Задание 1

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

In [127]:

```
# время выполнения 4.1083 s
from IPython.display import Image
import os
import pandas as pd
from datetime import datetime
import seaborn as sns
```

In [128]:

```
%%javascript
var k = IPython.notebook.kernel;
k.execute('this_nb_name_ext = ”' + IPython.notebook.notebook_name +
 ↪  '”');
```

```
[breaklines = true]
<IPython.core.display.Javascript object>
```

In [129]:

```
def thisfname(): #Имя этого блокнота
    this_notebook_name = os.path.splitext(this_nb_name_ext)[0]
    return this_notebook_name
```

In [130]:

```
def ndays(year,file):
    df = pd.read_csv(file, sep = ',' , engine='python') # чтение файла
    df['<DATE>'] = pd.to_datetime(df['<DATE>'], format='%Y%m%d')
    condition = (df['<DATE>'] >= datetime(year, 1, 1)) & (df['<DATE>']
     ↪  < datetime(year + 1, 1, 1))
    return len(df[condition])

# преобразлвание в latex таблицу
def _repr_latex_(self):
    return r'\scalebox{0.7}{\tabcolsep=0.11cm\centering{%s}}' %
     ↪  self.to_latex()

pd.set_option('display.notebook_repr_html', True)
pd.DataFrame._repr_latex_ = _repr_latex_  # monkey patch pandas
 ↪  DataFrame
```

In [131]:

```
tickers = ['AKRN', 'APTK', 'CHMK', 'LKOH', 'SBER'] # Тикеры
years = range(2000, 2019)
myDataPath = 'C:/Users/timha/OneDrive/Рабочий стол/лаба/'
```

In [132]:

```
dfNDays = pd.DataFrame(index = tickers)
for year in years:
    yearDays = []
    for ticker in tickers:
        yearDays.append(ndays(year, myDataPath + ticker + '.txt'))
    dfNDays[str(year)] = yearDays
dfNDays
```

Out [132]:

	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018
AKRN	0	0	0	0	0	0	50	248	245	249	248	248	255	250	250	250	252	252	254
APTK	0	0	0	19	0	1	188	248	245	248	248	248	255	250	250	250	252	252	254
CHMK	0	0	0	0	0	0	0	0	1	137	248	247	254	242	246	250	252	252	254
LKOH	249	251	250	249	250	248	248	248	246	249	248	248	255	250	250	250	252	252	254
SBER	249	251	250	249	250	248	238	246	246	249	248	248	255	250	250	250	252	252	254

In [133]:

```
dfNDays.to_csv(thisfname() + ".Табл Число ТД.csv", index=False,
 ↪  decimal=',', sep=';', encoding='utf-8-sig')
```

Как видно из таблицы не все компании торговались с 2000 по 2007 год, но в остальный года торги проводились во всех компаниях, поэтому с ними можно проводить дальнейшие исследования.

Задание 2

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

In [14]:

```
# время выполнения 4.18433 s
import IPython
import os
import pandas as pd
from datetime import datetime
```

In [21]:

```
%%javascript
var k = IPython.notebook.kernel;
k.execute('this_nb_name_ext = "' + IPython.notebook.notebook_name +
 ↪  '"');
```

```
[breaklines = true]
<IPython.core.display.Javascript object>
```

In [27]:

```
def thisfname(): #Имя этого блокнота
    this_notebook_name = os.path.splitext(this_nb_name_ext)[0]
    return this_notebook_name

# преобразлвание в latex таблицу
def _repr_latex_(self):
    return r'\scalebox{0.7}{\small\tabcolsep=0.11cm\centering{%s}}' %
     ↪  self.to_latex()

pd.set_option('display.notebook_repr_html', True)
pd.DataFrame._repr_latex_ = _repr_latex_  # monkey patch pandas
 ↪  DataFrame
```

In [28]:

```
#<TICKER>,<PER>,<DATE>,<TIME>,<OPEN>,<HIGH>,<LOW>,<CLOSE>,<VOL>
def yrvol(year,file,unit):
    csvtab = pd.read_csv(file, sep = ',', engine='python') # чтение
     ↪  файла и преобразование даты
    df = pd.DataFrame()
    df['date'] = pd.to_datetime(csvtab['<DATE>'], format='%Y%m%d') #
     ↪  преобразование '<DATE>' к datetime
    df['close']= csvtab['<CLOSE>'] # столбец df.close = csvtab.<CLOSE>
    df['vol'] = csvtab['<VOL>'] # столбец df.vol = csvtab.<VOL>
    df['rvol'] = df['close']*df['vol'] # df.close * df.vol
    condition = (df['date']>=datetime(year, 1, 1)) &
     ↪  (df['date']<datetime(year + 1, 1, 1)) # проверка на дату
    return sum(df['rvol'][condition])/unit # вернуть df.loc[df.close *
     ↪  df.vol][проверка на дату]
```

In [29]:

```
tickers = ['AKRN', 'APTK', 'CHMK', 'LKOH', 'SBER'] # Тикеры
myDataPath = 'C:/Users/timha/OneDrive/Рабочий стол/лаба/' # путь к
 ↪  файлам
years = range(2000, 2019) # дата
```

In [30]:

```
dfRVols = pd.DataFrame(index = tickers)
for year in years:
    yearRVols = []
    for ticker in tickers:
        yearTickerRVol = yrvol(year, myDataPath + ticker + '.txt',
         ↪  1000000000) # расчёт стоимостных объемов торгов
        yearRVols.append(round(yearTickerRVol,1)) # округление
    dfRVols[str(year)] = yearRVols
dfRVols
```

Out [30]:

	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018
AKRN	0.0	0.0	0.0	0.0	0.0	0.0	0.1	1.0	5.9	6.0	8.4	13.7	6.2	4.5	4.7	5.5	4.6	4.8	4.7
APTK	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.7	0.7	14.9	22.4	7.2	2.9	1.4	0.5	1.4	1.6	1.0	0.4
CHMK	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.9	6.6	1.2	0.2	0.1	0.1	0.7	0.3	0.3	0.1
LKOH	22.9	43.3	95.2	191.5	351.5	715.4	1688.1	1391.7	1571.2	991.8	762.8	964.2	691.3	467.4	681.0	711.3	597.8	464.1	868.7
SBER	6.7	4.5	22.3	21.7	74.9	94.1	409.7	1145.0	1612.6	4915.6	4075.7	5079.8	2970.2	1988.9	2674.0	2292.8	2228.6	2037.1	3507.0

In [31]:

```
dfRVols.to_csv(thisfname() + ".Объем торгов в миллиардах.csv",
 ↪  index=False, decimal=',', sep=';', encoding='utf-8-sig')
```

Как видно из таблицы наибольший объём торгов у Сбербанка, затем у компании Лукойл.

Задание 3

График цены закрытия первой (для данного варианта) компании за весь период. Здесь и далее цена закрытия рассчитывается на основе поля «CLOSE».

In [1]:

```
# время выполнения 0.217276 s
import IPython
import os
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime
```

In [5]:

```
%%javascript
var k = IPython.notebook.kernel;
k.execute('this_nb_name_ext = "' + IPython.notebook.notebook_name +
 ↪  '"');
```

```
[breaklines = true]
<IPython.core.display.Javascript object>
```

In [3]:

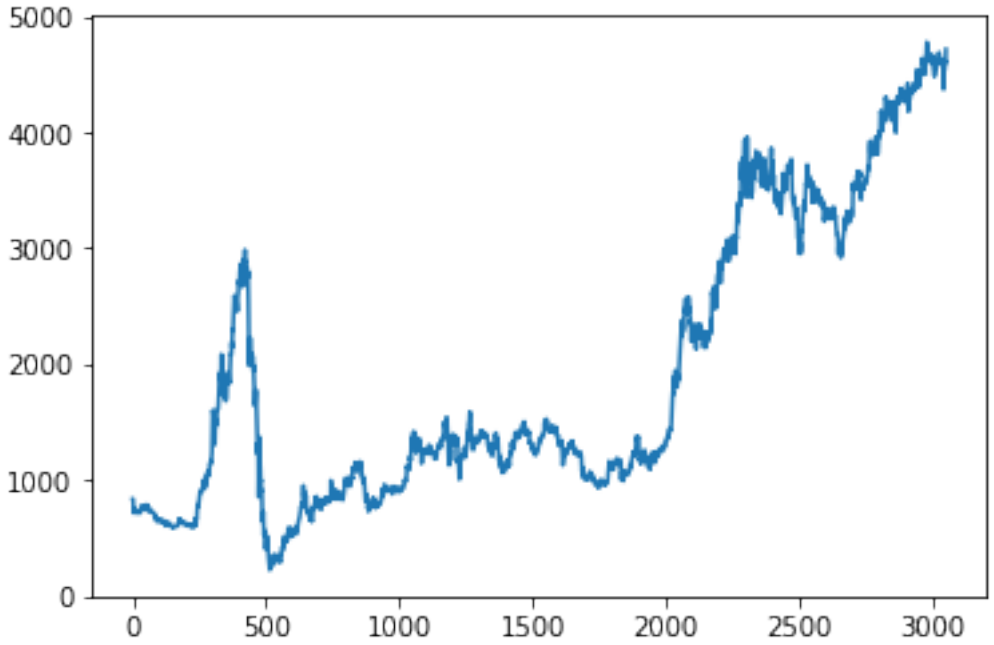
```
def thisfname(): #Имя этого блокнота
    this_notebook_name = os.path.splitext(this_nb_name_ext)[0]
    return this_notebook_name


def Close(y1,y2,file):
    csvtab = pd.read_csv(file , sep = ',', engine = 'python')
    df = pd.DataFrame()
    df['date'] = pd.to_datetime(csvtab['<DATE>'], format='%Y%m%d') #
     ↪  преобразование '<DATE>' к datetime
```

```
    df['close'] = csvtab['<CLOSE>'] # столбец df.close = csvtab.<CLOSE>
    condition = (df['date']>=datetime(y1, 1, 1)) &
     ↪  (df['date']<datetime(y2 + 1, 1, 1)) # проверка на дату
    return df['close'][condition]
```

In [6]:

```
myDataPath = 'C:/Users/timha/OneDrive/Рабочий стол/лаба/'
y = Close(2006, 2018, myDataPath + 'AKRN.txt')
plt.plot(y)
plt.savefig(thisfname() + ".График цены закрытия.png")
```



Из графтка видно, что цена закрытия сначала резко возросла, а затем резко упала, после чего цена закрытия постепенно росла.

Задание 4

Эмпирическая корреляционная 5x5 матрица дневных логарифмических доходностей всех компаний варианта за весь период (коэффициенты корреляции округляются до 0,001). Период может быть сужен при отсутствии данных.

In [1]:

```
# время выполния 0.426256 s
import IPython
import os
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

```
from datetime import datetime
import seaborn as sns
```

In [7]:

```
%%javascript
var k = IPython.notebook.kernel;
k.execute('this_nb_name_ext = ”' + IPython.notebook.notebook_name +
 ↪  '”');
```

```
[breaklines = true]
<IPython.core.display.Javascript object>
```

In [3]:

```
def thisfname(): #Имя этого блокнота
    this_notebook_name = os.path.splitext(this_nb_name_ext)[0]
    return this_notebook_name
```

In [4]:

```
tickers = ['AKRN', 'APTK', 'CHMK', 'LKOH', 'SBER'] # Тикеры
myDataPath = 'C:/Users/timha/OneDrive/Рабочий стол/лаба/' # путь к
 ↪  файлам
```

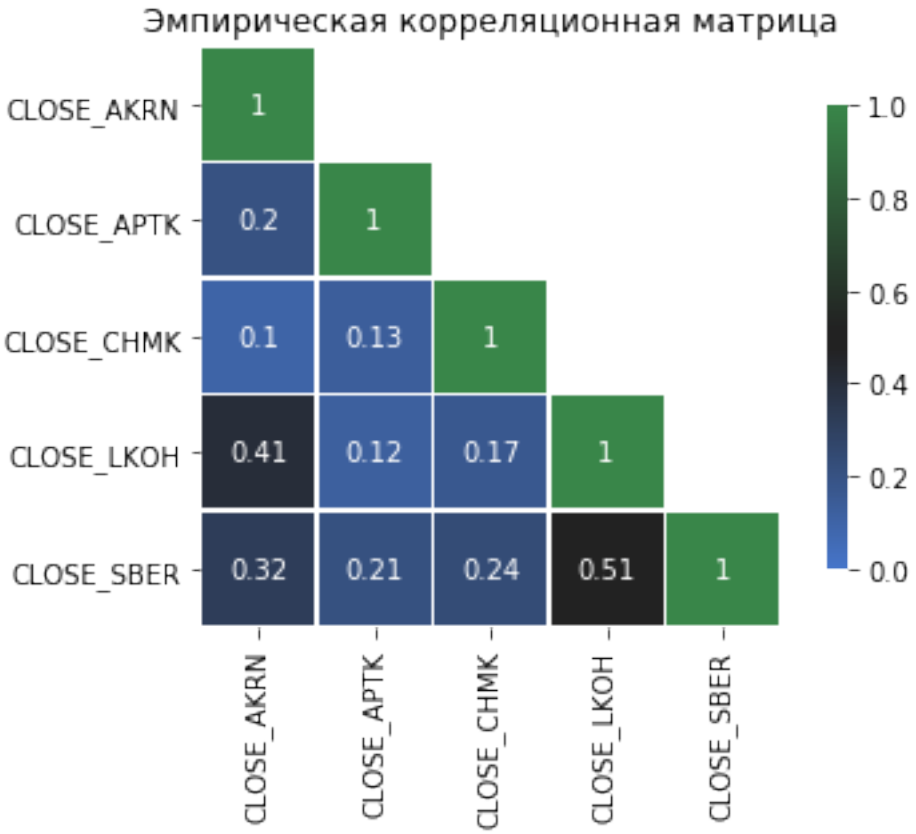
In [5]:

```
# чтение и преобразование файла
def read_exp(file, ticker):
    df = pd.DataFrame()
    csvtab = pd.read_csv(file, sep = ',' , engine='python') # чтение
     ↪  файла
    df['DATA_' + ticker] = pd.to_datetime(csvtab['<DATE>'],
     ↪  format='%Y%m%d')
    df['CLOSE_' + ticker] = csvtab['<CLOSE>']
    df.set_index('DATA_' + ticker, inplace = True)
    return df

# красивый вывод ковариационной матрицы
def pretty_corr(newtab):
    mask = np.zeros_like(newtab)
#     print(np.triu_indices_from(mask))
    mask[np.triu_indices_from(mask, +1)] = True
    cmap = sns.diverging_palette(255, 133, center=”dark”, as_cmap =
     ↪  True)
    ax = sns.heatmap(
        newtab,
        vmin=0, vmax=1,
        annot=True,
        mask = mask,
        cmap = cmap,
        square=True,
        linewidths=.5,
```

```
        cbar_kws={"shrink": .8}
    )
    ax
    plt.title('Эмпирическая корреляционная матрица')
    plt.show()

# чтение файлов
df_AKRN = df_APTK = df_CHMK = df_LKOH = df_SBER = pd.DataFrame()
list_of_data = [df_AKRN, df_APTK, df_CHMK, df_LKOH, df_SBER]
# преобразуем для concat
for i in range(len(list_of_data)):
    list_of_data[i] = read_exp(myDataPath + tickers[i] + '.txt',
     ↪  tickers[i])
# соединяем все таблицы в одну
newtab = pd.concat(list_of_data, join='inner', axis = 1) # таблица не
 ↪  содержащая пропусков
newtab = np.log(newtab.divide(newtab.shift(+1))) # логарифмическая
 ↪  доходность
newtab = newtab.dropna() # удаление NaN
newtab = round(newtab.corr(method='pearson'), 3) # коэффициент
 ↪  корреляции с округлением
pretty_corr(newtab)
```



In [8]:

```
newtab.to_csv(thisfname() + ".Эмпирическая Корр Матр ЛД.csv",
 ↪  index=False, decimal=',', sep=';', encoding='utf-8-sig')
```

Как видно из таблицы, наибольшая линейная зависимость возникает между логарифмическим доходностями компаний Сбербанк и Лукойл. Стоит отметить, что показатели коэффициента корреляции между логарифмическими доходностями компаний довольно сильно разница в значениях.

Задание 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 [29]:

```
# время выполнения 0.297988 s
import IPython
import os
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from datetime import datetime
import scipy.stats as st
from math import floor
```

In [30]:

```
%%javascript
var k = IPython.notebook.kernel;
k.execute('this_nb_name_ext = ”' + IPython.notebook.notebook_name +
 ↪  '”');
```

```
[breaklines = true]
<IPython.core.display.Javascript object>
```

In [31]:

```
# некоторые вспомогательные функии
def thisfname(): #Имя этого блокнота
    this_notebook_name = os.path.splitext(this_nb_name_ext)[0]
    return this_notebook_name

# читает файл AKRN.txt
def read_AKRN(file, year):
    csvtab = pd.read_csv(file, sep=',', engine='python') # чтение файла
    df = pd.DataFrame()
    df['date'] = pd.to_datetime(csvtab['<DATE>'], format='%Y%m%d')
    df['CLOSE_AKRN'] = csvtab['<CLOSE>']
    df = df[(df['date']>=datetime(year, 1, 1)) &
     ↪  (df['date']<datetime(year + 1, 1, 1))]
    df.set_index('date', inplace = True)
    return df
```

```
# расчитывается квантиль
def norm_quantile(newtab, level):
    newtab = np.sort(newtab)
    if (newtab.shape[0] * level) % 1 != 0:
        return newtab[int(floor(newtab.shape[0] * level))]
    elif (newtab.shape[0] * level) % 1 == 0:
        return (newtab[int(floor(newtab.shape[0] * level)) - 1] +
         ↪  newtab[int(floor(newtab.shape[0] * level))]) / 2

# преобразлвание в latex таблицу
def _repr_latex_(self):
    return self.to_latex()

pd.set_option('display.notebook_repr_html', True)
pd.DataFrame._repr_latex_ = _repr_latex_  # monkey patch pandas
 ↪  DataFrame
```

In [32]:

```
# 5
tickers = ['AKRN', 'APTK', 'CHMK', 'LKOH', 'SBER'] # Тикеры
myDataPath = 'C:/Users/timha/OneDrive/Рабочий стол/лаба/' # путь к
 ↪  файлам
year = 2018 # последний полный год
```

In [33]:

```
newtab = read_AKRN(myDataPath + tickers[0] + '.txt', year) # чтение
 ↪  файла
newtab = 100*np.log(newtab.divide(newtab.shift(+1))) # логарифмическая
 ↪  доходность за 2018 год
newtab = newtab.dropna() # удаление NaN
newtab['prob'] = 1/newtab.shape[0]
# таблица частот
low = int(round(norm_quantile(newtab['CLOSE_AKRN'], 0.01)) - 5)
hight = int(round(norm_quantile(newtab['CLOSE_AKRN'], 0.99)) + 5)
interval = np.arange(low, hight + 1, 1)
interval[0] -= 1 # для включения нижней границы
interval[len(interval) - 1] += 1
# интервал в одном случае попал в другом нет
new_1 = np.array(pd.cut(newtab['CLOSE_AKRN'], interval, right =
 ↪  False).value_counts(sort = False))
new_2 = np.array(pd.cut(newtab['CLOSE_AKRN'], interval, right =
 ↪  True).value_counts(sort = False))
new_3 = (new_1 + new_2)/2
final = pd.DataFrame()
# таблица частот готова
final['lo'], final['hi'], final['fr'] = np.arange(low, hight, 1),
 ↪  np.arange(low + 1, hight + 1, 1), new_3
final
```

Out [33]:

	lo	hi	fr
0	-8	-7	0.0
1	-7	-6	0.0
2	-6	-5	0.0
3	-5	-4	0.0
4	-4	-3	2.0
5	-3	-2	4.0
6	-2	-1	11.0
7	-1	0	93.0
8	0	1	115.0
9	1	2	23.0
10	2	3	3.0
11	3	4	2.0
12	4	5	0.0
13	5	6	0.0
14	6	7	0.0
15	7	8	0.0

In [34]:

```
final.to_csv(thisfname() + ".Объем торгов в миллиардах.csv",
 ↪  index=False, decimal=',', sep=';', encoding='utf-8-sig')
```

На основании таблицы можно сделать вывод, что значения логарифмической доходности наиболее плотно распределены в окрестностях 8.

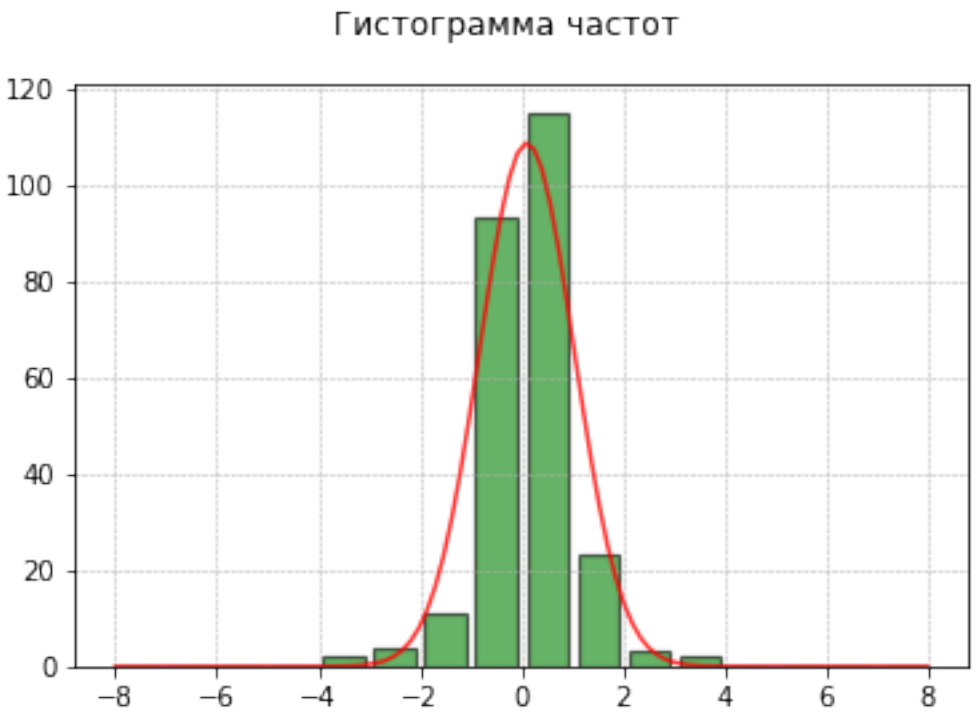
Задание 6

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

In [35]:

```
# 6
div2 = (final['lo'] + final['hi'])/2
n = final['fr'].sum()
prob = final['fr']/n
dlaw = pd.DataFrame()
dlaw['val'], dlaw['prob'] = div2, prob # дискретное распределение
mu = (dlaw['val']*dlaw['prob']).sum() # мат. ожидание
var = [(((dlaw['val']**2)*dlaw['prob']).sum() - mu**2)**(0.5)] #
 ↪  дисперсия
# нормальное распределение
plt.plot(np.linspace(low, hight, 100), st.norm.pdf(np.linspace(low,
 ↪  hight, 100), mu, var)*n , 'r', alpha = 0.8)
```

```
# гистограмма
x = np.arange(low + 0.5, hight + 0.5, 1)
y = final['fr'].values.tolist()
plt.bar(x, y, color = 'green', alpha = 0.6, edgecolor='black',
 ↪  linewidth = 1.2)
plt.suptitle('Гистограмма частот')
plt.grid(linestyle='--', linewidth=0.5)
plt.show()
```



На основе данной гистограммы можно сделать вывод, что логарифмическая доходность приблизительно соответствует непрерывному нормальному распределению. В то же время, не сложно убедиться, что реальные данные сильно выбиваются из математической модели непрерывного нормального распределения.

Задание 7

Таблица квантилей (уровни: $0,1; 0,2; \ldots; 0,9$) 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. Кроме того, создайте файл, содержащий 1000 квантилей уровней: $0,0005; 0,0015; 0,0025; \ldots; 0,9995$.

In [21]:

```
# время выполнения 13.003 s
import IPython
import os
```

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from datetime import datetime
import scipy.stats as st
from math import floor
```

In [22]:

```
%%javascript
var k = IPython.notebook.kernel;
k.execute('this_nb_name_ext = ”' + IPython.notebook.notebook_name +
 ↪  '”');
```

```
[breaklines = true]
<IPython.core.display.Javascript object>
```

In [23]:

```
# некоторые вспомогательные функии
def thisfname(): #Имя этого блокнота
    this_notebook_name = os.path.splitext(this_nb_name_ext)[0]
    return this_notebook_name

# расчитывается квантиль
def norm_quantile(newtab, level):
    newtab = np.sort(newtab)
    if (newtab.shape[0] * level) % 1 != 0:
        return newtab[int(floor(newtab.shape[0] * level))]
    elif (newtab.shape[0] * level) % 1 == 0:
        return (newtab[int(floor(newtab.shape[0] * level)) - 1] +
         ↪  newtab[int(floor(newtab.shape[0] * level))]) / 2

# преобразлвание в latex таблицу
def _repr_latex_(self):
    return self.to_latex()

pd.set_option('display.notebook_repr_html', True)
pd.DataFrame._repr_latex_ = _repr_latex_  # monkey patch pandas
 ↪  DataFrame
```

In [24]:

```
V = 1 # вариант
n = 240 - 5*V # 235
levels = np.arange(0.1, 1, 0.1) # уровни квантилей
m = 100000 # объём
Levels = np.arange(0.0005, 1, 0.001) # уровни квантилей
y = st.norm.pdf([-1, 1], 0, 1) # нормальное распределение
zero = [0]*m
for i in range(m):
    zero[i] = st.kurtosis(np.random.normal(0, 1, n), fisher=True) #
     ↪  выбока из нормального распределения и подсчёт эксцесса
```

```
ewal = pd.DataFrame() # таблица распределения относительных частот
ewal['val'] = zero
ewal['prob'] = 1/m # вероятности
ewal['val'] = np.sort(ewal['val']) #сортировка
tab_Quantile = []
for i in range(len(Levels)): # Вектор 1000 квантилей по выборке объема
 ↪  n=235
    tab_Quantile.append(round(norm_quantile(ewal['val'], Levels[i]),
     ↪  3)) # таблица квантилей Levels
tab_Quantile = pd.DataFrame(tab_Quantile)
tab_quantile = []
for k in range(len(levels)): # Табл 9 квантилей по выборке объема n=235
    tab_quantile.append(round(norm_quantile(ewal['val'], levels[k]),
     ↪  3))
final = pd.DataFrame()
final['quan'] = levels
final['prob'] = tab_quantile
final
```

Out [24]:

	quan	prob
0	0.1	-0.382
1	0.2	-0.282
2	0.3	-0.205
3	0.4	-0.134
4	0.5	-0.065
5	0.6	0.011
6	0.7	0.097
7	0.8	0.205
8	0.9	0.375

In [25]:

```
tab_Quantile.to_csv(thisfname() + ".Вектор 1000 квантилей по выборке
 ↪  объема n=235.csv", index=False, decimal=',', sep=';',
 ↪  encoding='utf-8-sig')
final.to_csv(thisfname() + ".Табл 9 квантилей по выборке объема
 ↪  n=235.csv", index=False, decimal=',', sep=';',
 ↪  encoding='utf-8-sig')
```

Задание 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). Загрузите найденные в предыдущем пункте квантили

в вектор qs. Затем найдите искомую матрицу вероятностей, используя эмпирическое распределение вектора qs.

In [39]:

```
# время выполнения 12.6176 s
import IPython
import os
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from datetime import datetime
import scipy.stats as st
from math import floor
from collections import Counter
```

In [40]:

```
%%javascript
var k = IPython.notebook.kernel;
k.execute('this_nb_name_ext = ”' + IPython.notebook.notebook_name +
 ↪  '”');
```

```
[breaklines = true]
<IPython.core.display.Javascript object>
```

In [41]:

```
# 8
# некоторые вспомогательные функии
def thisfname(): #Имя этого блокнота
    this_notebook_name = os.path.splitext(this_nb_name_ext)[0]
    return this_notebook_name

# фильтрует по дате
def date_filter(csvtab, year):
    csvtab['<DATE>'] = pd.to_datetime(csvtab['<DATE>'],
     ↪  format='%Y%m%d')
    return csvtab[(csvtab['<DATE>']>=datetime(year, 1, 1)) &
     ↪  (csvtab['<DATE>']<datetime(year + 1, 1, 1))]

# 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 range(csvtab_prob.shape[1]):
            if data.loc[i][g] < float(namecol[k]):
                stack += csvtab_prob.loc[0][namecol[k]]
        prob.append(round(stack, 1))
    return prob
```

```
# преобразлвание в latex таблицу
def _repr_latex_(self):
    return r'\scalebox{0.7}{\tabcolsep=0.11cm\centering{%s}}' %
     ↪  self.to_latex()

pd.set_option('display.notebook_repr_html', True)
pd.DataFrame._repr_latex_ = _repr_latex_  # monkey patch pandas
 ↪  DataFrame
```

In [42]:

```
V = 1 # вариант 1
n = 240 - 5*V # 235
# чтение файла с квантилями
file = 'C:/Users/timha/OneDrive/Рабочий стол/лаба/' # путь к файлам
qs = pd.read_csv('C:/Users/timha/' + 'Lab_Python_task_7.Вектор 1000
 ↪  квантилей по выборке объема n=235.csv', sep=';', decimal=',',
 ↪  engine='python', names = ['val']) # чтение файла
qs = qs.loc[1:]
tickers = ['AKRN', 'APTK', 'CHMK', 'LKOH', 'SBER'] # Тикеры
years = range(2000, 2019) # дата
zer0 = pd.DataFrame(np.zeros((len(tickers), len(years))), columns =
 ↪  years) # матрица размером len(tickers) x len(years)
for i in range(len(tickers)):
    ticDF = []
    Newtab = pd.read_csv(file + tickers[i] + '.txt', decimal='.',
     ↪  sep=',', engine='python')
    for k in range(len(years)):
        newtab = date_filter(Newtab, years[k])
        table = newtab['<CLOSE>']
        if table.shape[0] < 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(st.kurtosis(End_year, fisher=True)) # подсчёт
         ↪  эксцесса и заполнение zer0
    zer0.loc[i][:] = pd.Series(ticDF)
csvtab_prob = pd.DataFrame(Counter(qs['val']), index=[0])/qs.shape[0] #
 ↪  распределение относительных частот
# survival function
for i in range(len(tickers)):
    zer0.loc[i] = data_Sf(zer0.loc[i], csvtab_prob, i)
zer0.set_index([tickers], inplace = True)
zer0
```

Out [42]:

	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018
AKRN	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
APTK	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
CHMK	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
LKOH	0.0	0.0	0.9	0.0	0.0	0.0	0.0	0.6	0.0	0.0	0.1	0.0	0.0	0.0	0.0	0.3	0.0	0.0	0.0
SBER	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.2	0.0	0.0	0.0

In [43]:

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

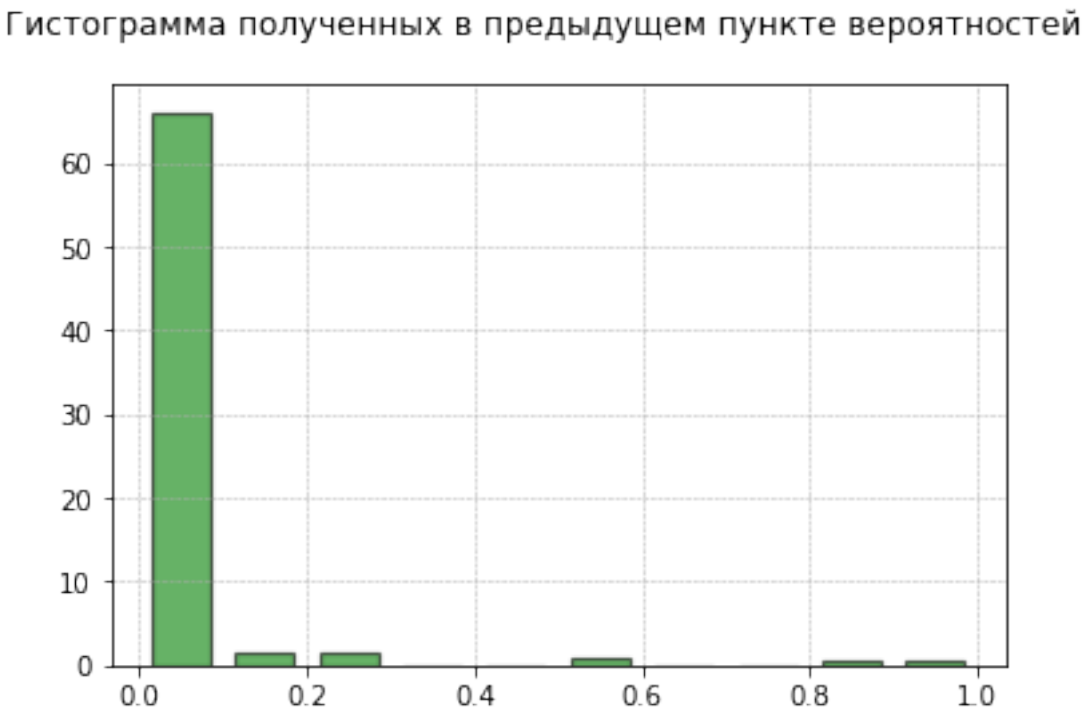
Задание 9

Гистограмма полученных в предыдущем пункте вероятностей.

In [44]:

```
# 9
# переводит таблицу в вектор без 1
def not_more1(vector):
    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])
    return one_list

# переведём таблицу в вектор без 1
vector = pd.DataFrame(not_more1(zer0.values.tolist()))
# считаем интервал
interval = np.arange(0, 1.1, 0.1)
interval[0] -= 0.1 # для включения нижней границы
interval[len(interval) - 1] += 0.1
# интрервал в одном случае попал в другом нет
new_1 = np.array(pd.cut(vector[0], interval, right =
 ↪  False).value_counts(sort = False))
new_2 = np.array(pd.cut(vector[0], interval, right =
 ↪  True).value_counts(sort = False))
new_3 = (new_1 + new_2)/2
x = np.arange(0.05, 1.05, 0.1)
# строим bar т.к. bins в hist не правильно считает кол-во попаданий
plt.bar(x, new_3, 0.07, color = 'green', alpha = 0.6,
 ↪  edgecolor='black', linewidth = 1.2, align='center')
plt.grid(linestyle='--', linewidth=0.5)
plt.suptitle('Гистограмма полученных в предыдущем пункте вероятностей')
plt.show()
```



Из данной гистограммы видно, что значения сосредоточены около нуля. Это подтверждает вывод о том, что данные эмперические распределения не имеют нормального характера.