0.1)//np.log(1-p)
```

Out[21]:

```
0.0
```

In [22]:

```
k = 1
for b in [means_5[a], means_10[a],means_100[a],means_1000[a],means_100000[a]]:
    if(k==1):
        print('n = 5')
    if(k==2):
        print('n = 10')
    if(k==3):
        print('n = 100')
    if(k==4):
        print('n = 1000')
    if(k==5):
        print('n = 100000')
    print(np.quantile(b, 0.5))
    k += 1
```

```
n = 5
3.0
n = 10
2.5
n = 100
1.0
n = 1000
2.0
n = 100000
1.0
```

In [23]:

```
#Сравнение
geom.median(p)
```

Out[23]: 1.0

In [24]:

```
#Сравнение
np.log(1-0.5)//np.log(1-p)
```

Out[24]: 1.0

In [25]:

```
k = 1
for b in [means_5[a], means_10[a],means_100[a],means_1000[a],means_100000[a]]:
    if(k==1):
        print('n = 5')
    if(k==2):
        print('n = 10')
    if(k==3):
        print('n = 100')
    if(k==4):
        print('n = 1000')
    if(k==5):
        print('n = 100000')
    print(np.quantile(b, 0.7))
    k += 1
```

```
n = 5
3.0
n = 10
3.3
n = 100
2.0
n = 1000
2.0
n = 100000
2.0
```

In [26]:

```
#Сравнение
np.log(1-0.7)//np.log(1-p)
```

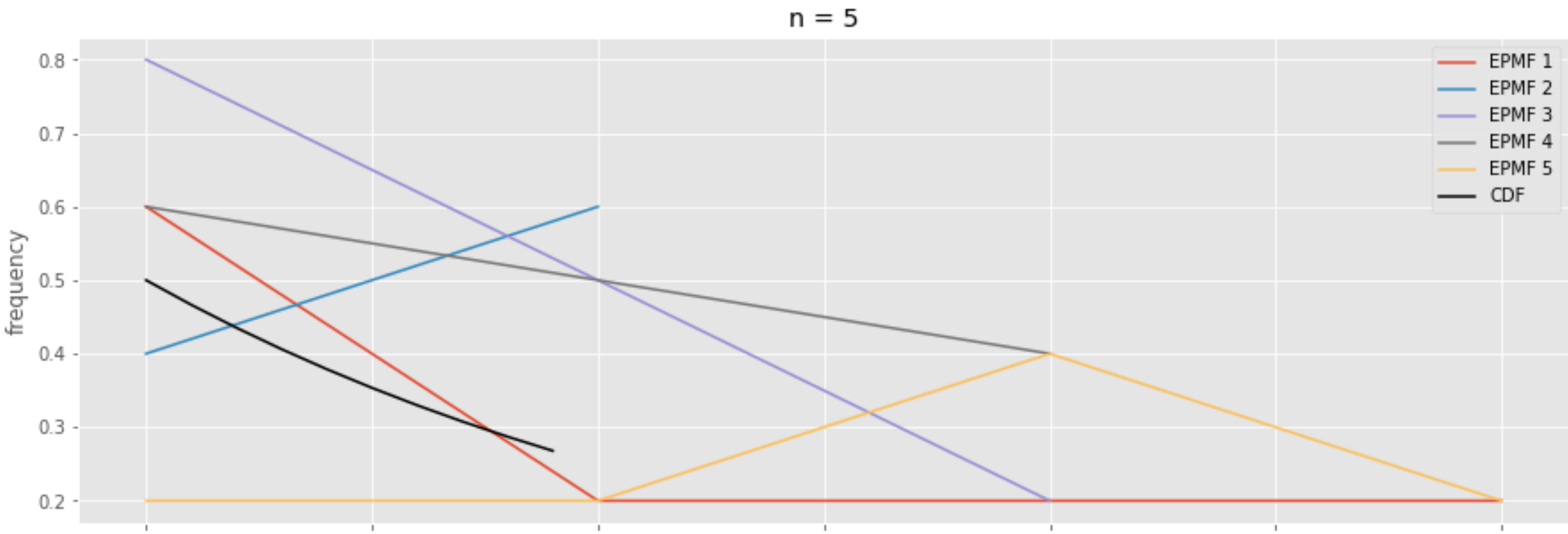
Out[26]:

```
1.0
```

2.1.4 Построение гистограммы и полигона частот

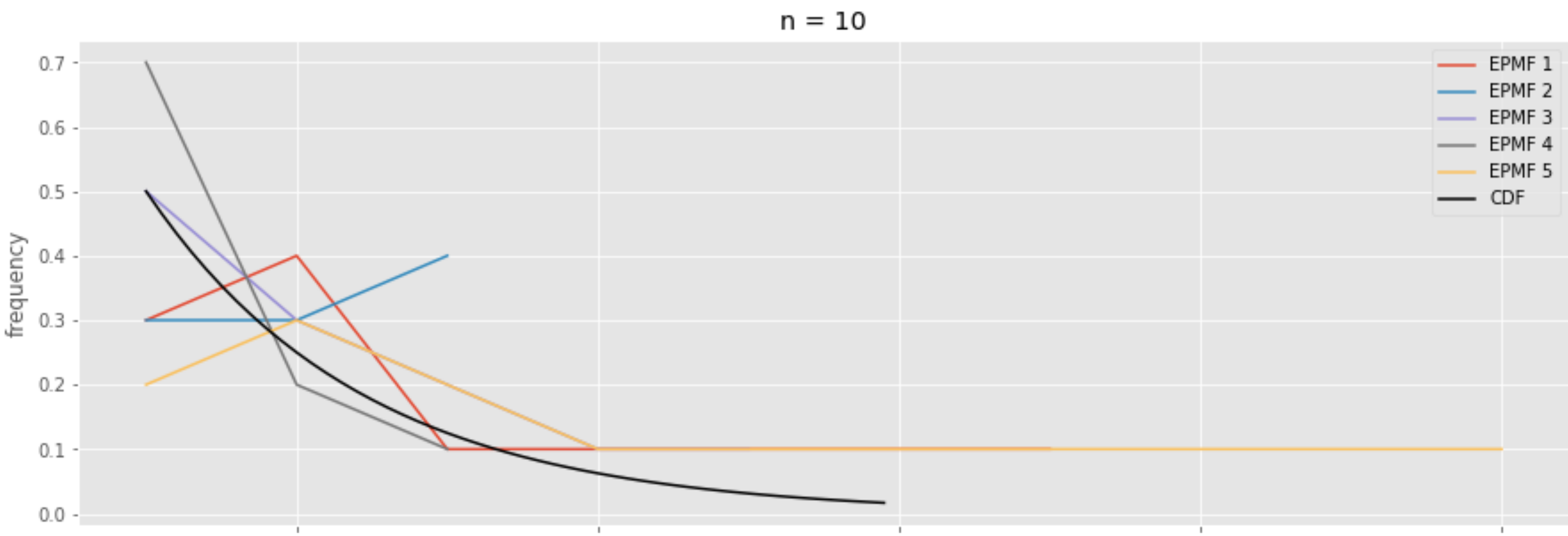
In [29]:

```python
#n=5
for a in range(5):
    b=means_5[a]
    b=sorted(b)
    x=[]
    y=[]
    c=Counter(b)
    for i in c:
        x.append(i)
        y.append(b.count(i)/5.0)
    plt.plot(x,y,label="EPMF "+str(a+1))
    plt.legend(loc='lower right')
plt.title("n = 5")
n=np.arange(1,2,0.1)#Построение
plt.plot(n,p*(1-p)**(n-1),'k-',label='CDF')#функции вероятности
plt.legend()#распределения
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
```



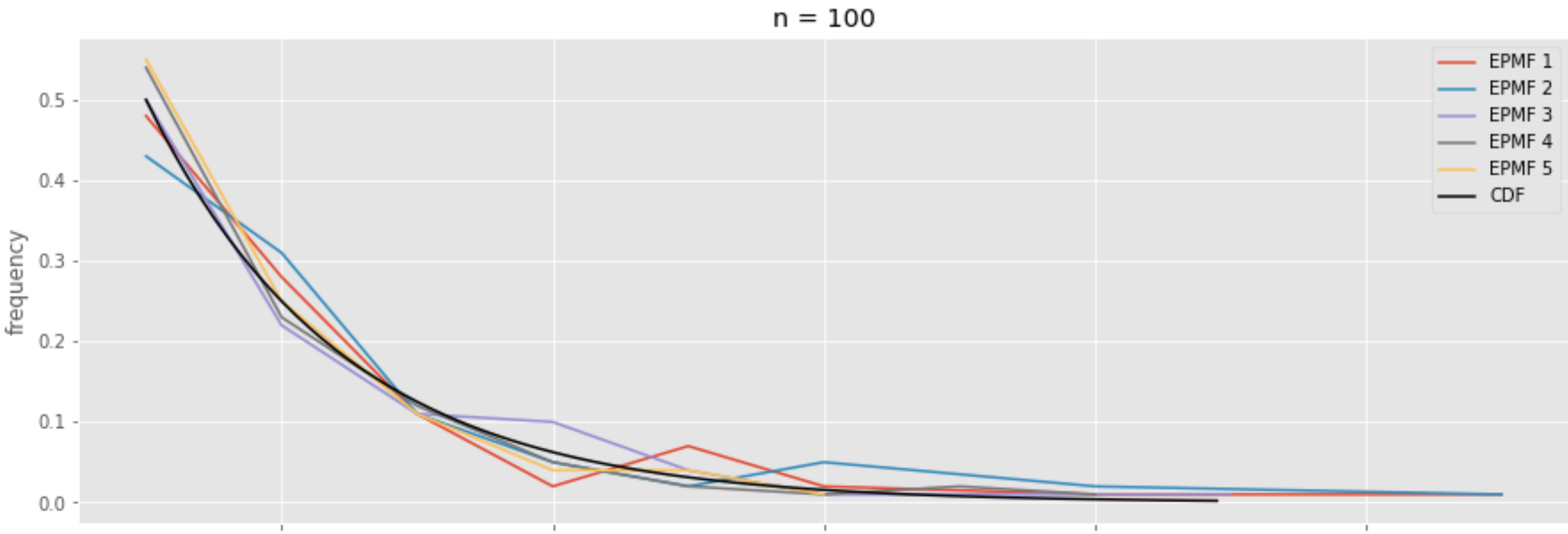
In [30]:

```
#n=10
for a in range(5):
    b=means_10[a]
    b=sorted(b)
    x=[]
    y=[]
    c=Counter(b)
    for i in c:
        x.append(i)
        y.append(b.count(i)/10.0)
    plt.plot(x,y,label="EPMF "+str(a+1))
    plt.legend(loc='lower right')
plt.title("n = 10")
n=np.arange(1,6,0.1)#Построение
plt.plot(n,p*(1-p)**(n-1),'k-',label='CDF')#функции вероятности
plt.legend()#распределения
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
```



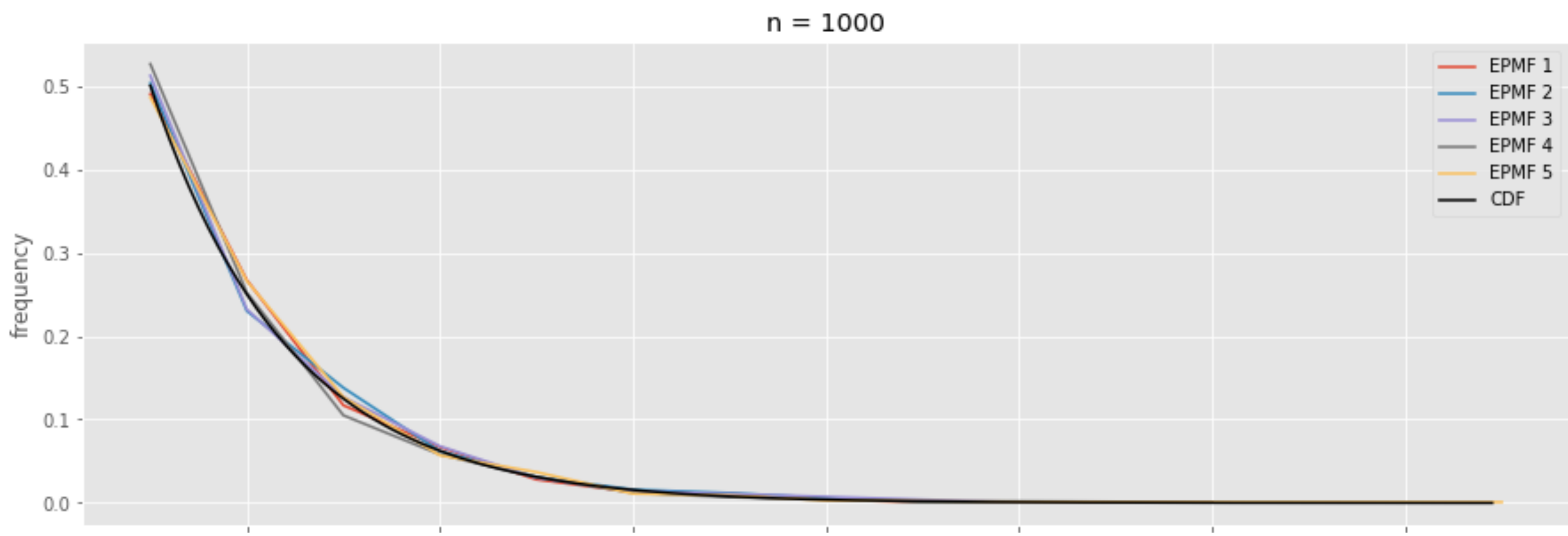
In [31]:

```
#n=100
for a in range(5):
    b=means_100[a]
    b=sorted(b)
    x=[]
    y=[]
    c=Counter(b)
    for i in c:
        x.append(i)
        y.append(b.count(i)/100.0)
    plt.plot(x,y,label="EPMF "+str(a+1))
    plt.legend(loc='lower right')
plt.title("n = 100")
n=np.arange(1,9,0.1)#Построение
plt.plot(n,p*(1-p)**(n-1),'k-',label='CDF')#функции вероятности
plt.legend()#распределения
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
```



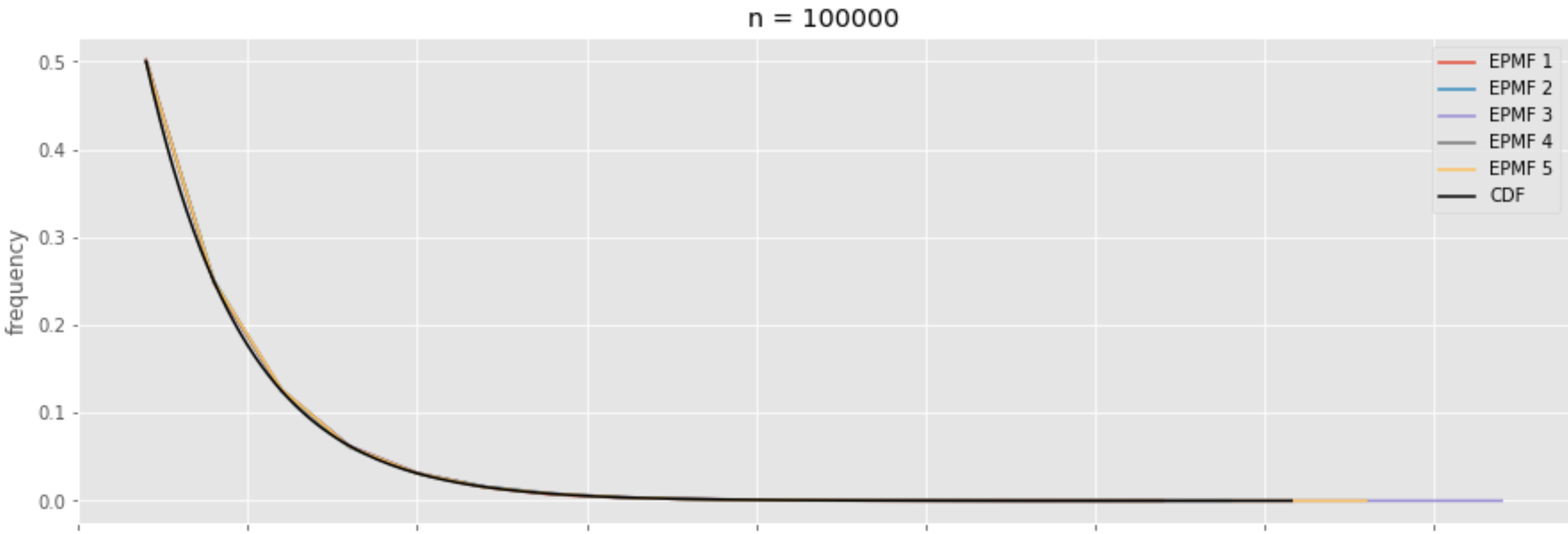
In [32]:

```
#n=1000
for a in range(5):
    b=means_1000[a]
    b=sorted(b)
    x=[]
    y=[]
    c=Counter(b)
    for i in c:
        x.append(i)
        y.append(b.count(i)/1000.0)
    plt.plot(x,y,label="EPMF "+str(a+1))
    plt.legend(loc='lower right')
plt.title("n = 1000")
n=np.arange(1,15,0.1)#Построение
plt.plot(n,p*(1-p)**(n-1),'k-',label='CDF')#функции вероятности
plt.legend()#распределения
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
```



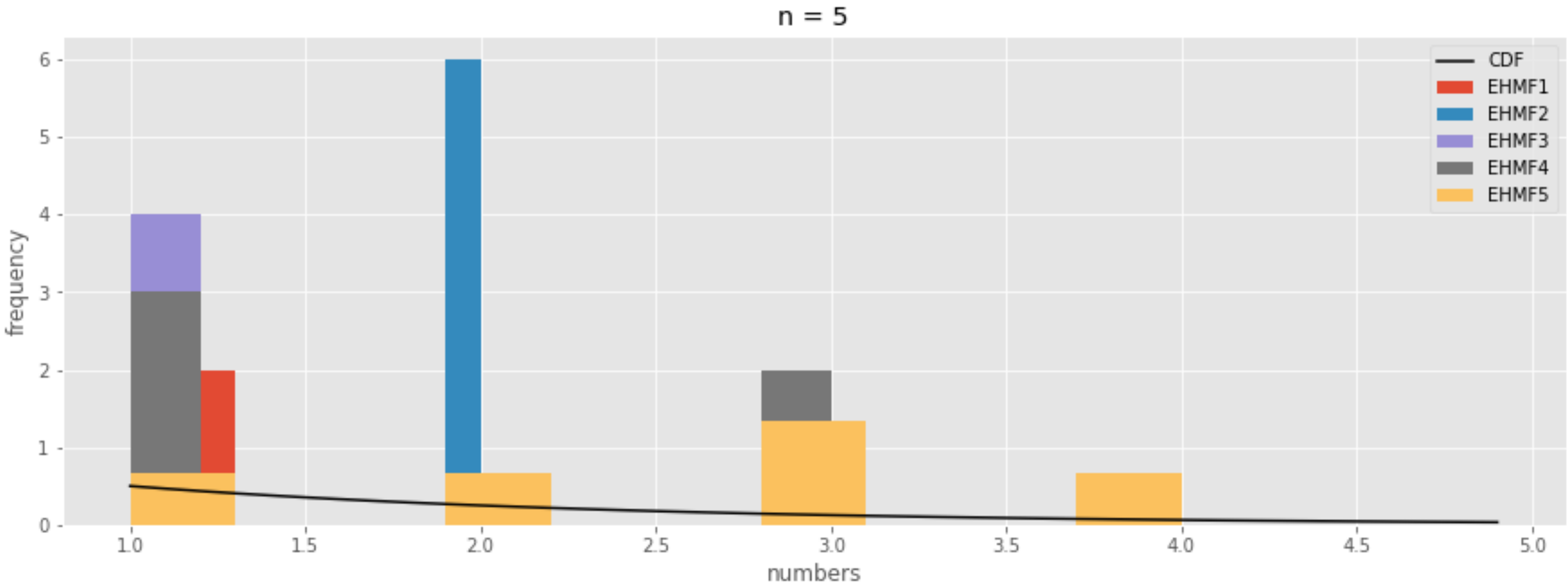
In [33]:

```
#n=100000
for a in range(5):
    b=means_100000[a]
    b=sorted(b)
    x=[]
    y=[]
    c=Counter(b)
    for i in c:
        x.append(i)
        y.append(b.count(i)/100000.0)
    plt.plot(x,y,label="EPMF "+str(a+1))
    plt.legend(loc='lower right')
plt.title("n = 100000")
n=np.arange(1,18,0.1)#Построение
plt.plot(n,p*(1-p)**(n-1),'k-',label='CDF')#функции вероятности
plt.legend()#распределения
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
```



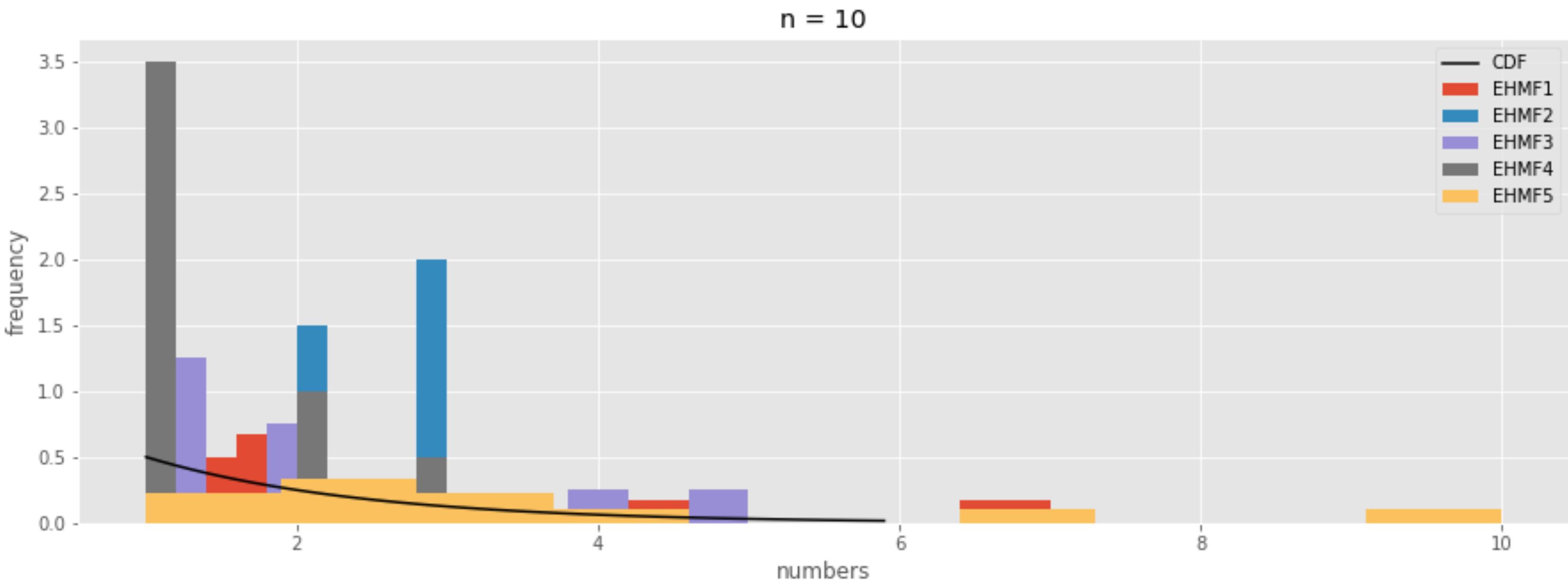
In [34]:

```
#n=5
for a in range(5):
    plt.hist(means_5[a],density=True,label='EHMF{}'.format(a+1))
    plt.legend()
n=np.arange(1,5,0.1)#Построение
plt.plot(n,p*(1-p)**(n-1),'k-',label='CDF')#функции вероятности
plt.legend()#распределения
plt.title("n = 5")
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
```



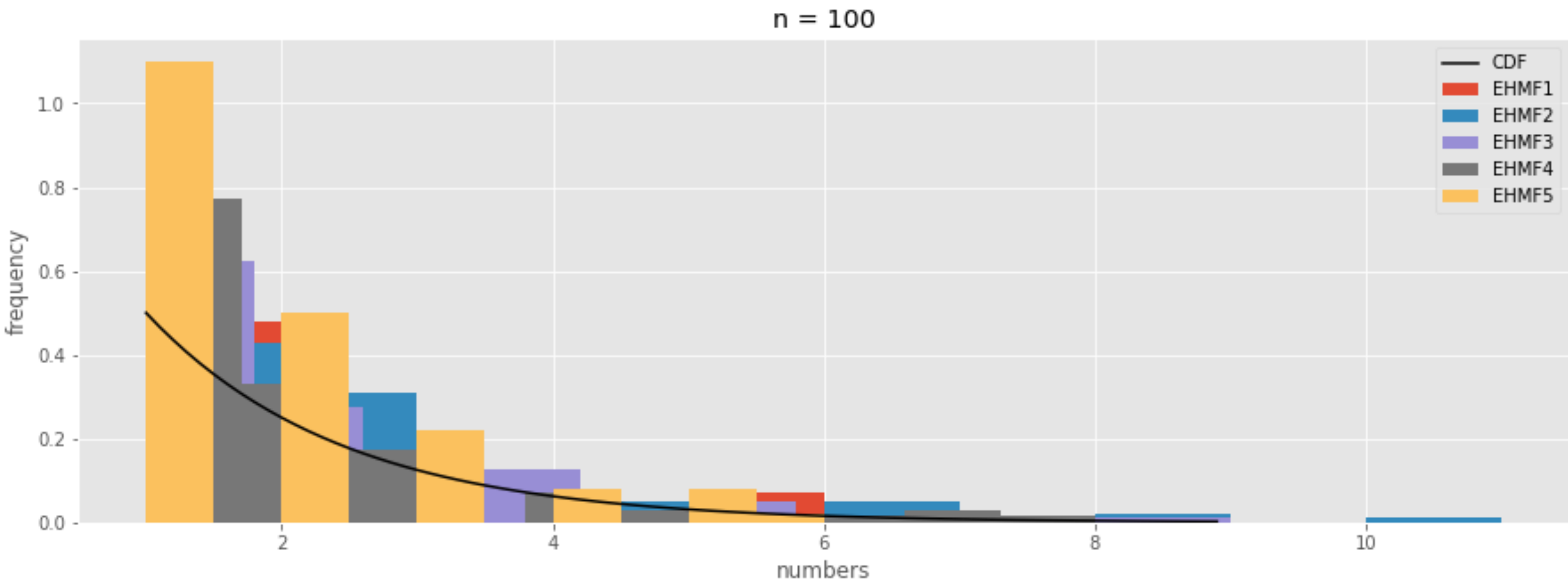
In [35]:

```
#n=10
for a in range(5):
    plt.hist(means_10[a],density=True,label='EHMF{}'.format(a+1))
    plt.legend()
n=np.arange(1,6,0.1)#Построение
plt.plot(n,p*(1-p)**(n-1),'k-',label='CDF')#функции вероятности
plt.legend()#распределения
plt.title("n = 10")
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
```



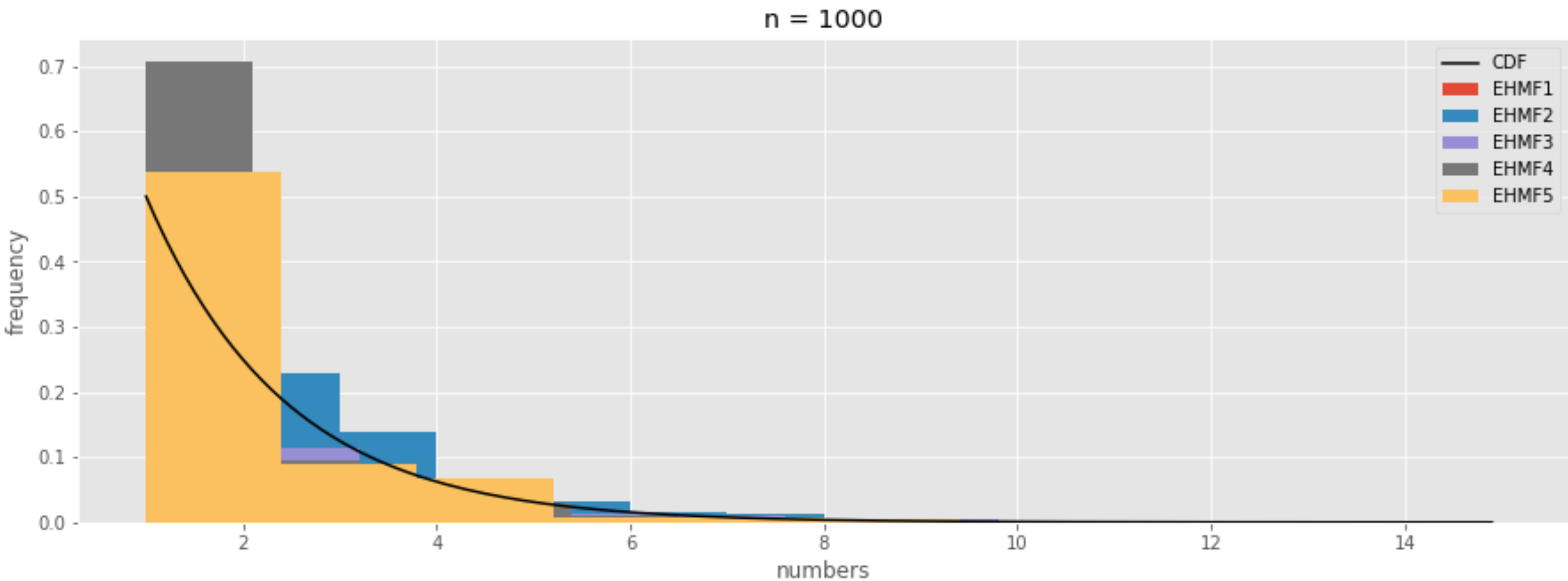
In [36]:

```
#n=100
for a in range(5):
    plt.hist(means_100[a],density=True,label='EHMF{}'.format(a+1))
    plt.legend()
n=np.arange(1,9,0.1)#Построение
plt.plot(n,p*(1-p)**(n-1),'k-',label='CDF')#функции вероятности
plt.legend()#распределения
plt.title("n = 100")
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
```



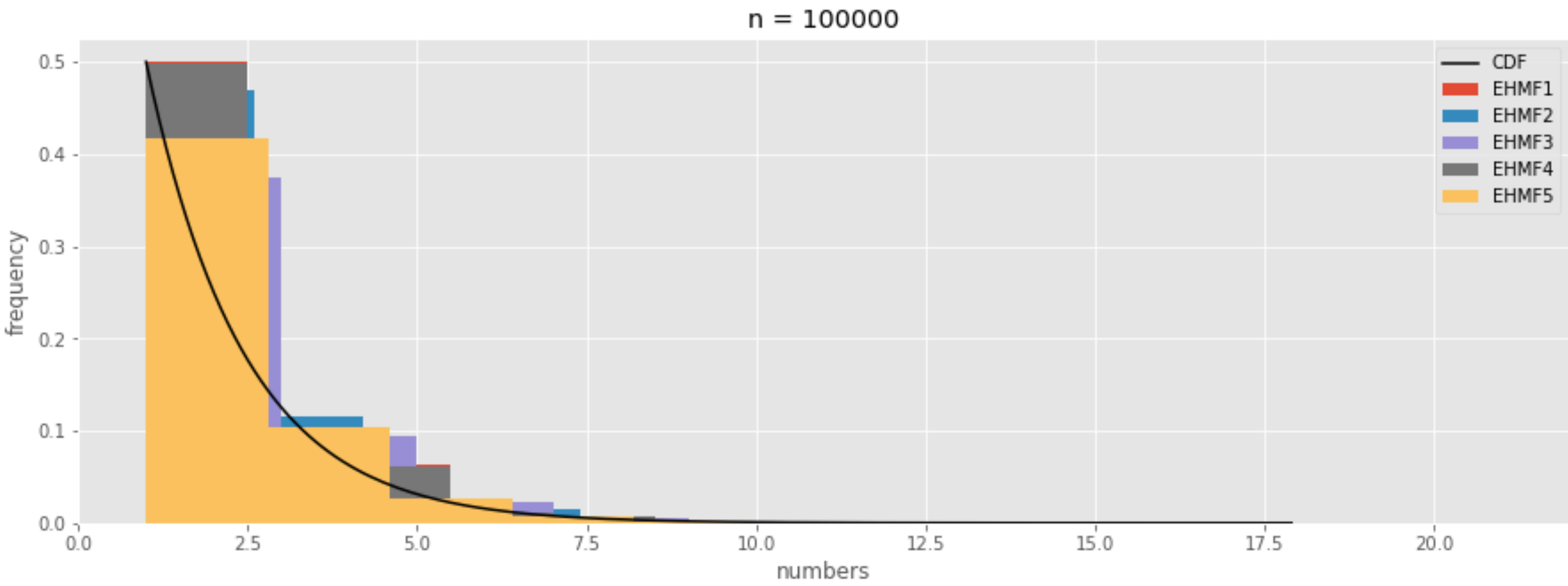
In [37]:

```python
#n=1000
for a in range(5):
    plt.hist(means_1000[a],density=True,label='EHMF{}'.format(a+1))
    plt.legend()
n=np.arange(1,15,0.1)#Построение
plt.plot(n,p*(1-p)**(n-1),'k-',label='CDF')#функции вероятности
plt.legend()#распределения
plt.title("n = 1000")
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
```



In [38]:

```
#n=100000
for a in range(5):
    plt.hist(means_100000[a],density=True,label='EHMF{}'.format(a+1))
    plt.legend()
n=np.arange(1,18,0.1)#Построение
plt.plot(n,p*(1-p)**(n-1),'k-',label='CDF')#функции вероятности
plt.legend()#распределения
plt.title("n = 100000")
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
```



Если наблюдаемая в эксперименте случайная величина ξ дискретна и принимает значения $a_1, a_2, ..$, то более наглядное представление о ее законе распределения дадут относительные частоты $\nu_r^* = \frac{\nu_r}{n}$, где ν_r - число элементов выборки $X = (X_1, \ldots, X_n)$, принявших значение $a_r : \nu_r = \sum_{j=1}^n I(X_j = a_r), r = 1, 2, ..$, т.е. ν_r^* сближается с ростом n с теоретической вероятностью $P\{\xi = a_r\}$, и потому, по крайней мере для больших выборок, относительные частоты ν_r^* можно рассматривать в качестве приближенных значений (оценок) для неизвестных вероятностей $P\{\xi = a_r\}$.
Наглядным представлением данных является полигон частот, который представляет собой ломаную с вершинами в точках $(a_r; \nu_r), r = 1, 2, \ldots$
Можно рассматривать также статистический ряд {{$(a_r; \nu_r)$}}.
На графиках выше наглядно подтверждаются наши теоретические знания.

2.2 Распределение Максвелла

2.2.1 Моделирование выбранных случайных величин

In [39]:
```
# Создание случайной величины с распределением Максвелла, зависящим
# от параметра lambda
lambd=1.0
maxwell_rv=sts.maxwell(scale=lambd)
```

In [40]:

```
#Генерация выборки объема n = 5 с выводом
for n in[5]:
    means__5=[]
    for i in range(5):
        sample=maxwell_rv.rvs(n)
        means__5.append(sample)
        print(sample)
```

```
[2.85963449 0.26823227 1.02528533 1.38912998 1.94808635]
[1.61895478 1.94273306 1.10185976 0.71156656 0.78448761]
[2.06143807 1.22665031 0.3648358  0.1455319  1.34925294]
[3.06609579 2.00054473 1.94926054 1.79530409 0.61857771]
[1.28424246 1.14887368 1.33076897 0.95321557 1.88883784]
```

In [41]:

```
#Генерация выборки объема n = 10 с выводом
for n in[10]:
    means__10=[]
    for i in range(5):
        sample=maxwell_rv.rvs(n)
        means__10.append(sample)
        print(sample)
```

```
[1.86580351 3.08168615 1.28488923 0.57498455 1.09721514 0.7074752
 2.31672318 1.94965621 1.59190871 2.15145171]
[0.78930587 0.87672893 1.05604767 1.21079832 0.84314466 1.31206446
 0.73987739 1.6164658  2.05973467 0.82250845]
[0.83671771 1.70480997 2.54945558 2.05252014 0.76078221 0.3697578
 2.14176662 2.13027461 1.88533054 0.570532  ]
[2.11559546 1.04763887 1.63568796 2.39081196 3.16153799 1.10702455
 2.24797022 1.31088133 1.09330636 1.34882989]
[1.45948172 1.01536527 1.18815338 0.49157912 1.12197421 1.75195823
 1.12485663 1.10302412 1.37487594 2.14866571]
```

In [42]:
```
#Генерация выборки объема n = 100 без вывода
for n in[100]:
    means__100=[]
    for i in range(5):
        sample=maxwell_rv.rvs(n)
        means__100.append(sample)
```

In [43]:
```
#Генерация выборки объема n = 1000 без вывода
for n in[1000]:
    means__1000=[]
    for i in range(5):
        sample=maxwell_rv.rvs(n)
        means__1000.append(sample)
```

In [44]:
```
#Генерация выборки объема n = 100000 без вывода
for n in[100000]:
    means__100000=[]
    for i in range(5):
        sample=maxwell_rv.rvs(n)
        means__100000.append(sample)
```

In [45]:
```
#Вернёмся к медиане и убедимся, что в пункте 1.2.1 она была найдена верно
maxwell.median()
```

Out[45]:
```
1.5381722544550522
```

2.2.2 Построение эмпирической функции распределения

In [46]:

```
#n=5
for a in range(5):
    b=means__5[a]
    b=sorted(b)
    v=len(b)
    N=[]
    for i in range((v-1)):
        N.append(b.count(b[i]))
        x=[]
        y=[]
        t=0
        x.append(0.0)
        y.append(0.0)
        x.append(b[0])
        y.append(0.0)
    for i in range((v-1)):
        t+=N[i]
        x.append(b[i])
        y.append(float(t/v))
        x.append(b[i+1])
        y.append(float(t/v))
    x.append(b[v-1])
    y.append(1)
    x.append(b[v-1]+2)
    y.append(1)
    plt.plot(x,y,label="ECDF "+str(a+1))
    plt.legend(loc='lower right')
plt.title("Эмпирическая функция выборки объема: "+str(v))
x=np.linspace(0,5,100)#Построение
cdf=maxwell_rv.cdf(x)#теоретической функции
plt.plot(x,cdf,label='CDF')#распределения
plt.legend()
plt.xlabel("numbers")
plt.ylabel("probability")
plt.show()
```

In [47]:

```python
#n=10
for a in range(5):
    b = means__10[a]
    b = sorted(b)
    v = len(b)
    N = []
    for i in range((v-1)):
        N.append(b.count(b[i]))
        x=[]
        y=[]
        t=0
        x.append(0.0)
        y.append(0.0)
        x.append(b[0])
        y.append(0.0)
    for i in range((v-1)):
        t+=N[i]
        x.append(b[i])
        y.append(float(t/v))
        x.append(b[i+1])
        y.append(float(t/v))
    x.append(b[v-1])
    y.append(1)
    x.append(b[v-1]+2)
    y.append(1)
    plt.plot(x,y,label="ECDF "+str(a+1))
    plt.legend(loc='lower right')
plt.title("Эмпирическая функция выборки объема: "+str(v))
x=np.linspace(0,5,100)#Построение
cdf=maxwell_rv.cdf(x)#теоретической функции
plt.plot(x,cdf,label='CDF')#распределения
plt.legend()
plt.xlabel("numbers")
plt.ylabel("probability")
plt.show()
```

Эмпирическая функция выборки объема: 10

In [48]:

```python
#n=100
for a in range(5):
    b=means__100[a]
    b=sorted(b)
    v=len(b)
    N=[]
    for i in range((v-1)):
        N.append(b.count(b[i]))
        x=[]
        y=[]
        t=0
        x.append(0.0)
        y.append(0.0)
        x.append(b[0])
        y.append(0.0)
    for i in range((v-1)):
        t+=N[i]
        x.append(b[i])
        y.append(float(t/v))
        x.append(b[i+1])
        y.append(float(t/v))
    x.append(b[v-1])
    y.append(1)
    x.append(b[v-1]+2)
    y.append(1)
    plt.plot(x,y,label="ECDF "+str(a+1))
    plt.legend(loc='lower right')
plt.title("Эмпирическая функция выборки объема: "+str(v))
x=np.linspace(0,5,100)#Построение
cdf=maxwell_rv.cdf(x)#теоретической функции
plt.plot(x,cdf,label='CDF')#распределения
plt.legend()
plt.xlabel("numbers")
plt.ylabel("probability")
plt.show()
```

In [49]:

```
#n=1000
for a in range(5):
    b=means__1000[a]
    b=sorted(b)
    v=len(b)
    N=[]
    for i in range((v-1)):
        N.append(b.count(b[i]))
        x=[]
        y=[]
        t=0
        x.append(0.0)
        y.append(0.0)
        x.append(b[0])
        y.append(0.0)
    for i in range((v-1)):
        t+=N[i]
        x.append(b[i])
        y.append(float(t/v))
        x.append(b[i+1])
        y.append(float(t/v))
    x.append(b[v-1])
    y.append(1)
    x.append(b[v-1]+2)
    y.append(1)
    plt.plot(x,y,label="ECDF "+str(a+1))
    plt.legend(loc='lower right')
plt.title("Эмпирическая функция выборки объема: "+str(v))
x=np.linspace(0,5,100)#Построение
cdf=maxwell_rv.cdf(x)#теоритической функции
plt.plot(x,cdf,label='CDF')#распределения
plt.legend()
plt.xlabel("numbers")
plt.ylabel("probability")
plt.show()
```

2.2.3 Построение вариационного ряда выборки

In [50]:

```
#Вариационный ряд для выборки объема n=5 с выводом
for a in range(5):
    b=means__5[a]
    b=sorted(b)
    print(b)
```

```
[0.2682322717726196, 1.0252853262255397, 1.3891299766326262, 1.948086347877021, 2.859634488546546]
[0.7115665603506331, 0.7844876090587238, 1.1018597585263727, 1.6189547808430573, 1.942733061095644
6]
[0.14553190161724394, 0.364835798735535, 1.2266503057821223, 1.3492529406630769, 2.061438067233203
8]
[0.6185777082115862, 1.7953040901490582, 1.94926054186658, 2.000544726961293, 3.0660957861143627]
[0.9532155653922162, 1.148873684138907, 1.284242459402836, 1.3307689660880058, 1.8888378445724914]
```

In [51]:

```
#Вариационный ряд для выборки объема n=10 с выводом
for a in range(5):
    b=means__10[a]
    b=sorted(b)
    print(b)
```

```
[0.5749845509366696, 0.707475198314964, 1.0972151446298386, 1.284889228558776, 1.5919087144802937,
1.8658035113709832, 1.9496562133045796, 2.1514517067360814, 2.316723181584452, 3.081686151190713]
[0.7398773929288421, 0.7893058686573586, 0.8225084515809106, 0.8431446622181724, 0.876728926954281
4, 1.0560476697281345, 1.2107983181931672, 1.312064458895024, 1.6164658019594953, 2.059734665986926
2]
[0.3697577967794566, 0.5705320019262893, 0.7607822123827164, 0.8367177119013597, 1.704809969733966,
1.885330543165392, 2.0525201448757584, 2.130274611970833, 2.141766623829636, 2.5494555755265034]
[1.0476388694259509, 1.0933063582250218, 1.107024545963545, 1.3108813318343593, 1.3488298931738225,
1.6356879645991924, 2.115595459156639, 2.247970217761531, 2.3908119648693895, 3.1615379948971247]
[0.4915791152652938, 1.0153652731173193, 1.1030241213068572, 1.1219742128932553, 1.124856626460365
4, 1.1881533780081537, 1.3748759390540006, 1.4594817178204689, 1.751958230147981, 2.14866570526144]
```

In [52]:

```
#Вариационный ряд для выборки объема n=100 без вывода
for a in range(5):
    b=means__100[a]
    b=sorted(b)
```

In [53]:

```
#Вариационный ряд для выборки объема n=1000 без вывода
for a in range(5):
    b=means__1000[a]
    b=sorted(b)
```

In [54]:

```
#Вариационный ряд для выборки объема n=100000 без вывода
for a in range(5):
    b=means__100000[a]
    b=sorted(b)
```

Возникли сложности при вычислении теоретических значений квантилей, однако был найден справочник:"Справочник по вероятностным распределениям" Р.Н.Вадзинский. В нём была найдена таблица для приближенного решения уравнения $x_\alpha = \lambda m_\alpha$, где $x_\alpha = \lambda m_\alpha$ - квантиль порядка α распределения Максвелла

In [55]:

```
k=1
for b in [means__5[a],means__10[a],means__100[a],means__1000[a],means__100000[a]]:
    if(k==1):
        print('n = 5')
    if(k==2):
        print('n = 10')
    if(k==3):
        print('n = 100')
    if(k==4):
        print('n = 1000')
    if(k==5):
        print('n = 100000')
    print(np.quantile(b,0.1))
    k+=1
```

```
n = 5
1.0314788128908925
n = 10
0.9629866573321167
n = 100
0.6086318164716557
n = 1000
0.7809229897006393
n = 100000
0.7682332882428174
```

Сравнение со значением (с теоретическим) из таблицы:

$\alpha \approx 0.76$

In [56]:
```
k=1
for b in [means__5[a],means__10[a],means__100[a],means__1000[a],means__100000[a]]:
    if(k==1):
        print('n = 5')
    if(k==2):
        print('n = 10')
    if(k==3):
        print('n = 100')
    if(k==4):
        print('n = 1000')
    if(k==5):
        print('n = 100000')
    print(np.quantile(b,0.5))
    k+=1
```

```
n = 5
1.284242459402836
n = 10
1.1565050022342596
n = 100
1.5362455266176298
n = 1000
1.5125225518605903
n = 100000
1.5392797124106719
```

In [57]:
```
#Сравнение
maxwell.median()
```

Out[57]:
```
1.5381722544550522
```

In [58]:

```
k=1
for b in [means__5[a],means__10[a],means__100[a],means__1000[a],means__100000[a]]:
    if(k==1):
        print('n = 5')
    if(k==2):
        print('n = 10')
    if(k==3):
        print('n = 100')
    if(k==4):
        print('n = 1000')
    if(k==5):
        print('n = 100000')
    print(np.quantile(b,0.7))
    k+=1
```

```
n = 5
1.3214636647509717
n = 10
1.400257672683941
n = 100
1.7808030379040272
n = 1000
1.8878305035673368
n = 100000
1.917753324109793
```

Сравнение со значением (с теоретическим) из таблицы:
$\alpha \approx 1.92$

С увеличением объема выборки э.ф.р. стремится к теоритической функции распределения, следовательно, эмпирические квантили так же стремятся к теоритическим по определению. Что и можно наблюдать выше.

2.2.4 Построение гистограммы и полигона частот

In [59]:

```
#n=5
for a in range(5):
    b=means__5[a]
    mas=list(range(1,6))
    p=[0,0,0,0,0]
    for i in range(5):
        mas[i]=b[i]
        if mas[i]>0 and mas[i]<1:
            p[0]=p[0]+1
        if mas[i]>1 and mas[i]<2:
            p[1]=p[1]+1
        if mas[i]>2 and mas[i]<3:
            p[2]=p[2]+1
        if mas[i]>3 and mas[i]<4:
            p[3]=p[3]+1
        if mas[i]>4 and mas[i]<5:
            p[4]=p[4]+1
    print()
    dob=[]
    bod=[]
    keks=0.5
    for i in range(5):
        dob.append(keks)
        bod.append(p[i]/5.0)
        keks+=1
    plt.plot(dob,bod,label='EPMF'+str(a+1))
rv=maxwell()
x=np.linspace(0,5,100)
plt.plot(x,rv.pdf(x),'k-',lw=2,label='PMF')
plt.legend()
plt.title("n = 5")
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
```

In [60]:

```
#n=10
for a in range(5):
    b=means__10[a]
    mas=list(range(1,11))
    p=[0,0,0,0,0]
    for i in range(10):
        mas[i]=b[i]
        if mas[i]>0 and mas[i]<1:
            p[0]=p[0]+1
        if mas[i]>1 and mas[i]<2:
            p[1]=p[1]+1
        if mas[i]>2 and mas[i]<3:
            p[2]=p[2]+1
        if mas[i]>3 and mas[i]<4:
            p[3]=p[3]+1
        if mas[i]>4 and mas[i]<5:
            p[4]=p[4]+1
    print()
    dob=[]
    bod=[]
    keks=0.5
    for i in range(5):
        dob.append(keks)
        bod.append(p[i]/10.0)
        keks+=1
    plt.plot(dob,bod,label='EPMF'+str(a+1))
rv=maxwell()
x=np.linspace(0,5,100)
plt.plot(x,rv.pdf(x),'k-',lw=2,label='PMF')
plt.legend()
plt.title("n = 10")
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
```

In [61]:

```
#n=100
for a in range(5):
    b=means__100[a]
    mas=list(range(1,101))
    p=[0,0,0,0,0]
    for i in range(100):
        mas[i]=b[i]
        if mas[i]>0 and mas[i]<1:
            p[0]=p[0]+1
        if mas[i]>1 and mas[i]<2:
            p[1]=p[1]+1
        if mas[i]>2 and mas[i]<3:
            p[2]=p[2]+1
        if mas[i]>3 and mas[i]<4:
            p[3]=p[3]+1
        if mas[i]>4 and mas[i]<5:
            p[4]=p[4]+1
    print()
    dob=[]
    bod=[]
    keks=0.5
    for i in range(5):
        dob.append(keks)
        bod.append(p[i]/100.0)
        keks+=1
    plt.plot(dob,bod,label='EPMF'+str(a+1))
rv=maxwell()
x=np.linspace(0,5,100)
plt.plot(x,rv.pdf(x),'k-',lw=2,label='PMF')
plt.legend()
plt.title("n = 100")
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
```

In [62]:

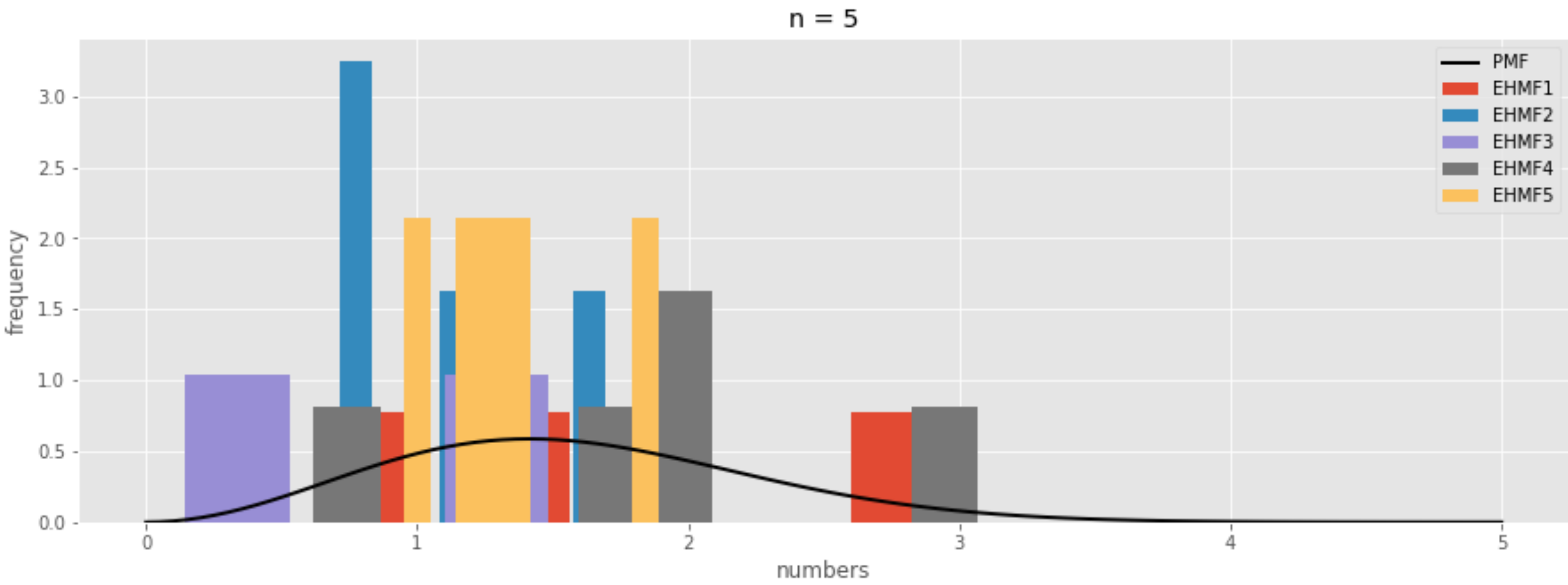
```
#n=1000
for a in range(5):
    b=means__1000[a]
    mas=list(range(1,1001))
    p=[0,0,0,0,0]
    for i in range(1000):
        mas[i]=b[i]
        if mas[i]>0 and mas[i]<1:
            p[0]=p[0]+1
        if mas[i]>1 and mas[i]<2:
            p[1]=p[1]+1
        if mas[i]>2 and mas[i]<3:
            p[2]=p[2]+1
        if mas[i]>3 and mas[i]<4:
            p[3]=p[3]+1
        if mas[i]>4 and mas[i]<5:
            p[4]=p[4]+1
    print()
    dob=[]
    bod=[]
    keks=0.5
    for i in range(5):
        dob.append(keks)
        bod.append(p[i]/1000.0)
        keks+=1
    plt.plot(dob,bod,label='EPMF'+str(a+1))
rv=maxwell()
x=np.linspace(0,5,100)
plt.plot(x,rv.pdf(x),'k-',lw=2,label='PMF')
plt.legend()
plt.title("n = 1000")
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
```

In [63]:

```
#n=100000
for a in range(5):
    b=means__100000[a]
    mas=list(range(1,100001))
    p=[0,0,0,0,0]
    for i in range(100000):
        mas[i]=b[i]
        if mas[i]>0 and mas[i]<1:
            p[0]=p[0]+1
        if mas[i]>1 and mas[i]<2:
            p[1]=p[1]+1
        if mas[i]>2 and mas[i]<3:
            p[2]=p[2]+1
        if mas[i]>3 and mas[i]<4:
            p[3]=p[3]+1
        if mas[i]>4 and mas[i]<5:
            p[4]=p[4]+1
    print()
    dob=[]
    bod=[]
    keks=0.5
    for i in range(5):
        dob.append(keks)
        bod.append(p[i]/100000.0)
        keks+=1
    plt.plot(dob,bod,label='EPMF'+str(a+1))
rv=maxwell()
x=np.linspace(0,5,100)
plt.plot(x,rv.pdf(x),'k-',lw=2,label='PMF')
plt.legend()
plt.title("n = 100000")
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
```

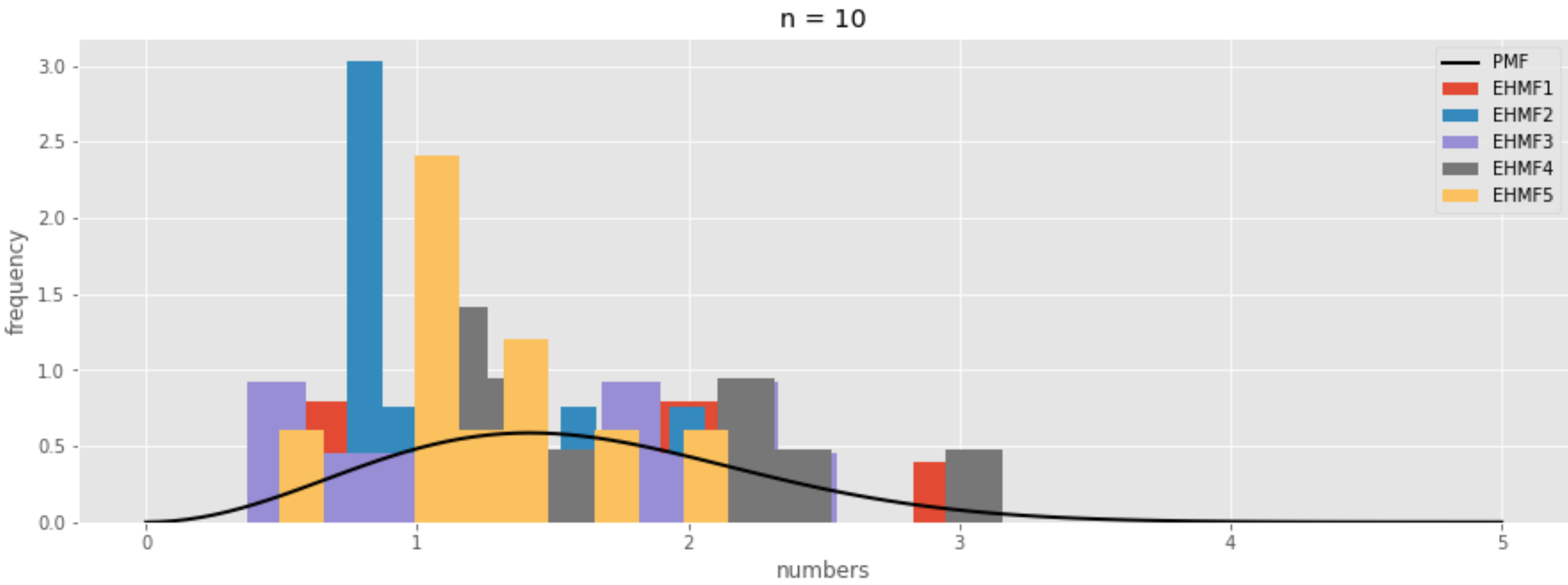
In [64]:

```
#n=5
for a in range(5):
    plt.hist(means__5[a],density=True,label='EHMF{}'.format(a+1))
    plt.legend()
rv=maxwell()
x=np.linspace(0,5,100)
plt.plot(x,rv.pdf(x),'k-',lw=2,label='PMF')
plt.legend()
plt.title("n = 5")
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
```



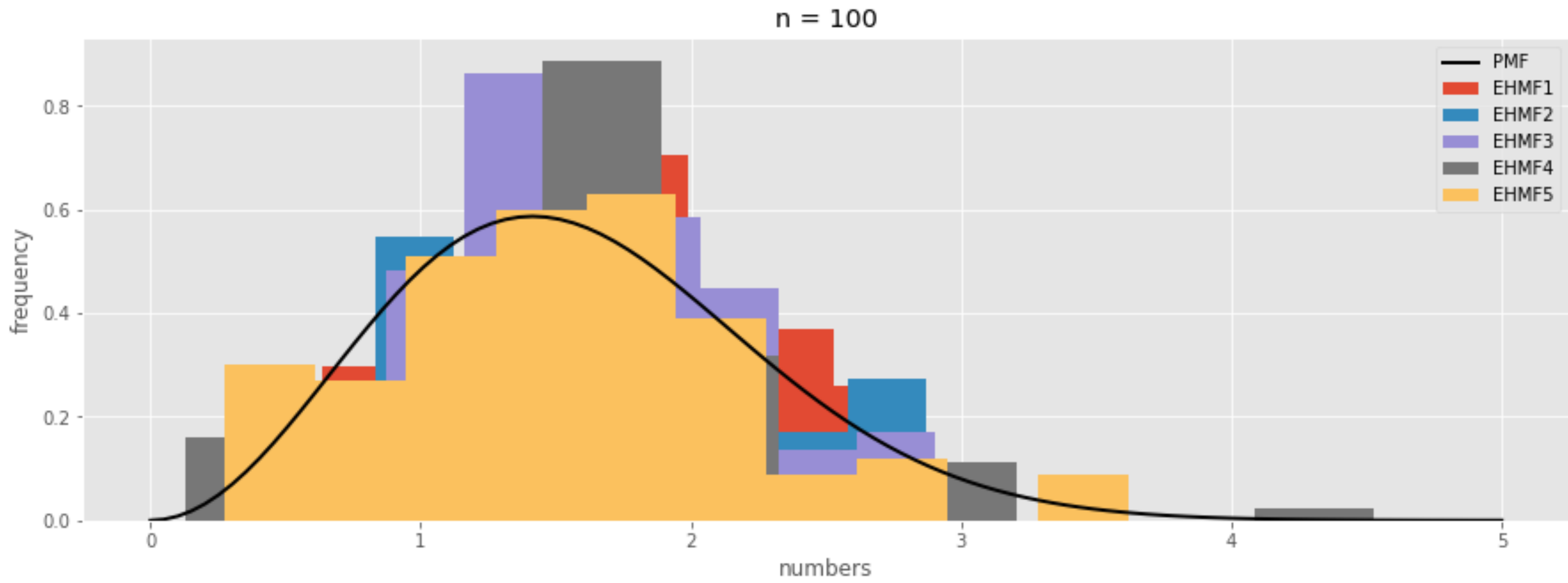
In [65]:

```
#n=10
for a in range(5):
    plt.hist(means__10[a],density=True,label='EHMF{}'.format(a+1))
    plt.legend()
rv=maxwell()
x=np.linspace(0,5,100)
plt.plot(x,rv.pdf(x),'k-',lw=2,label='PMF')
plt.legend()
plt.title("n = 10")
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
```



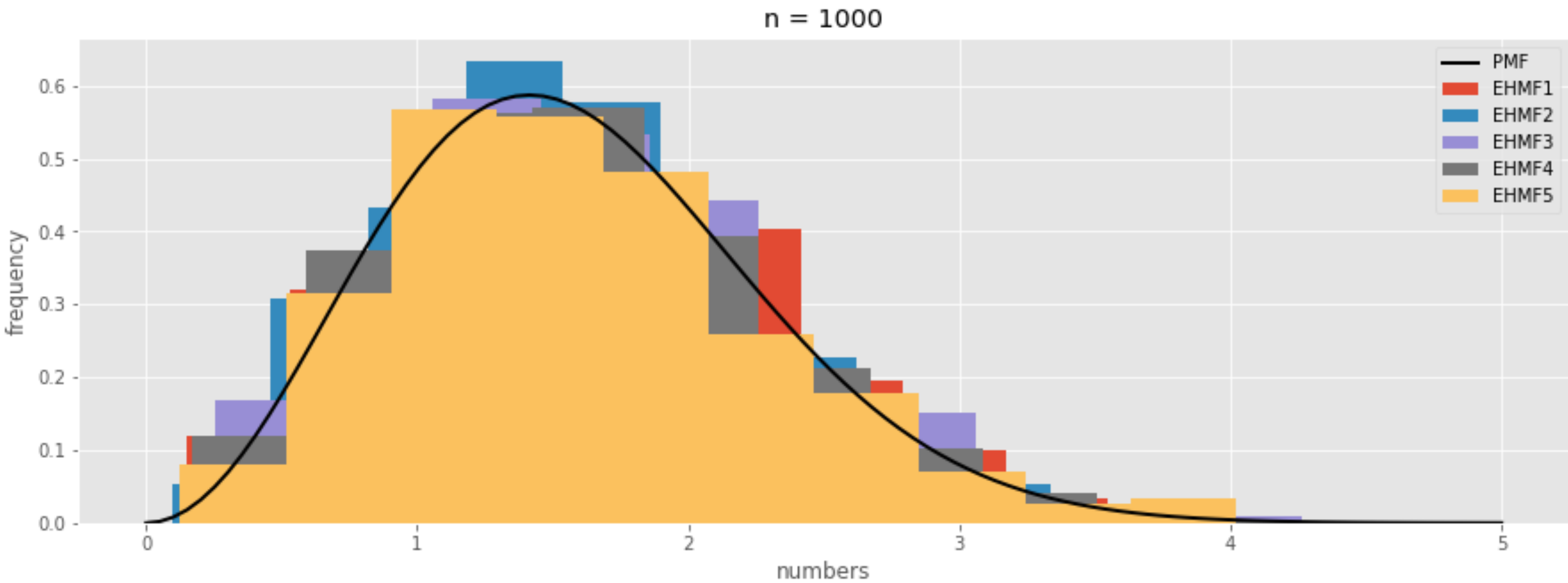
In [66]:

```
#n=100
for a in range(5):
    plt.hist(means__100[a],density=True,label='EHMF{}'.format(a+1))
    plt.legend()
rv=maxwell()
x=np.linspace(0,5,100)
plt.plot(x,rv.pdf(x),'k-',lw=2,label='PMF')
plt.legend()
plt.title("n = 100")
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
```



In [67]:

```python
#n=1000
for a in range(5):
    plt.hist(means__1000[a],density=True,label='EHMF{}'.format(a+1))
    plt.legend()
rv=maxwell()
x=np.linspace(0,5,100)
plt.plot(x,rv.pdf(x),'k-',lw=2,label='PMF')
plt.legend()
plt.title("n = 1000")
plt.xlabel("numbers")
plt.ylabel("frequency")
plt.show()
```



Для непрерывной случайной величины ξ, обладающей непрерывной плотностью f(x), также можно построить по соответствующей выборке $X = (X_1, \ldots, X_n)$ статистический аналог $\hat{f}_n(x)$ для плотности f(x), который называется гистограммой. Для этого используется методгруппировки, в соответствии с которым область Δ возможных значений ξ разбивается на некоторое число N непересекающихся интервалов $\Delta_1, \ldots, \Delta_N$ (так что $\Delta = \bigcup_{r=1}^{N} \Delta_r$, подсчитывают числа $\nu_1, \ldots, \nu_N$ наблюдений $X_1, \ldots, X_n$, попавших в соответствующие интервалы: $\nu_r = \sum_{j=1}^{N} I(X_j \in \Delta_r), r = 1, \ldots, N$ (так что $\sum_{r=1}^{N} \nu_r = n,$
и строят кусочно-постоянную функцию

$$\hat{f}_n(x) = \frac{\nu_r}{n|\Delta_r|}$$

при x $\in \Delta_r, r = 1, \ldots, N$
Здесь $|\Delta_r|$ - длина интервала Δ_r. То, что построенная по такому правилу гистограмма $\hat{f}_n(x)$ действительно "похожа" на теоретическую плотность f(x), следует из закона больших чисел, согласно которому при $n \to \infty$ относительная частота $\frac{\nu_r}{n}$ сближается с теоретической вероятностью

$$P\{\xi \in \Delta_r\} = \int_{\Delta_r} f(x)dx$$

Но этот интеграл по теореме о среднем равен $f(a_r)|\Delta_r|$ где a_r - некоторая внутренняя точка интервала Δ_r(при малом Δ_r в качестве a_r можно взять, например, середину интервала), Таким образом, при больших n и достаточно "мелком" разбиении $\{\Delta_r\}\hat{f}_n(x) \approx f(a_r)$ при $x \in \Delta_r$ т.е. гистограмма $\hat{f}_n(x)$ будет достаточно хорошо приближать график плотности f(x), следовательно, $\hat{f}_n(x)$ можно рассматривать в качестве статистического аналога (оценки) для f(x).Наряду с гистограммой, в качестве приближения для неизвестной теоретической плотности f(x) можно использовать кусочно-линейный график называемый полигоном частот. Он также считается статистическим аналогом теоретической плотности. Данные на полигоне частот и гистограммах подтверждают теоретические знания: с увеличением объема выборки полигон частот и гистограммы практически совпадают с теоретической плотностью f(x).