Discrete distribution

In [8]:

```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
from IPython.display import Math, Latex
from IPython.core.display import Image
```

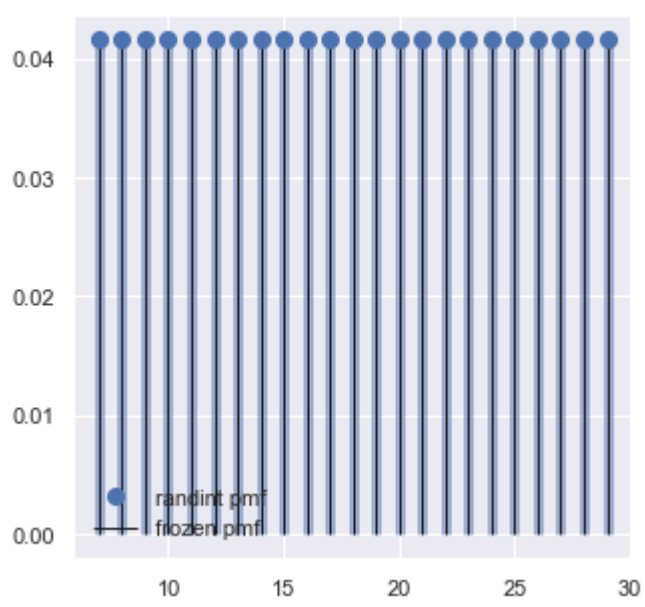
In [9]:

```
import seaborn as sns
sns.set(color_codes=True)#set plotting style
sns.set(rc={'figure.figsize':(5,5)})#set plot sizes
```

Uniform discrete distribution

In [12]:

```
from scipy.stats import randint#for x value we need random data(random integer)
import matplotlib.pyplot as plt
fig,ax= plt.subplots(1,1)

#calculate a few 1st moments:
low,high =7,31 #low high a,b
mean,var,skew,kurt=randint.stats(low,high,moments='mvsk')#mvsk=mean,var,skew,kur

#display p.m.f
x=np.arange(randint.ppf(0.01,low,high),
          randint.ppf(0.99,low,high)) #ppf=percent point function convert data 7,31 wala in


ax.plot(x,randint.pmf(x,low,high),'bo',ms=8,label='randint pmf')
ax.vlines(x,0,randint.pmf(x,low,high),colors='b',lw=5,alpha=0.5)
# Alternatively, the distribution object can be called (as a function)
# to fix the shape and location . This returns a "frozon"RV object holding
# The given parameters fixed.


# Frozen the distribution and display the frozen 'pmf':#Check accuracy of 'cdf' and 'ppf':
rv=randint(low,high)
ax.vlines(x,0,rv.pmf(x),colors='k',linestyles='-',lw=1,label='frozen pmf')
ax.legend(loc='best',frameon=False)
plt.show()

#Check accuracy of 'cdf' and 'ppf':
prob=randint.cdf(x,low,high)
np.allclose(x,randint.ppf(prob,low,high))
#True


#Generate random numbers:
r=randint.rvs(low,high,size=1000)



#mvsk=mean,var,skew,kurt
#ppf=percent point function convert data 7,31 wala into 0 to 1 ki range mai cdf ka inverse
#low high a,b
#alpha-color ferquency
#frameon=false beacase i am plotting all in one graph
```



In [13]:

```
prob=randint.cdf(x,low,high)
np.allclose(x,randint.ppf(prob,low,high))
```

Out[13]:

```
True
```

In [ ]:

```

```