In [1]:

```
import numpy as np
import scipy.stats as sps
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.distributions.empirical_distribution import ECDF
from statsmodels.sandbox.stats.multicomp import multipletests
from collections import Counter
from tqdm import tqdm_notebook

red = '#FF3300'
blue = '#0099CC'
green = '#00CC66'

%matplotlib inline
sns.set(style='ticks', font_scale=1.7)
```

Критерии согласия

Критерий согласия Пирсона (хи-квадрат)

H_0: Выборка из некоторого класса распределений

H_1: H_0 не верна.

`chisquare` (https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.chisquare.html#scipy.stats.chisquare)
`(f_obs, f_exp=None, ddof=0)`

- `f_obs` --- число элементов выборки, попавших в каждый из интервалов
- `f_exp` --- ожидаемое число (по умолчанию равномерное)
- `ddof` --- поправка на число степеней свободы. Статистика асимптотически будет иметь распределение $k - 1 - ddof$, где k --- число интервалов.

In [2]:

```
sps.chisquare([16, 18, 16, 14, 12, 12])
```

Out[2]:

```
Power_divergenceResult(statistic=2.0, pvalue=0.8491450360846096)
```

In [3]:

```
sps.chisquare([16, 18, 16, 14, 12, 12], f_exp=[16, 16, 16, 16, 20, 4])
```

Out[3]:

```
Power_divergenceResult(statistic=19.7, pvalue=0.0014224993317060594)
```

In [4]:

```
1 sps.chisquare([16, 18, 16, 14, 12, 12], f_exp=[16, 16, 16, 16, 24, 0])
```

```
/usr/local/lib/python3.7/dist-packages/scipy/stats/stats.py:5048: Runt
imeWarning: divide by zero encountered in true_divide
  terms = (f_obs - f_exp)**2 / f_exp
```

Out[4]:

```
Power_divergenceResult(statistic=inf, pvalue=0.0)
```

Эксперимент с группами крови

In [5]:

```
1 sps.chisquare([121, 120, 79, 33], f_exp=np.array([0.343, 0.340, 0.224, 0.093])
```

Out[5]:

```
Power_divergenceResult(statistic=0.001011144526478114, pvalue=0.974632
7438096731)
```

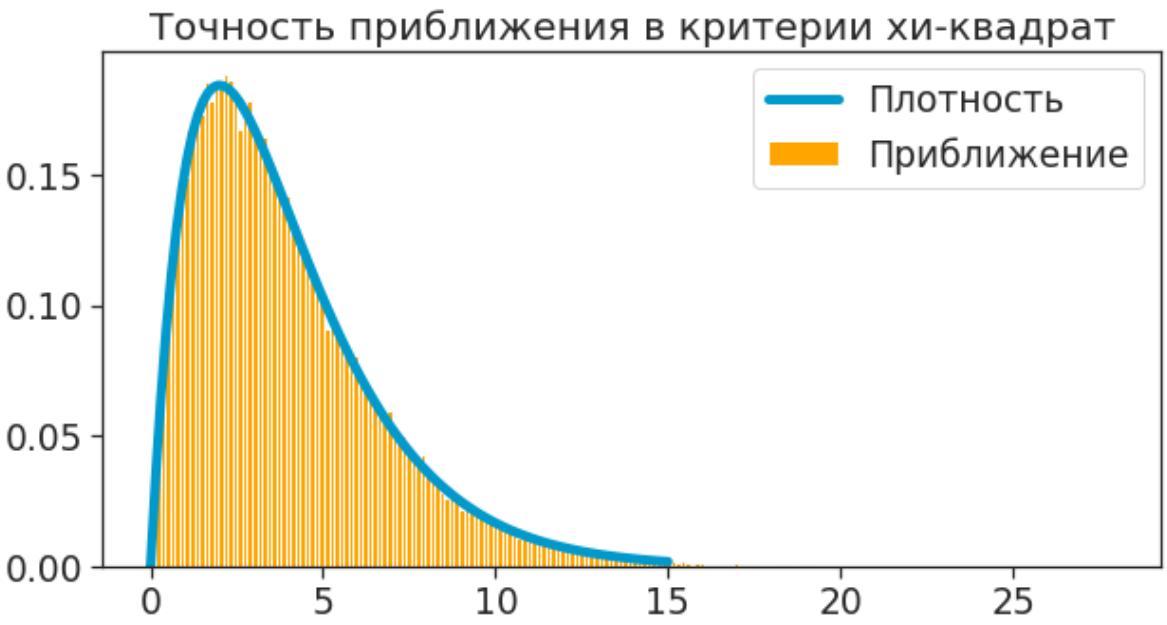
Посмотрим, насколько хорошо выполняется асимптотика при справедливости нулевой гипотезы для разных распределений.

1. Равномерное на 5 элементах

In [6]:

```
sample_size = 1000
sample_count = 100000
chisq_values = np.zeros(sample_count)

for i in tqdm_notebook(range(sample_count)):
    sample = sps.randint(low=0, high=5).rvs(size=sample_size)
    f_obs = np.array(list(Counter(sample).values()))
    chisq_values[i] = sps.chisquare(f_obs)[0]

plt.figure(figsize=(10, 5))
grid = np.linspace(0, 15, 100)
plt.plot(grid, sps.chi2(df=4).pdf(grid),
         lw=5, color=blue, label='Плотность')
plt.hist(chisq_values, bins=200,
         color='orange', density=True, label='Приближение')
plt.title('Точность приближения в критерии хи-квадрат')
plt.legend();
```

100% 100000/100000 [11:21<00:00, 146.68it/s]

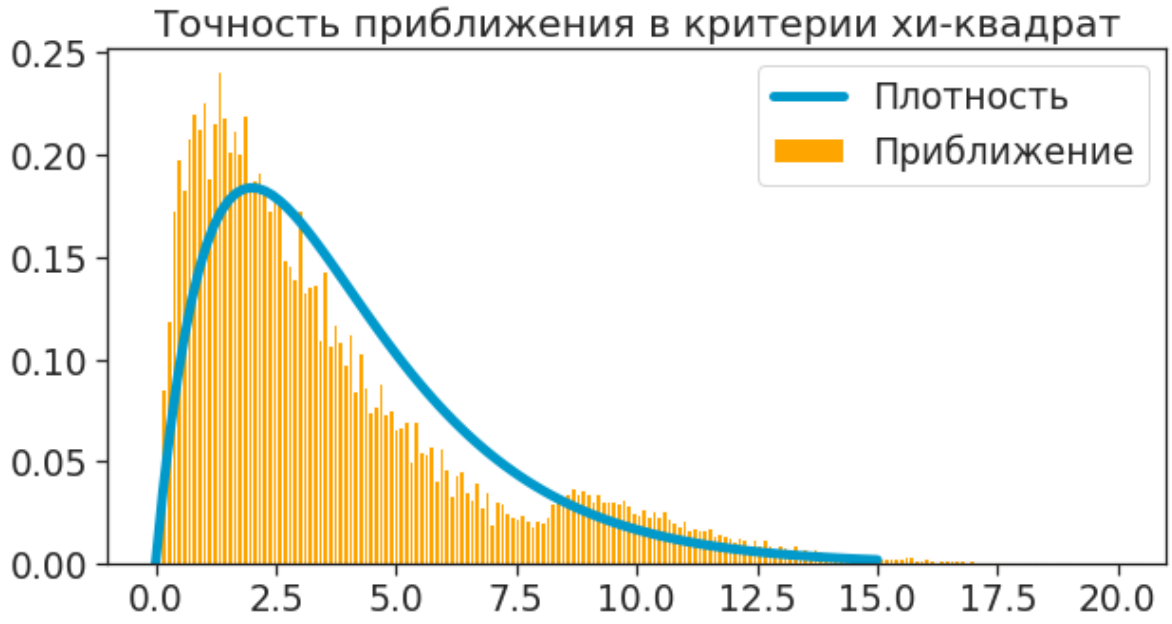


2. Равномерное на 4 элементах, а 5-й с очень маленькой вероятностью.

In [7]:

```
chisq_values = np.zeros(sample_count)

epsilon = 0.0001
pk = np.array([0.25 - epsilon / 4] * 4 + [epsilon])
some_distribution = sps.rv_discrete(name='1', values=(np.arange(5), pk))

for i in tqdm_notebook(range(sample_count)):
    sample = some_distribution.rvs(size=sample_size)
    f_obs = np.array(list(Counter(sample).values()))
    if len(f_obs) == 4: f_obs = np.append(f_obs,[0])
    f_exp = pk * sample_size
    chisq_values[i] = sps.chisquare(f_obs, f_exp=f_exp)[0]

plt.figure(figsize=(10, 5))
grid = np.linspace(0, 15, 100)
plt.plot(grid, sps.chi2(df=4).pdf(grid),
         lw=5, color=blue, label='Плотность')
plt.hist(chisq_values, bins=200,
         color='orange', density=True, label='Приближение', range=(0, 21))
plt.title('Точность приближения в критерии хи-квадрат')
plt.xlim((-1, 21));
plt.legend();
```

100% 100000/100000 [00:44<00:00, 2248.06it/s]



Вывод: асимптотика ломается, если есть интервалы с малой вероятностью или с малым числом элементов в выборке.

Критерий согласия Колмогорова (-Смирнова) {для нормальных Лиллиефорса}

H_0: Выборка из некоторого класса распределений

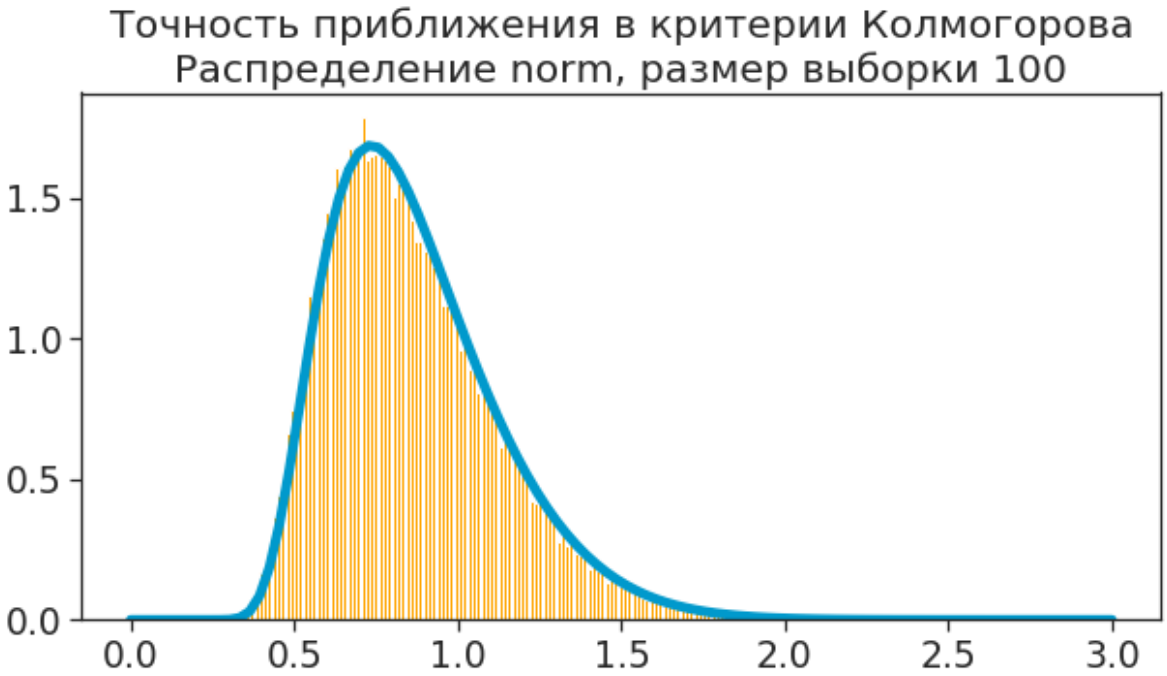
H_1: H_0 не верна.

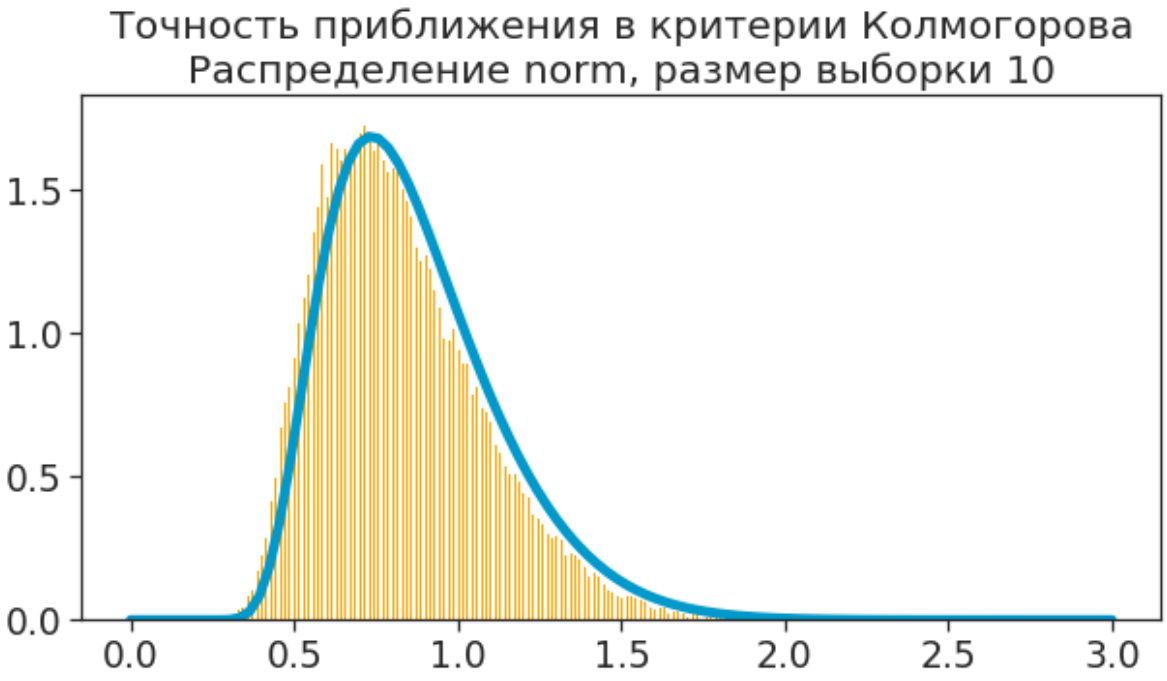
`kstest` (https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.kstest.html#scipy.stats.kstest)
`(rvs, cdf, args=())`

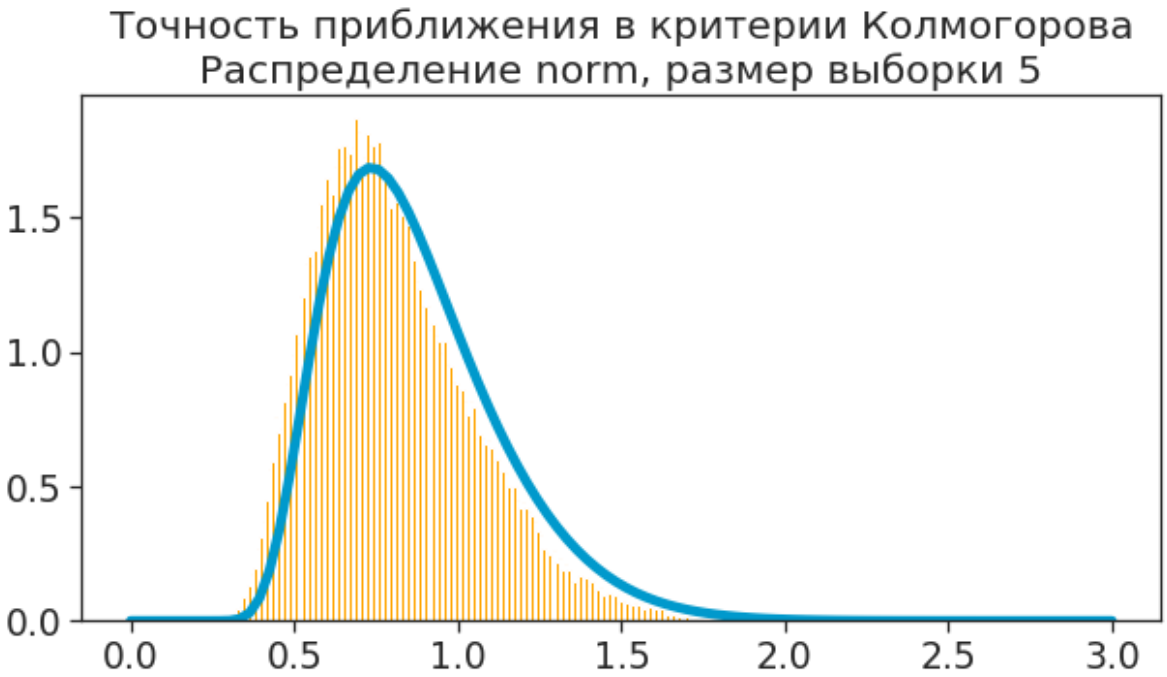
- `rvs` --- выборка
- `cdf` --- функция распределения (сама функция или ее название)
- `args` --- параметры распределения

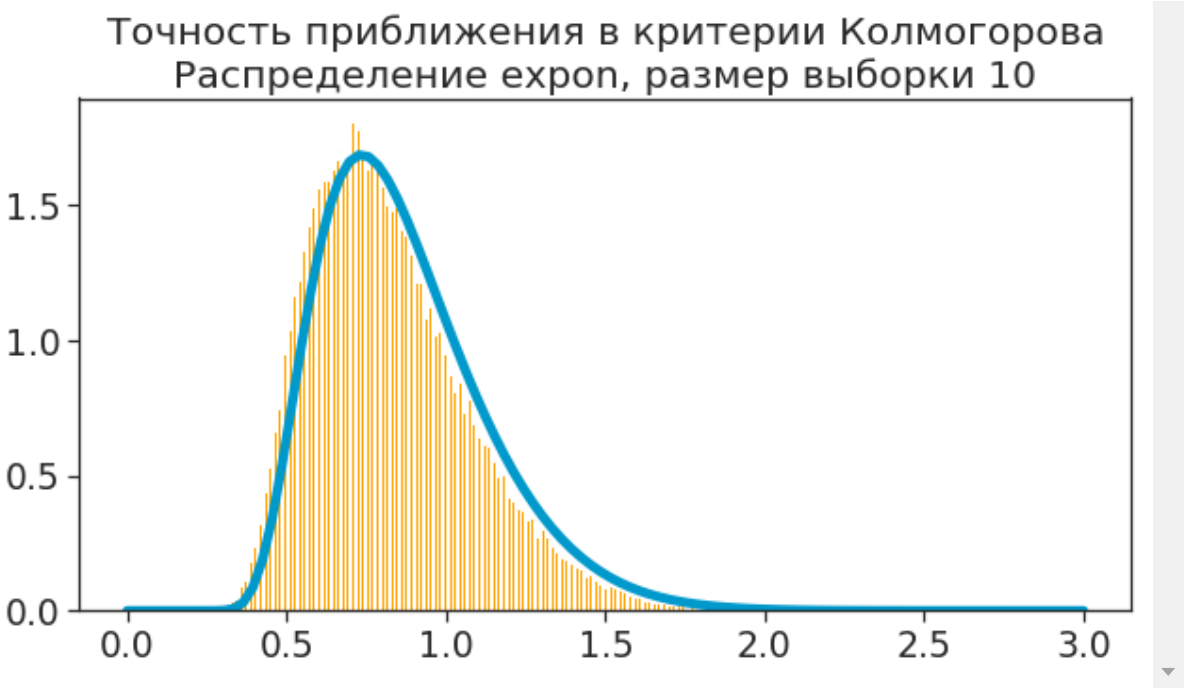
In [8]:

```
grid = np.linspace(0, 3, 100)
n_iter = 100000

for size, distr in zip([100, 10, 5, 10], [sps.norm] * 3 + [sps.expon]):
    D = []

    samples = distr.rvs(size=(n_iter, size))
    for i in tqdm_notebook(range(n_iter), leave=False):
        D.append(sps.kstest(samples[i], cdf=distr.cdf)[0])

    plt.figure(figsize=(10, 5))
    plt.plot(grid, sps.kstwobign.pdf(grid),
             lw=5, label='Kolmogorov pdf', color=blue)
    plt.hist(np.array(D) * np.sqrt(size), bins=200, density=True,
             label='hist $\\sqrt{n} D_n$', color='orange')
    plt.title('Точность приближения в критерии Колмогорова\n' \
              + 'Распределение {}, размер выборки {}'.format(distr.name, size))
    plt.show()
```




Точность приближения в критерии Колмогорова
Распределение norm, размер выборки 10
1.5
1.0
0.5
0.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0


Точность приближения в критерии Колмогорова
Распределение norm, размер выборки 5
1.5
1.0
0.5
0.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0



Рассмотрим случай $H_0: \mathcal{N}(0, 1)$

In [9]:

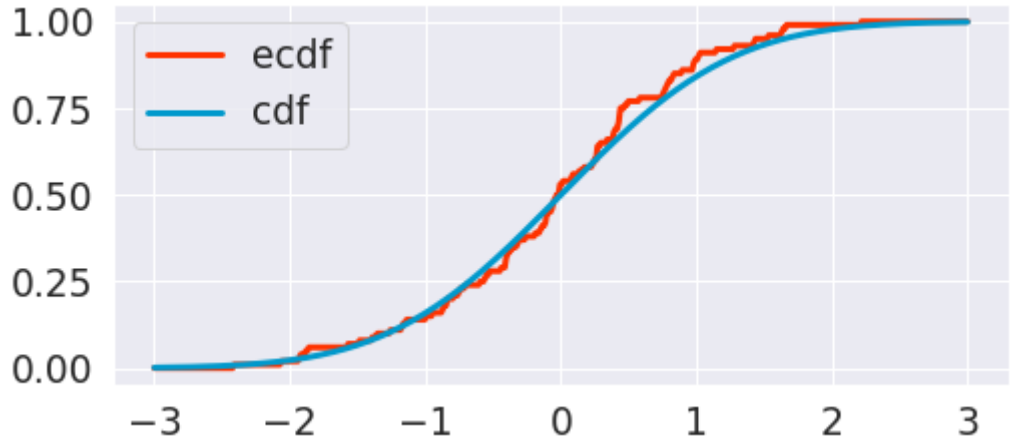
```
sns.set(style='darkgrid', font_scale=1.7)

def apply_kstest(sample, cdf=sps.norm.cdf):
    print(sps.kstest(sample, sps.norm.cdf))

    ecdf = ECDF(sample)
    grid = np.linspace(-3, 3, 500)

    plt.figure(figsize=(8, 3.5))
    plt.plot(grid, ecdf(grid), color=red, label='ecdf', lw=3)
    plt.plot(grid, cdf(grid), color=blue, label='cdf', lw=3)
    plt.legend()
    plt.show()
```

Нулевая гипотеза верна:

In [10]:

```
1 apply_kstest(sps.norm.rvs(size=100))
```

```
KstestResult(statistic=0.08237600150211866, pvalue=0.4877223292527129)
```

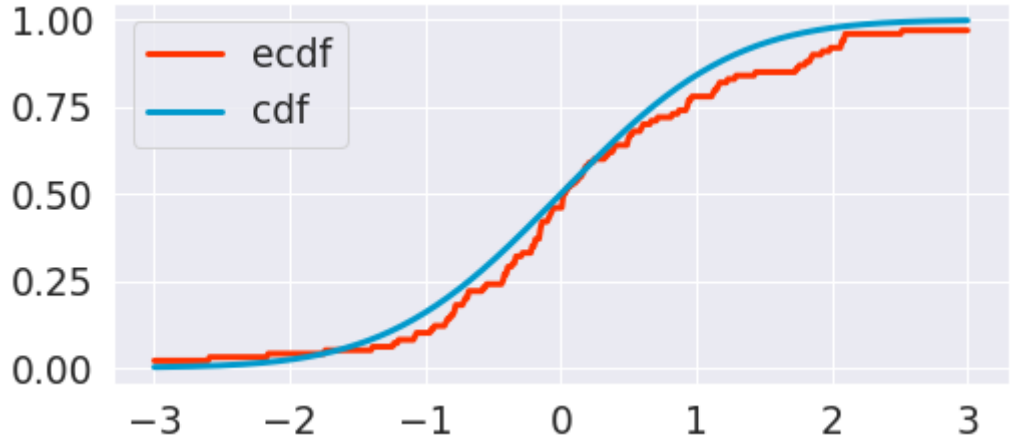


Истинное распределение - Лапласс

In [11]:

```
1 apply_kstest(sps.laplace.rvs(size=100))
```

```
KstestResult(statistic=0.10815556124535541, pvalue=0.1791469170063987
6)
```

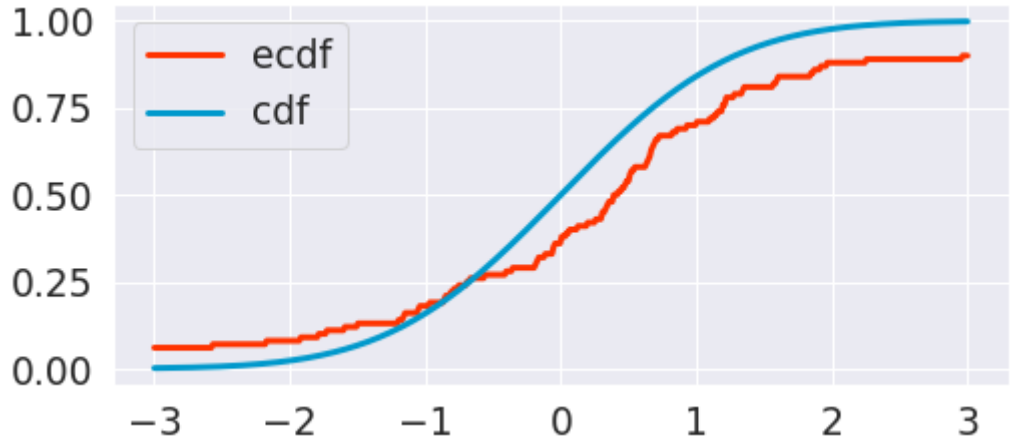


Истинное распределение - Коши

In [12]:

```
1 apply_kstest(sps.cauchy.rvs(size=100))
```

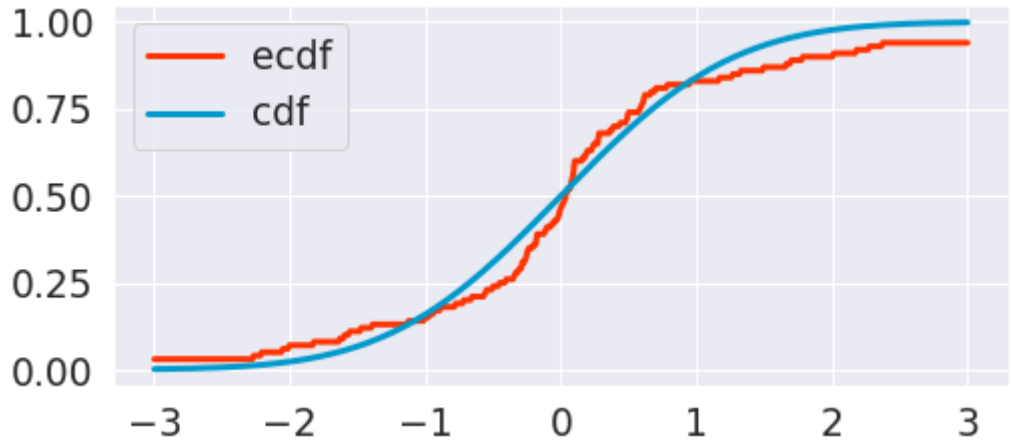
```
KstestResult(statistic=0.18756670386283075, pvalue=0.00148921222793695
9)
```



In [13]:

```
1 apply_kstest(sps.cauchy(scale=0.5).rvs(size=100))
```

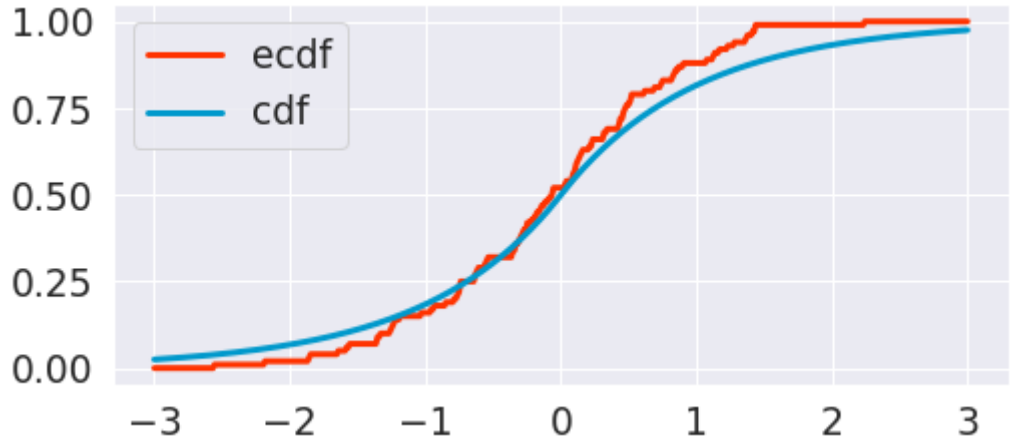
```
KstestResult(statistic=0.10255129072305458, pvalue=0.2278742984580734)
```



In [14]:

```
1 apply_kstest(sps.norm.rvs(size=100), cdf=sps.laplace.cdf)
```

```
KstestResult(statistic=0.09056612392589924, pvalue=0.3652768891115752)
```



Вывод: Критерий не чувствителен к хвостам распределений.

Критерий Андерсона-Дарлинга

H_0: Выборка из некоторого класса распределений

H_1: H_0 не верна.

`anderson` (https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.anderson.html#scipy.stats.anderson) `(x, dist='norm')`

- `x` --- выборка
- `dist : {'norm', 'expon', 'logistic', 'gumbel'}` --- распределение

Возвращает `(statistic, critical_values, significance_level)`. В `critical_values` записаны пороги для критериев, которым соответствуют уровни значимости из `significance_level` в процентах. То есть критерий `{statistic > critical_values[i]}` имеет уровень значимости `significance_level[i]`.

In [15]:

```
1 sample = sps.norm.rvs(size=100)
2 sps.anderson(sample)
```

Out[15]:

```
AndersonResult(statistic=0.3551151814259441, critical_values=array([0.
555, 0.632, 0.759, 0.885, 1.053]), significance_level=array([15. , 10.
,  5. ,  2.5,  1. ]))
```

In [16]:

```
sample = sps.laplace.rvs(size=100)
sps.anderson(sample)
```

Out[16]:

```
AndersonResult(statistic=2.1492631844960783, critical_values=array([0.
555, 0.632, 0.759, 0.885, 1.053]), significance_level=array([15. , 10.
,  5. ,  2.5,  1. ]))
```

In [17]:

```
sample = sps.cauchy.rvs(size=100)
sps.anderson(sample)
```

Out[17]:

```
AndersonResult(statistic=30.650605026187066, critical_values=array([0.
555, 0.632, 0.759, 0.885, 1.053]), significance_level=array([15. , 10.
,  5. ,  2.5,  1. ]))
```

Q-Q plot

`probplot` (https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.probplot.html#scipy.stats.probplot) `(x, sparams=(), dist='norm', fit=True, plot=None)`

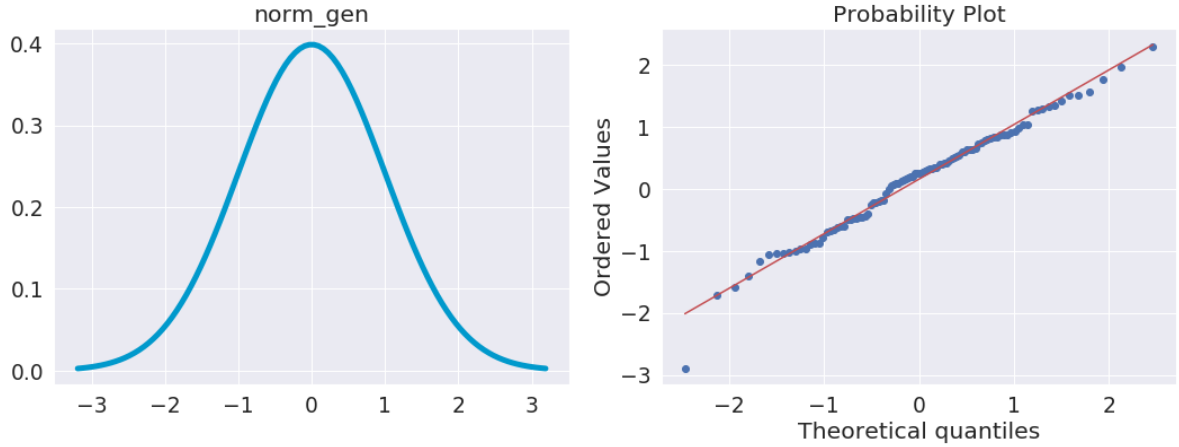
- `x` --- выборка
- `sparams` --- параметры распределения
- `dist` --- распределение
- `fit` --- подгонять ли линейную регрессию под точки
- `plot` --- рисовать на уже существующей фигуре

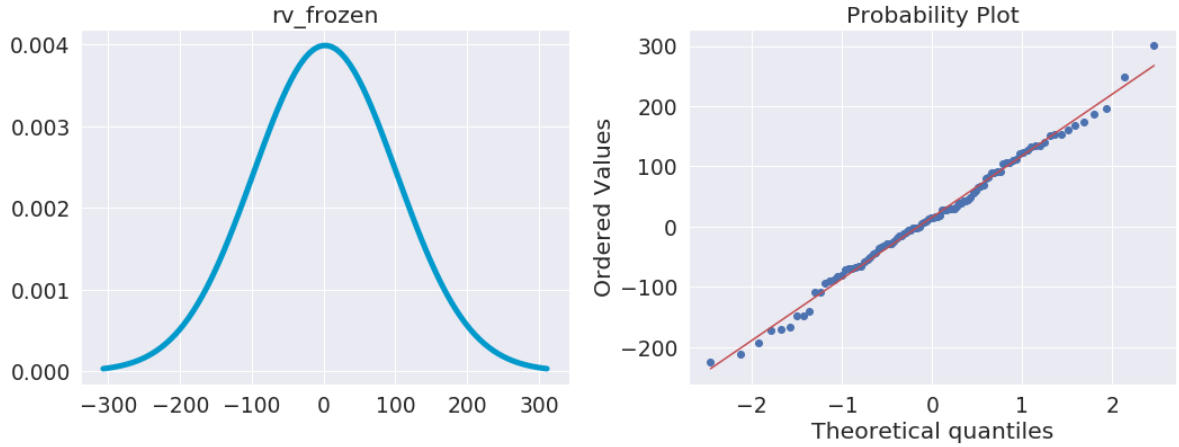
Возвращает:

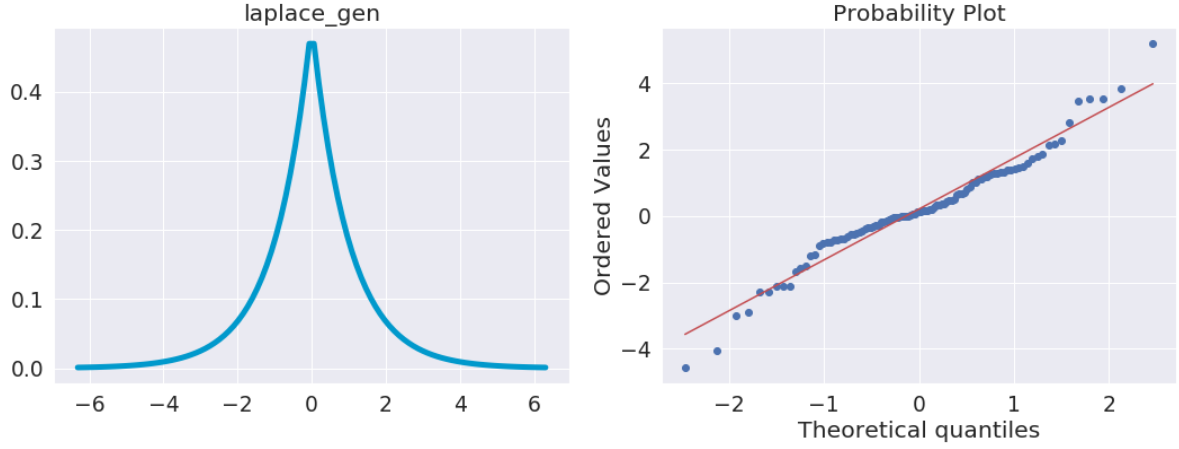
- `(osm, osr)` --- теоретические и выборочные квантили
- `(slope, intercept, r)` --- не возвращается в случае `fit=False` и `plot=None` . Значения: коэффициенты линейной регрессии и RSS

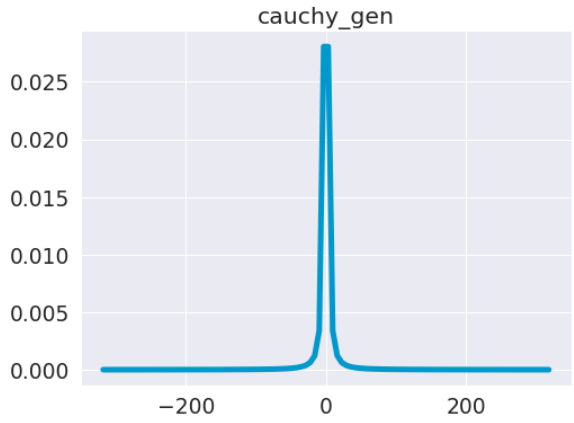
In [18]:

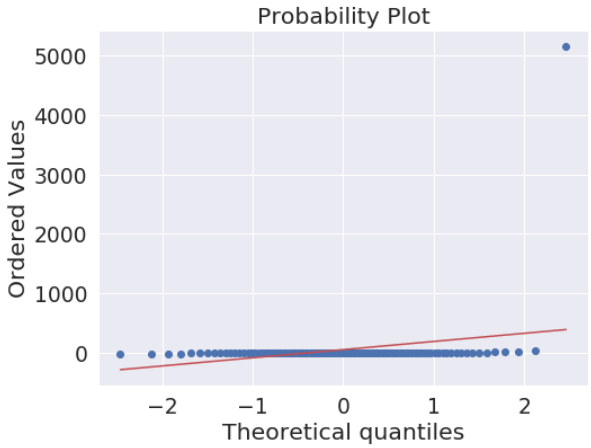
```
for distr in [sps.norm, sps.norm(2, 100), sps.laplace, sps.cauchy,
              sps.expon, sps.gamma(a=3), sps.uniform]:
    sample = distr.rvs(size=100)

    plt.figure(figsize=(15, 6))

    ax = plt.subplot(1, 2, 1)
    grid = np.linspace(distr.ppf(0.001)-0.1, distr.ppf(0.999)+0.1, 100)
    plt.plot(grid, distr.pdf(grid), lw=5, color=blue)
    plt.title(str(distr).split(' ')[0].split('.')[-1])

    ax = plt.subplot(1, 2, 2)
    sps.probplot(sample, plot=ax)

    plt.tight_layout()
    plt.show()
```

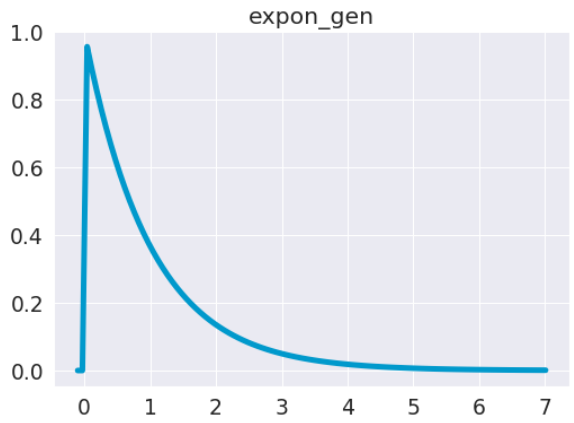


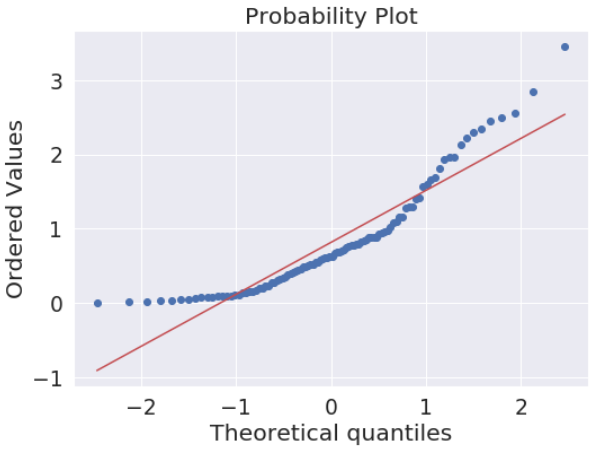


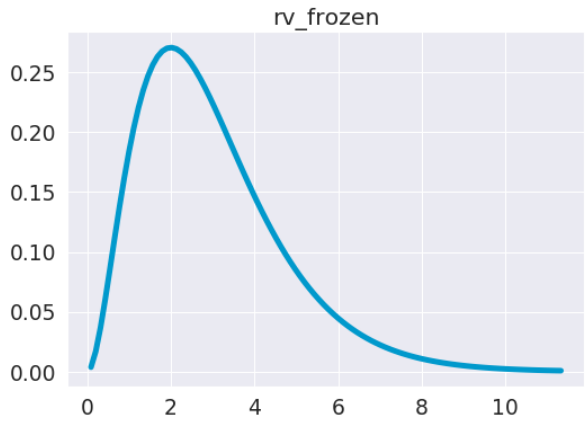


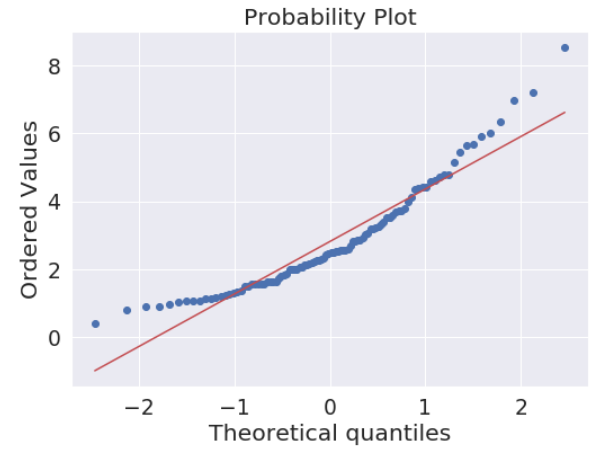

cauchy_gen
0.025
0.020
0.015
0.010
0.005
0.000
−200
0
200

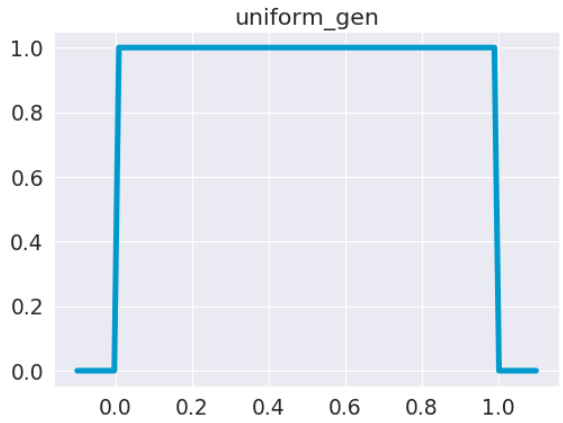

Probability Plot
Ordered Values
5000
4000
3000
2000
1000
0
−2
−1
0
1
2
Theoretical quantiles

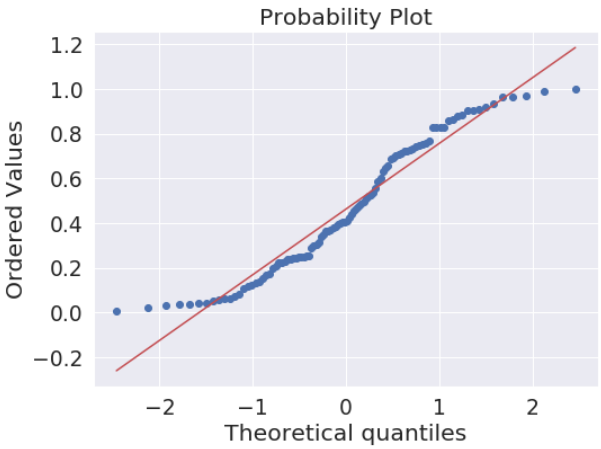

expon_gen
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
5
6
7


Probability Plot
Ordered Values
3
2
1
0
−1
−2
−1
0
1
2
Theoretical quantiles


rv_frozen
0.25
0.20
0.15
0.10
0.05
0.00
0
2
4
6
8
10


Probability Plot
Ordered Values
8
6
4
2
0
−2
−1
0
1
2
Theoretical quantiles


uniform_gen
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Probability Plot
Ordered Values
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−2
−1
0
1
2
Theoretical quantiles

Критерий Шапиро-Уилка

H_0: Выборка из нормального распределения

H_1: H_0: не верна.

shapiro (https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.shapiro.html#scipy.stats.shapiro)(x): W, p-value

In [19]:

```
1 sps.shapiro(sps.norm.rvs(size=100))
```

Out[19]:

```
(0.9901418089866638, 0.6760627031326294)
```

In [20]:

```
1 sps.shapiro(sps.norm(20, 100).rvs(size=100))
```

Out[20]:

```
(0.991076648235321, 0.7510301470756531)
```

In [21]:

```
1 sps.shapiro(sps.laplace.rvs(size=100))
```

Out[21]:

```
(0.9878672957420349, 0.498473197221756)
```

In [22]:

```
1 sps.shapiro(sps.cauchy.rvs(size=100))
```

Out[22]:

```
(0.2799574136734009, 6.384199354729874e-20)
```

Критерий Жарка-Бера

H_0: Выборка из нормального распределения

H_1: H_0 не верна.

jarque_bera (https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.jarque_bera.html#scipy.stats.jarque_bera)(x): jb_value, p-value

In [23]:

```
1 sps.jarque_bera(sps.norm.rvs(size=100))
```

Out[23]:

```
(7.107976892375481, 0.028610301142788175)
```

In [24]:

```
1 sps.jarque_bera(sps.norm(20, 100).rvs(size=100))
```

Out[24]:

```
(0.9921562789026134, 0.608914059003439)
```

In [25]:

```
1 sps.jarque_bera(sps.laplace.rvs(size=100))
```

Out[25]:

```
(21.054147137236328, 2.6800940745586033e-05)
```

In [26]:

```
1 sps.jarque_bera(sps.cauchy.rvs(size=100))
```

Out[26]:

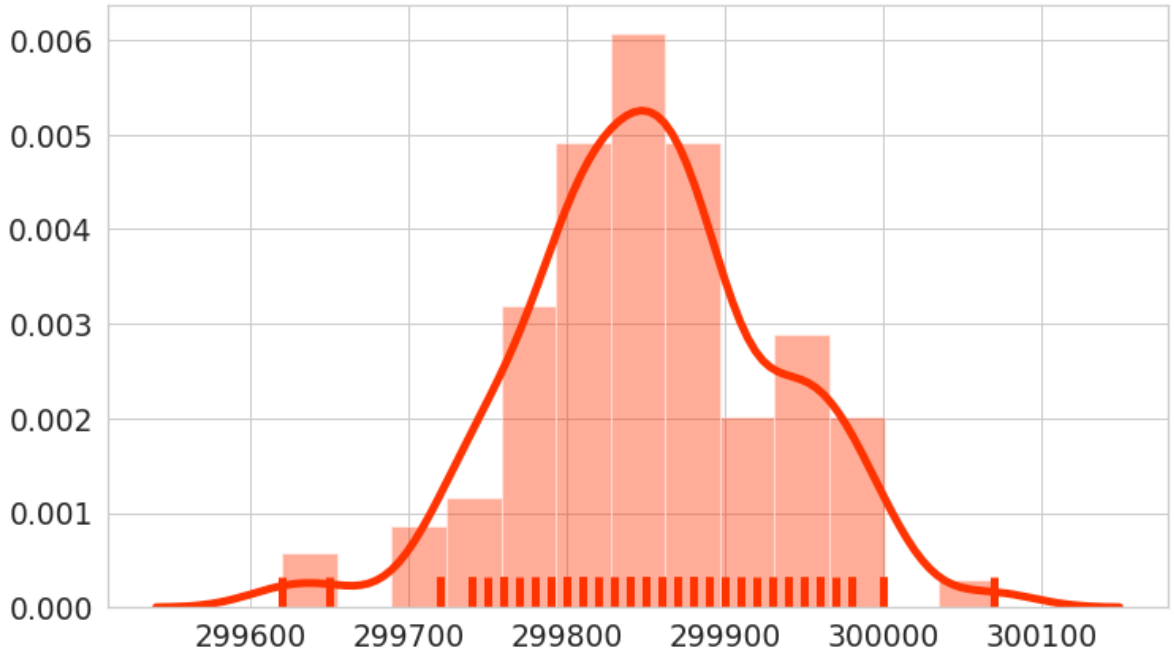
```
(28338.26644038969, 0.0)
```

Пример: эксперимент Майкельсона

Данные классического эксперимента Майкельсона по измерению скорости света с помощью вращающегося зеркала, 100 наблюдений:

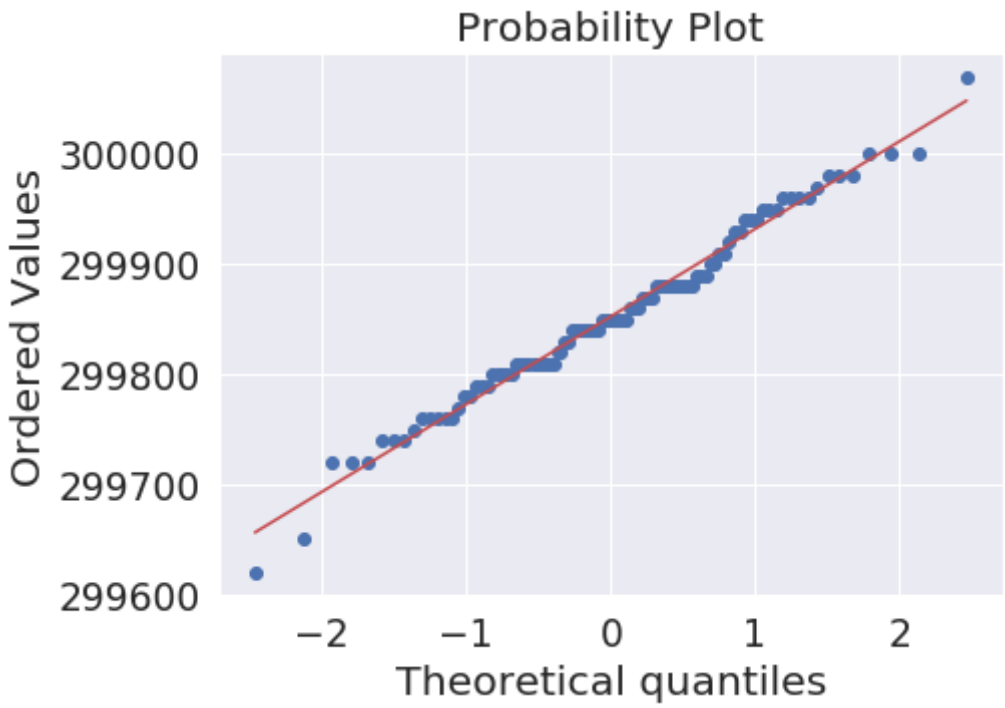
In [27]:

```
speed = np.loadtxt('speed.txt')

sns.set(style='whitegrid', font_scale=1.7)
plt.figure(figsize=(12, 7))
sns.distplot(speed, rug=True,
             color=red, kde_kws={'lw': 5}, rug_kws={'lw': 5});
```



In [28]:

```
sns.set(style='darkgrid', font_scale=1.7)
plt.figure(figsize=(7, 5))
sps.probplot(speed, plot=plt.subplot(111));
```



In [29]:

```
pvalues = [sps.kstest(speed, sps.norm(*sps.norm.fit(speed)).cdf)[1],
           sps.shapiro(speed)[1],
           sps.jarque_bera(speed)[1]]

pvalues
```

Out[29]:

```
[0.4813808419520875, 0.5140784382820129, 0.8628948044152577]
```

In [30]:

```
multipletests(pvalues)
```

Out[30]:

```
(array([False, False, False]),
 array([0.86050917, 0.86050917, 0.8628948 ]),
 0.016952427508441503,
 0.016666666666666666)
```
