Федеральное государственное образовательное бюджетное учреждение высшего образования

Финансовый университет при Правительстве РФ

Факультет «Прикладная математика и информационные технологии»

Кафедра «Теория вероятностей и математическая статистика»

Лабораторная работа №1
Вариант №3
по дисциплине «Математическая статистика»

Работу выполнил
студент группы ПМ 2-2
Бахматов А. В.

Научный руководитель:

доцент, д.ф.-м.н.

Рябов П. Е.

Москва 2019

Характеристики ПК:

Тип процессора: Intel Core i5-4200M

Тактовая частота: 2,4 ГГц

Частота системной шины: 99.87 MHz

Объем кэша второго уровня: 2 x 256 KB 8-way

Лабораторная работа: пункт 1

Таблица числа торговых дней по компаниям варианта для каждого календарного года за период 2000-2019 (каждому календарному году соответствует отдельная колонка).

In [1]:
```
import IPython
import os
import pandas as pd
```
executed in 3.00s, finished 12:10:34 2019-04-04

In [2]:
```
%%javascript
var k = IPython.notebook.kernel;
k.execute('this_nb_name_ext = "' +
IPython.notebook.notebook_name +
'"');
```
executed in 28ms, finished 12:10:36 2019-04-04

In [3]:
```
thisfname = os.path.splitext(this_nb_name_ext)[0]   #Имя этого блокнота

def numberOfTradeDays(myDataPath = '',years = range(2019), tickers = None):

    def ndays(year,file):
        df = pd.read_csv(file)
        condition = (df['<DATE>']>=year*10000) & (df['<DATE>']<(year+1)*10000)
        return len(df[condition])

    def tickers_list(path):    возвращает список имен файлов, имеющих разрешение txt или csv
        files = []
        # r=root, d=directories, f = files
        for r, d, f in os.walk(path):
            for file in f:
                if '.txt' or '.csv' in file:
                    files.append(file[:-4])
        return files


    if not tickers:
        tickers = tickers_list(myDataPath)
    else:
        for i in range(len(tickers)):
            if '.txt' in tickers[i] or '.csv' in tickers[i]:
                tickers[i] = tickers[i][:-4]


    dfNDays = pd.DataFrame()
    dfNDays['Тикер'] = tickers

    for year in years:

        yearDays = []

        for ticker in tickers:
            yearDays.append(ndays(year, myDataPath + ticker + '.txt'))

        dfNDays[str(year)] = yearDays

    return dfNDays
```
executed in 26ms, finished 12:10:37 2019-04-04

In [4]:

```
years = range(2000,2020)
path = 'Data/'

df = numberOfTradeDays(path, years)
df.to_csv('Results/' + thisfname + ".Табл Число ТД.csv", index=False, sep=';', encoding = 'utf-8-sig')
```

executed in 7.26s, finished 12:10:45 2019-04-04

In [5]:

```
df
```

executed in 58ms, finished 12:10:47 2019-04-04

	Тикер	2000	2001	2002	2003	2004	2005	2006	2007	2008	...	2010	2011	2012	2013	2014	2015	2016	2017	201
0	APTK	0	0	0	19	0	1	188	248	245	...	248	248	255	250	250	250	252	252	254
1	AVAZ	60	229	250	249	250	248	248	193	245	...	248	248	255	250	250	250	252	252	229
2	OGKB	0	0	0	0	0	0	98	210	244	...	248	248	255	250	250	250	252	252	254
3	PLZL	0	0	0	0	0	0	164	248	246	...	248	248	255	250	250	250	252	252	254
4	VTBR	0	0	0	0	0	0	0	153	246	...	248	248	255	250	250	250	252	252	254

5 rows × 21 columns

Лабораторная работа: пункт 2

Таблица годовых стоимостных объемов торгов по компаниям варианта за весь период. Объем торгов рассчитывается в миллиардах с округлением до 0,1.

In [1]:
```
# Imports ↔
```
executed in 1.16s, finished 12:13:07 2019-04-04

In [2]:
```
%%javascript ↔
```
executed in 25ms, finished 12:13:11 2019-04-04

In [3]:
```
thisfname = os.path.splitext(this_nb_name_ext)[0]
#<TICKER>,<PER>,<DATE>,<TIME>,<OPEN>,<HIGH>,<LOW>,<CLOSE>,<VOL>
def yrvol(year,file,unit):
    csvtab = pd.read_csv(file)
    df = pd.DataFrame()
    df['date'] = csvtab['<DATE>']
    df['close']= csvtab['<CLOSE>']
    df['vol'] = csvtab['<VOL>']
    df['rvol'] = df['close']*df['vol']
    condition = (df['date']>=year*10000) & (df['date']<(year+1)*10000)
    return sum(df['rvol'][condition])/unit

def tickers_list(path):
    files = []
    # r=root, d=directories, f = files
    for r, d, f in os.walk(path):
        for file in f:
            if '.txt' or '.csv' in file:
                files.append(file[:-4])
    return files
```
executed in 17ms, finished 12:13:36 2019-04-04

In [4]:
```
myDataPath = 'Data/'
tickers = tickers_list(myDataPath)
years = range(2010,2019)
```
executed in 9ms, finished 12:13:39 2019-04-04

In [6]:
```
dfRVols = pd.DataFrame()
dfRVols['Тикер'] = tickers

for year in years:
    yearRVols = []

    for ticker in tickers:
        yearTickerRVol = yrvol(year, myDataPath + ticker + '.txt',10**9)
        yearRVols.append(round(yearTickerRVol,1))

    dfRVols[str(year)] = yearRVols

dfRVols.to_csv('Results/' + thisfname + ".Объем торгов в миллиардах.csv", index=False,
decimal=',', sep=';', encoding='utf-8-sig')
```
executed in 3.65s, finished 12:14:20 2019-04-04

In [7]:
```
dfRVols
```
executed in 44ms, finished 12:14:22 2019-04-04

	Тикер	2010	2011	2012	2013	2014	2015	2016	2017	2018
0	APTK	22.4	7.2	2.9	1.4	0.5	1.4	1.6	1.0	0.4
1	AVAZ	12.1	2.3	2.0	0.6	0.4	0.6	0.6	0.6	0.7
2	OGKB	22.2	16.9	13.5	10.4	4.7	6.2	7.2	13.6	8.0
3	PLZL	60.3	60.5	9.9	4.7	4.6	6.9	5.5	33.9	55.5
4	VTBR	642.7	658.4	495.2	483.8	564.1	407.9	254.3	184.8	259.2

Лабораторная работа: пункт 3

График цены закрытия первой (для данного варианта) компании за весь период.

In [4]:
```
# Imports ↔
```
executed in 21ms, finished 12:24:07 2019-04-04

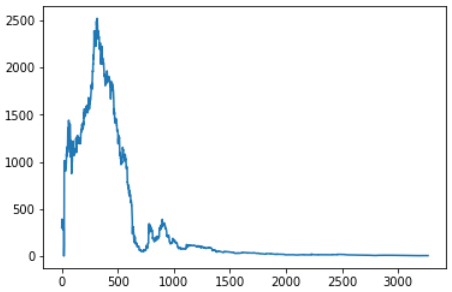
In [5]:
```
%%javascript ↔
```
executed in 11ms, finished 12:24:09 2019-04-04

```
Populating the interactive namespace from numpy and matpl
otlib
```

In [6]:
```
thisfname = os.path.splitext(this_nb_name_ext)[0]
def Close(y1,y2,file):
    csvtab = pd.read_csv(file)
    df = pd.DataFrame()
    df['date'] = csvtab['<DATE>']
    df['close']= csvtab['<CLOSE>']
    condition = (df['date']>=y1*10000) & (df['date']<(y2+1)*10000)
    return df['close'][condition]
```
executed in 20ms, finished 12:24:17 2019-04-04

In [7]:
```
# изменения формата графиков (для большего качества в pdf) - далее этот код добавляется в
импортах
ip = get_ipython()
ibe = ip.configurables[-1]
ibe.figure_formats = { 'pdf', 'png'}

myDataPath = 'Data/'
y = Close(2000,2019, myDataPath + 'APTK.txt')
plt.plot(y);
savefig('Results/' + thisfname + ".График цены закрытия.png");
```
executed in 3.56s, finished 12:24:22 2019-04-04



Лабораторная работа: пункт 4

Эмпирическая корреляционная 5x5 матрица дневных логарифмических доходностей всех компаний варианта за весь период (коэффициенты корреляции округляются до 0,001).

In [1]:
```
# Imports ↔
```
executed in 3.89s, finished 12:32:38 2019-04-04

In [4]:
```
%%javascript ↔
```
executed in 11ms, finished 12:34:05 2019-04-04

```
Populating the interactive namespace from numpy and matpl
otlib
```

In [5]:
```
thisfname = os.path.splitext(this_nb_name_ext)[0]

def tickers_list(path): ↔

years = range(2000,2020)
path = 'Data/'

df = pd.DataFrame()
tickers = tickers_list(path)
DF = []
for ticker in tickers:
    df0 = pd.read_csv(path + ticker)
    df0 = df0[['<DATE>','<CLOSE>']]
    condition = (df0['<DATE>']>=years[0]*10000) & (df0['<DATE>']<=years[-1]*10000)
    df0 = df0[condition].reset_index()
    DF.append(df0)

s = set(DF[0]['<DATE>'])
for i in DF:                #общие для всех компании даты торгов
    s = s & set(i['<DATE>'])
s = list(s)
s.sort()
s = pd.Series(s)

for i in range(len(DF)):
    DF[i] = DF[i][DF[i]['<DATE>'].isin(s)].reset_index()

M = []
for df in DF:
    l = []

    for i in range(len(df)-1):
        l.append(log(df['<CLOSE>'][i+1]/df['<CLOSE>'][i]))   #лог дневная доходность
    M.append(np.array(l,dtype=float))
```
executed in 23.3s, finished 12:34:30 2019-04-04

In [6]:
```
CC = pd.DataFrame(np.round(np.corrcoef(M),3))
CC
CC.to_csv('Results/' + thisfname + ".Эмпирическая Корр Матр ЛД.csv", index=False, header = False,
sep=';', encoding = 'utf-8-sig')
```
executed in 63ms, finished 12:34:33 2019-04-04

	0	1	2	3	4
0	1.000	0.123	0.334	0.201	0.331
1	0.123	1.000	0.238	0.090	0.168
2	0.334	0.238	1.000	0.254	0.500
3	0.201	0.090	0.254	1.000	0.379
4	0.331	0.168	0.500	0.379	1.000

Лабораторная работа: пункт 5

Таблица интервальных частот дневной логарифмической доходности первой(для данного варианта) компании за последний полный календарный год. Здесь и далее дневная логарифмическая доходность рассчитывается на основе поля «CLOSE» с коэффициентом 100.

In [1]:
```
# Imports ↔
```
executed in 3.96s, finished 12:43:35 2019-04-04

In [2]:
```
%%javascript ↔
```
executed in 13ms, finished 12:43:40 2019-04-04

```
Populating the interactive namespace from numpy and matpl
otlib
```

In [3]:
```
thisfname = os.path.splitext(this_nb_name_ext)[0]

def dquantile(a,q):   #труе функция квантиля
    from numpy import sort, size
    from math import floor
    a = sort(a)
    n = a.size

    if type(q) == float:
        if n*q % 1 == 0:
            return 0.5*(a[int(n*q-1)]+a[int(n*q)])
        else:
            return a[floor(n*q)]
    else:
        q = q.copy()
        for i in range(q.size):
            if q[i] == 0:
                q[i] = a[0]
            elif n*q[i] % 1 == 0:
                q[i] = 0.5*(a[int(n*q[i]-1)]+a[int(n*q[i])])
            else:
                q[i] = a[floor(n*q[i])]
        return q
def bin_interval(l):

    a = dquantile(l,0.01)
    a = round(a) - 5
    b = dquantile(l,0.99)
    b = round(b) + 5
    return np.arange(a,b+1)

year = 2018  #последний полный календарный год
path = 'Data/'
ticker = 'APTK.txt'
v = 3   #номер варианта
n  = 240 - 5*v   #колво используемых логдоходностей

df = pd.read_csv(path + ticker)
condition = (df['<DATE>']>=year*10000) & (df['<DATE>']<(year+1)*10000)
df = df[condition].reset_index()
df = df['<CLOSE>']

l = []
for i in range(len(df)-n-1,len(df)-1):    #считаем только последние n логдоходностей
    l.append(float(log(df[i+1]/df[i]))*100)
```
executed in 573ms, finished 12:43:48 2019-04-04

In [4]:

```
val, interval = np.histogram(l,bins=bin_interval(l))
val = list(map(float,val))   #сделаем все элементы в val вещественными
final = pd.DataFrame()

for i in l:   #Элемент матрицы, попавший вне всех интервалов, увеличивает на 1 частоту ближайшего
к нему интервала
    if i < interval[0]:
        val[0] += 1
    if i > interval[-1]:
        val[-1] += 1

for i in l:   #Элемент на границе интервалов добавляет 0.5 к каждому из них
    for j in range(len(interval)):
        if i == interval[j]:
            val[j-1] += 0.5
            val[j] -= 0.5

final['lo'] = interval[:-1]
final['hi'] = interval[1:]
final['fr'] = val
```

executed in 43ms, finished 12:43:50 2019-04-04

In [5]:

```
final.to_csv('Results/' + thisfname + ".Табл Набл Част ЛД#АРТК.csv", index=False, sep=';',
encoding = 'utf-8-sig')
```

executed in 19ms, finished 12:43:51 2019-04-04

In [6]:

```
final
```

executed in 32ms, finished 12:43:53 2019-04-04

	lo	hi	fr
0	-10.0	-9.0	0.0
1	-9.0	-8.0	0.0
2	-8.0	-7.0	0.0
3	-7.0	-6.0	1.0
4	-6.0	-5.0	1.0
5	-5.0	-4.0	6.0
6	-4.0	-3.0	5.0
7	-3.0	-2.0	12.0
8	-2.0	-1.0	25.0
9	-1.0	0.0	87.5
10	0.0	1.0	53.5
11	1.0	2.0	18.0
12	2.0	3.0	7.0
13	3.0	4.0	4.0
14	4.0	5.0	2.0
15	5.0	6.0	1.0
16	6.0	7.0	0.0
17	7.0	8.0	1.0
18	8.0	9.0	0.0
19	9.0	10.0	1.0

Лабораторная работа: пункт 6

Гистограмма частот, соответствующая таблице частот из предыдущего пункта, и график плотности нормального распределения в подходящих единицах измерения (на одном рисунке).

In [1]:
```
# Imports ↔
```
executed in 2.58s, finished 14:20:00 2019-04-04

```
Populating the interactive namespace from numpy and matpl
otlib
```

In [2]:
```
%%javascript ↔
```
executed in 19ms, finished 14:20:00 2019-04-04

In [3]:
```
thisfname = os.path.splitext(this_nb_name_ext)[0]
def dquantile(a,q):   #труе функция квантиля ↔
def bin_interval(l): ↔

file = 'Results/ПМ2-2 Бахматов Лаб Jupyter п5.Табл Набл Част ЛД#АРТК.csv'

DF = pd.read_csv(file, sep=';', engine='python', encoding='utf-8-sig')

lrs = (DF['lo']+DF['hi'])/2
n = sum(DF['fr'])
relfrs = DF['fr']/n
custm = stats.rv_discrete(name = 'custm', values=(lrs, relfrs))
mu = custm.mean()
sigma = custm.std()
```
executed in 127ms, finished 14:20:01 2019-04-04

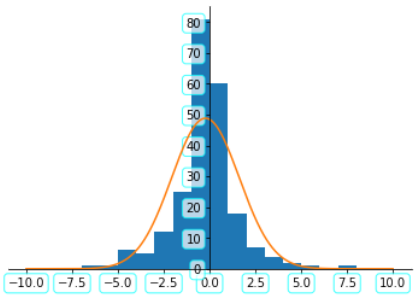
In [4]:
```
%store -r l #переменная из прошлого пункта
```
executed in 10ms, finished 14:20:02 2019-04-04

In [5]:
```
xx = np.arange(min(DF['lo']), max(DF['hi']), 0.01)
fitNL = stats.norm.pdf(xx, mu, sigma)

fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.spines['left'].set_position('center')   #настройка осей
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')

plt.hist(l, bins=bin_interval(l));
plt.plot(xx, fitNL*n);

bbox = dict(boxstyle="round", ec="cyan", fc="white", alpha=0.7)   #настройка отметок на осях
plt.setp(ax.get_xticklabels(), bbox=bbox);
plt.setp(ax.get_yticklabels(), bbox=bbox);
plt.savefig('Results/' + thisfname);
```
executed in 1.51s, finished 14:20:05 2019-04-04



Лабораторная работа: пункт 7

Таблица квантилей (уровни: 0,1; 0,2; …; 0,9) распределения эмпирического эксцесса. Эмпирический эксцесс рассчитывается по случайной выборке объема n = 240 − 5V (V — номер варианта), которая извлекается из стандартного нормального распределения m = 100000 раз. +файл, содержащий 1000 квантилей уровней: 0,0005; 0,0015; 0,0025 …; 0,9995.

In [1]:
```
# Imports ↔
```
executed in 2.68s, finished 12:55:48 2019-04-04

In [2]:
```
%%javascript ↔
```
executed in 16ms, finished 12:55:50 2019-04-04

```
Populating the interactive namespace from numpy and matpl
otlib
```

In [3]:
```
thisfname = os.path.splitext(this_nb_name_ext)[0]
def dquantile(a,q):   #труе функция квантиля ↔

from decimal import Decimal as Dec
def trurange(a,b,step):  #arange без погрешности
    a = Dec(str(a))
    b = Dec(str(b))
    step = Dec(str(step))
    return np.array(list(map(float,np.arange(a,b,step))))
```
executed in 34ms, finished 12:56:04 2019-04-04

In [4]:
```
v = 3
n = 240 - 5*v

lvs = np.arange(0.1, 1, 0.1)
Levels = trurange(0.0005, 1.0005, 0.001)
m = 100000
empEx = []
for i in range(m):
    empEx.append((stats.kurtosis(np.random.normal(loc=0.0, scale=1.0, size=n), fisher=True,
bias=True)))
```
executed in 37.7s, finished 12:56:46 2019-04-04

In [15]:
```
table1 = list(map(lambda x: round(x, 3), dquantile(empEx, lvs)))
table1, table1.index = pd.DataFrame(table1), lvs

table2 = list(map(lambda x: round(x, 3), dquantile(empEx, Levels)))
table2 = pd.DataFrame(table2)

table1.to_csv('Results/' + thisfname + '.Табл 9 квантилей по выборке объема n=' + str(n) +
'.csv', header = False, sep = ';', encoding = 'utf-8-sig')

table2.to_csv('Results/' + thisfname + '.Вектор 1000 квантилей по выборке объема n=' + str(n) +
'.csv', index = False, header = False, sep = ';', encoding = 'utf-8-sig')

pd.set_option('display.max_rows', 60)
```
executed in 143ms, finished 13:12:36 2019-04-04

In [16]:
```
table2
```

executed in 31ms, finished 13:12:37 2019-04-04

	0
0	-0.764
1	-0.714
2	-0.685
3	-0.664
4	-0.649
5	-0.637
6	-0.627
7	-0.617
8	-0.609
9	-0.602
10	-0.594
11	-0.587
12	-0.582
13	-0.576
14	-0.571
15	-0.566
16	-0.562
17	-0.557
18	-0.553
19	-0.548
20	-0.545
21	-0.541
22	-0.537
23	-0.533
24	-0.530
25	-0.527
26	-0.524
27	-0.521
28	-0.517
29	-0.514
...	...
970	0.663
971	0.670
972	0.677
973	0.687
974	0.696
975	0.705
976	0.715
977	0.726
978	0.734
979	0.743
980	0.757
981	0.769
982	0.783
983	0.797
984	0.810
985	0.826
986	0.842
987	0.860
988	0.882
989	0.900
990	0.921
991	0.946
992	0.973
993	1.007
994	1.054
995	1.102
996	1.158
997	1.257
998	1.394
999	1.711

1000 rows × 1 columns

In [11]:
```
table1
```

executed in 16ms, finished 13:10:18 2019-04-04

	0
0.1	-0.390
0.2	-0.289
0.3	-0.211
0.4	-0.137
0.5	-0.066
0.6	0.012
0.7	0.099
0.8	0.210
0.9	0.381

Лабораторная работа: пункт 8

Таблица вероятностей того, что эмпирический эксцесс, вычисленный по случайной выборке объема n из стандартного нормального распределения, окажется больше эмпирического эксцесса дневной логарифмической доходности, рассчитанного для каждой из 5 компаний варианта по n + 1 последним торговым дням календарного года (для каждого года за весь период).

In [1]:
```
# Imports ↔
```
executed in 17.4s, finished 19:53:01 2019-04-04

In [2]:
```
%%javascript↔
```
executed in 18ms, finished 19:53:03 2019-04-04

```
Populating the interactive namespace from numpy and matpl
otlib
```

In [ ]:
```
from collections import Counter
thisfname = os.path.splitext(this_nb_name_ext)[0]

def tickers_list(path): ↔
def date_filter(csvtab, year):
    condition = (csvtab['<DATE>']>=year*10000) & (csvtab['<DATE>']<(year + 1)*10000)
    return csvtab[condition]
#survival function
def data_Sf(data, csvtab_prob, g):
    prob = []
    data = data.reset_index()
    namecol = list(csvtab_prob.columns)
    for i in range(data.shape[0]):
        stack = 0
        for k in namecol:
            if data.loc[i][g] < float(k):
                stack += csvtab_prob.loc[0][k]
        prob.append(round(stack, 3))
    return prob

V = 3
n = 240 - 5*V

path = 'Data/'
years = range(2000, 2019)
q1000path = 'Results/ПМ2-2 Бахматов Лаб Jupyter п7.Вектор 1000 квантилей по выборке объема
n='+str(n)+'.csv'

qs = pd.read_csv(q1000path, sep=';', engine='python', encoding='utf-8', header = None,
decimal='.')[0]
tickers = tickers_list(path)
zer0 = pd.DataFrame(np.zeros((len(tickers), len(years))), columns = years)

for i in range(len(tickers)):
    ticDF = []
    Newtab = pd.read_csv(path + tickers[i] + '.txt', engine='python')
    for year in years:
        table = date_filter(Newtab, year)['<CLOSE>']

        if table.size < n:
            ticDF.append(-10) #устанавливаем значение не 0 так как при X > a возникнут проблемы
            continue
        log = np.log(table.divide(table.shift(+1))) #логарифмическая доходность
        log = log.dropna() #удаление NaN
        End_year = log.tail(n) #последние n элементов
        ticDF.append(stats.kurtosis(End_year, fisher=True))
    zer0.loc[i][:] = pd.Series(ticDF)
```

In [3]:
```
csvtab_prob = pd.DataFrame(Counter(qs), index=[0]) / qs.size #распределение относительных частот

for i in range(len(tickers)):
    zer0.loc[i] = data_Sf(zer0.loc[i], csvtab_prob, i)

zer0['Тикеры'] = tickers
zer0.set_index('Тикеры', inplace = True)

zer0.to_csv('Results/' + thisfname + ".Табл p-values.csv", index=True, decimal=',', sep=';',
encoding='utf-8-sig')
```

executed in 48.4s, finished 19:53:52 2019-04-04

In [4]:
```
zer0
```

executed in 129ms, finished 19:53:54 2019-04-04

	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018
Тикеры																			
APTK	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.000	0.0	0.000	0.0	0.000	0.00	0.000	0.0	0.000	0.0	0.000	0.0
AVAZ	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.000	0.0	0.000	0.0	0.000	0.00	0.000	0.0	0.000	0.0	0.000	0.0
OGKB	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.000	0.0	0.000	0.0	0.000	0.00	0.000	0.0	0.000	0.0	0.021	0.0
PLZL	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.002	0.0	0.000	0.0	0.003	0.00	0.000	0.0	0.000	0.0	0.000	0.0
VTBR	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.000	0.0	0.007	0.0	0.001	0.01	0.002	0.0	0.018	0.0	0.000	0.0

Лабораторная работа: пункт 9

Гистограмма полученных в пункте 8 вероятностей

In [5]:
```
#переведём таблицу в вектор без 1
vector = zer0.values.tolist()
one_list = []
for i in range(len(vector)):
    for k in range(len(vector[0])):
        if vector[i][k] < 1:
            one_list.append(vector[i][k])

plt.hist(one_list);
savefig('Results/' + thisfname);
```

executed in 1.02s, finished 19:53:59 2019-04-04