Подключаем необходимые библиотеки. Seaborn -- аналог matplotlib, но строит более красивые гистограммы

In [ ]:
```
import numpy as np
import scipy.stats
import seaborn as sns
```

Далее вводим наши данные

In [ ]:
```
alpha = 0.01
```

In [ ]:
```
n_i = np.array([12, 25, 77, 176, 192, 178, 83, 24, 13])
data = np.concatenate((np.array([1] * 12),
                       np.array([3] * 25),
                       np.array([5] * 77),
                       np.array([7] * 176),
                       np.array([9] * 192),
                       np.array([11] * 178),
                       np.array([13] * 83),
                       np.array([15] * 24),
                       np.array([17] * 13)))
n = len(data)
```

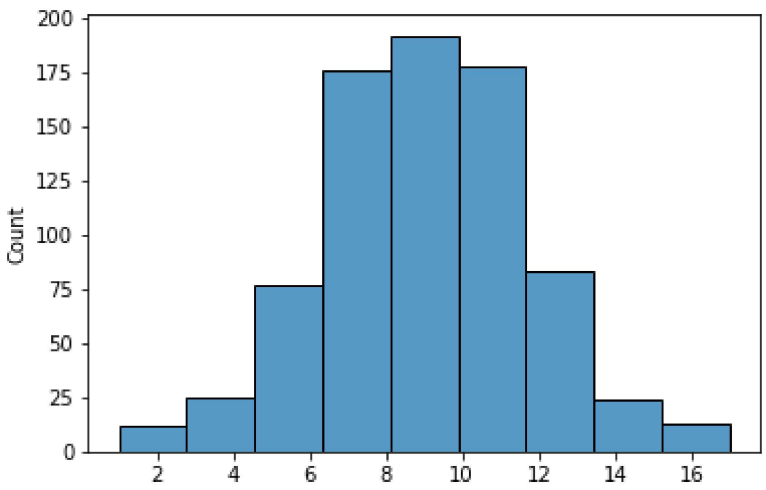
In [ ]:
```
print('Диапазон | Число наблюдений')
for i in range(9):
    print('{:>3} - {:>2}'.format(i * 2, (i + 1) * 2), '|', n_i[i])
```

```
Диапазон | Число наблюдений
  0 -  2 | 12
  2 -  4 | 25
  4 -  6 | 77
  6 -  8 | 176
  8 - 10 | 192
 10 - 12 | 178
 12 - 14 | 83
 14 - 16 | 24
 16 - 18 | 13
```

Строим гистограмму

In [ ]:
```
sns.histplot(data, bins=9)
```

Out[ ]:
```
<AxesSubplot:ylabel='Count'>
```



Находим точечные оценки математического ожидания (avg) и дисперсии (std2). Коэффициент $\frac{n}{n-1}$ введён, чтобы исправить неточность формулы, используемой библиотекой numpy**

In [ ]:
```
avg = np.mean(data)
std = np.std(data) * (n / (n - 1))
avg, std**2
```

Out[ ]:
```
(9.038461538461538, 9.432948663653248)
```

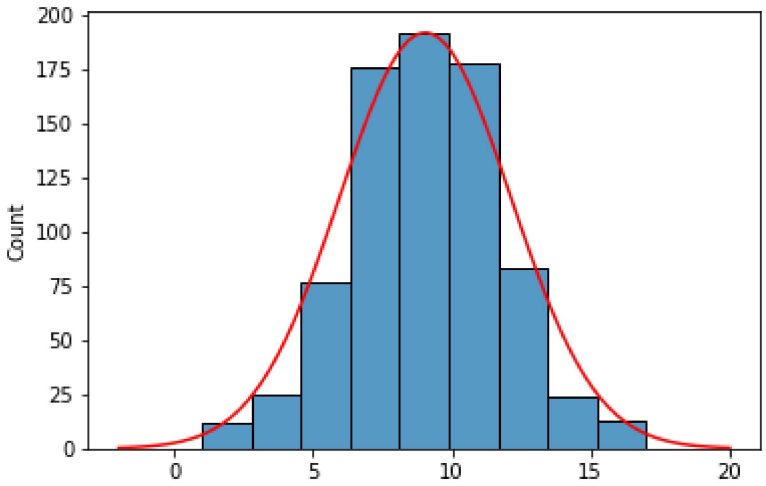
Строим теоретическую кривую нормального распределения с заданными значениями математического ожидания и дисперсии

In [ ]:
```
X = np.linspace(-2, 20, 100)
norm_dist = scipy.stats.norm.pdf(X, avg, std)

sns.histplot(data, bins=9)
sns.lineplot(X, norm_dist * 192 * (2 * np.pi) ** 0.5 * std, color='red')
```

```
c:\Users\Delta_C\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarn
ing: Pass the following variables as keyword args: x, y. From version 0.12, the on
ly valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[ ]:
```
<AxesSubplot:ylabel='Count'>
```



Вычисляем теоретические вероятности. Для проверки выводим сумму вероятностей

In [ ]:
```
th_prob = []
for i in range(0, 17, 2):
    if i == 0:
        th_prob.append(scipy.stats.norm.cdf(2, avg, std))
    elif i == 16:
        th_prob.append(1 - scipy.stats.norm.cdf(16, avg, std))
    else:
        th_prob.append(scipy.stats.norm.cdf(i + 2, avg, std) - scipy.stats.norm.cdf

th_prob = np.array(th_prob)
np.sum(th_prob)
```

Out[ ]:
```
1.0
```

Вычисляем теоретические частоты и значение χ^2

In [ ]:
```
th_freq = th_prob * n
chi_squared = np.sum((n_i - th_freq) ** 2 / th_freq)

chi_squared
```

Out[ ]:
```
10.647135282454835
```

Находим значение 1% квантиля распределения χ^2 с 6-ю степенями свободы и сравниваем полученные числа

In [ ]:
```
th_chi_squared = scipy.stats.chi2.ppf(1 - alpha, 6)
th_chi_squared
```

Out[ ]:
```
16.811893829770927
```

In [ ]:
```
chi_squared < th_chi_squared
```

Out[ ]:
```
True
```

Как видим, наблюдаемое значение меньше критического, а значит, нет оснований отвергать гипотезу