In [43]:
```
from itertools import groupby
import numpy as np
from scipy import stats
import scipy.stats as sps
import matplotlib.pyplot as plt
import pandas as pd
from scipy.special import gamma, binom,gammaln,factorial
%matplotlib inline


N = [100,250, 500]
Alp = [0.01, 0.05, 0.1]
```

In [58]:
```
def wald_wolfowitz(X):
    mean = np.mean(X)
    sequence = np.sign(X - mean)
    n_runs = sum(1 for s in groupby(sequence, lambda a: a))

    n = float(sum(1 for s in sequence if s == sequence[0]))
    m = float(sum(1 for s in sequence if s != sequence[0]))

    mean  = ((2 * n * m ) / (n + m)) + 1
    var = (2 * n * m * (2 * n * m - n - m )) / ((n + m)**2 * (n + m - 1))
    if var**2 == 0:
        return None
    else:
        return (n_runs - mean) / var**2
```

In [32]:
```
DISTRIBUTIONS = [
    stats.alpha(a=3.57, loc=0.0, scale=1.0), stats.anglit(loc=0.0, scale=1.(
    stats.arcsine(loc=0.0, scale=1.0), stats.beta(a=2.31, b=0.627, loc=0.0,
    stats.betaprime(a=5, b=6, loc=0.0, scale=1.0), stats.bradford(c=0.299,
    stats.burr(c=10.5, d=4.3, loc=0.0, scale=1.0), stats.cauchy(loc=0.0, sca
    stats.chi(df=78, loc=0.0, scale=1.0), stats.chi2(df=55, loc=0.0, scale=1
    stats.cosine(loc=0.0, scale=1.0), stats.dgamma(a=1.1, loc=0.0, scale=1.(
    stats.dweibull(c=2.07, loc=0.0, scale=1.0), stats.erlang(a=2, loc=0.0, s
    stats.expon(loc=0.0, scale=1.0), stats.exponnorm(K=1.5, loc=0.0, scale=1
    stats.exponweib(a=2.89, c=1.95, loc=0.0, scale=1.0), stats.exponpow(b=2
    stats.f(dfn=29, dfd=18, loc=0.0, scale=1.0), stats.fatiguelife(c=29, loc
    stats.fisk(c=3.09, loc=0.0, scale=1.0), stats.foldcauchy(c=4.72, loc=0.(
    stats.foldnorm(c=1.95, loc=0.0, scale=1.0), stats.frechet_r(c=1.89, loc=
    stats.frechet_l(c=3.63, loc=0.0, scale=1.0), stats.genlogistic(c=0.412,
    stats.genpareto(c=0.1, loc=0.0, scale=1.0), stats.gennorm(beta=1.3, loc=
    stats.genexpon(a=9.13, b=16.2, c=3.28, loc=0.0, scale=1.0), stats.genex
    stats.gausshyper(a=13.8, b=3.12, c=2.51, z=5.18, loc=0.0, scale=1.0), st
    stats.gengamma(a=4.42, c=-3.12, loc=0.0, scale=1.0), stats.genhalflogist
    stats.gilbrat(loc=0.0, scale=1.0), stats.gompertz(c=0.947, loc=0.0, scal
    stats.gumbel_r(loc=0.0, scale=1.0), stats.gumbel_l(loc=0.0, scale=1.0),
    stats.halfcauchy(loc=0.0, scale=1.0), stats.halflogistic(loc=0.0, scale=
    stats.halfnorm(loc=0.0, scale=1.0), stats.halfgennorm(beta=0.675, loc=0
    stats.hypsecant(loc=0.0, scale=1.0), stats.invgamma(a=4.07, loc=0.0, sca
    stats.invgauss(mu=0.145, loc=0.0, scale=1.0), stats.invweibull(c=10.6,
    stats.johnsonsb(a=4.32, b=3.18, loc=0.0, scale=1.0), stats.johnsonsu(a=2
    stats.ksone(n=1e+03, loc=0.0, scale=1.0), stats.kstwobign(loc=0.0, scale
    stats.laplace(loc=0.0, scale=1.0), stats.levy(loc=0.0, scale=1.0),
    stats.levy_l(loc=0.0, scale=1.0), stats.levy_stable(alpha=0.357, beta=-0
    stats.logistic(loc=0.0, scale=1.0), stats.loggamma(c=0.414, loc=0.0, sca
    stats.loglaplace(c=3.25, loc=0.0, scale=1.0), stats.lognorm(s=0.954, loc
    stats.lomax(c=1.88, loc=0.0, scale=1.0), stats.maxwell(loc=0.0, scale=1
    stats.mielke(k=10.4, s=3.6, loc=0.0, scale=1.0), stats.nakagami(nu=4.97
    stats.ncx2(df=21, nc=1.06, loc=0.0, scale=1.0), stats.ncf(dfn=27, dfd=27
    stats.nct(df=14, nc=0.24, loc=0.0, scale=1.0), stats.norm(loc=0.0, scale
    stats.pareto(b=2.62, loc=0.0, scale=1.0), stats.pearson3(skew=0.1, loc=0
    stats.powerlaw(a=1.66, loc=0.0, scale=1.0), stats.powerlognorm(c=2.14, s
    stats.powernorm(c=4.45, loc=0.0, scale=1.0), stats.rdist(c=0.9, loc=0.0
    stats.reciprocal(a=0.00623, b=1.01, loc=0.0, scale=1.0), stats.rayleigh
    stats.rice(b=0.775, loc=0.0, scale=1.0), stats.recipinvgauss(mu=0.63, lo
    stats.semicircular(loc=0.0, scale=1.0), stats.t(df=2.74, loc=0.0, scale=
    stats.triang(c=0.158, loc=0.0, scale=1.0), stats.truncexpon(b=4.69, loc=
    stats.truncnorm(a=0.1, b=2, loc=0.0, scale=1.0), stats.tukeylambda(lam=1
    stats.uniform(loc=0.0, scale=1.0), stats.vonmises(kappa=3.99, loc=0.0, s
    stats.vonmises_line(kappa=3.99, loc=0.0, scale=1.0), stats.wald(loc=0.0
    stats.weibull_min(c=1.79, loc=0.0, scale=1.0), stats.weibull_max(c=2.87
    stats.wrapcauchy(c=0.0311, loc=0.0, scale=1.0)
]
```

In [60]:
```
def create_criteria_samples(n, size = 100, alpha = 2, beta = 2,gam = 0.5, de
    criteria_samples = []
    for i in range(size):
        samples = [1 * (np.round(distr.rvs(size=n)) % 2 ) for distr in DISTR
        for sample in samples:
            ww = wald_wolfowitz(sample)
            if (ww!=None):
                criteria_samples.append(ww)

    return criteria_samples
```

In [61]:
```
quantiles = []
for i in range(3):
    criteria_samples = create_criteria_samples(N[i], size = 10**1)
    qw = []
    for j in range(3):
        quantile = sps.mstats.mquantiles(criteria_samples,1 - (Alp[j])/2)
        qw.append(quantile)
        print('При размере выборки %d, alpha= %.2f : критерий: {K(X) >= %.2f
    quantiles.append(qw)
```

```
При размере выборки 100, alpha= 0.01 : критерий: {K(X) >= 52.06}
При размере выборки 100, alpha= 0.05 : критерий: {K(X) >= 2.05}
При размере выборки 100, alpha= 0.10 : критерий: {K(X) >= 0.39}
При размере выборки 250, alpha= 0.01 : критерий: {K(X) >= 14.30}
При размере выборки 250, alpha= 0.05 : критерий: {K(X) >= 1.13}
При размере выборки 250, alpha= 0.10 : критерий: {K(X) >= 0.04}
При размере выборки 500, alpha= 0.01 : критерий: {K(X) >= 10.23}
При размере выборки 500, alpha= 0.05 : критерий: {K(X) >= 0.86}
При размере выборки 500, alpha= 0.10 : критерий: {K(X) >= 0.03}
```

In [64]:
```
data = pd.read_csv('sample2.csv', header = None)
sample = np.array(data.values[:,0])

stat = wald_wolfowitz(sample)

print ("Критерий Вальда-Вольфовица говорит нам :" + str(np.abs(stat) >= quan
print ("значение статистики при этом: %.3f" % stat)
```

```
Критерий Вальда-Вольфовица говорит нам :[False] при уровне доверия 0.05
значение статистики при этом: -0.061
```

Ну то есть при уровне доверия в 0,1 можно