Федеральное государственное образовательное бюджетное учреждение
высшего профессионального образования
«Финансовый университет при Правительстве Российской Федерации»

Кафедра «Теория вероятностей и математическая статистика»

Курсовая работа

на тему:

«Проверка нормальности минутной логарифмической доходности с использованием инструментария “Jupyter Notebook” и “Wolfram Mathematica”»

Вид исследуемых данных:

«Котировки акций компаний, входящих в нефтегазовый сектор России и торгующихся на Московской бирже»

Выполнил:
студент группы ПМ20-4
Козлов М. В.
Научный руководитель:
доцент, канд. физ.-мат. наук, Шамраева В. В.

Москва 2022

Содержание

Введение

Курсовая работа проверяет гипотезу о том, что логарифмическая доходность на минутных таймфреймах распределена согласно нормальному закону.

Проверка генеральной гипотезы будет осуществляться с помощью критерия согласия χ^2 (Хи-квадрат) Пирсона и критерия согласия Колмогорова для проверки равномерности распределения P-значений на отрезке [0; 1] для реальных и модельных данных соответственно. H_0 (нулевая гипотеза) состоит в том, что логарифм минутной доходности нормально распределён.

В первую очередь будет проведён анализ данных. Далее будет дана теоретическая справка о выбранных критериях. В третьей части будет проверка критерием хи-квадрат на модельных данных. В заключение проверим гипотезу на реальных данных.

В исследовании используются данные котировок компаний, входящих в нефтегазовый сектор России и торгующихся на Московской бирже. Выбраны минутные котировки за период [01.01.2021; 31.12.2021]. Предварительный анализ данных с большой вероятностью исключит некоторые акции и сократит количество рассматриваемых компаний (ввиду недостатков в выбранных данных).

Источник котировок: finam.ru

В качестве среды для проверки выбраны «Jupyter Notebook» и «Wolfram Mathematica».

Новизна работы будет заключаться в том, что используется среда «Wolfram Mathematica» и анализируются минутные котировки.

Предварительный анализ данных

Возьмём нефтегазовый сектор московской биржи. Он включает в себя множество тикеров крупных российских компаний, занимающихся добычей и обработкой нефтегазовых продуктов. Для исследования выбраны не все тикеры из списка, а только 7 наиболее крупных по капитализации.

Стоимость акций рассчитывается в российских рублях.

После выбора на основе капитализации получили следующие тикеры:

Таблица 1. Тикеры компаний

№	ТИКЕР	НАЗВАНИЕ
1	GAZP	«ГАЗПРОМ ао»
2	LKOH	«ЛУКОЙЛ»
3	ROSN	«Роснефть»
4	NVTK	«Новатэк ао»
5	TATN	«Татнфт Зао»
6	SNGS	«Сургутнфгз»
7	SIBN	«Газпрнефть»

Следующий шаг – проанализировать количество торговых дней за каждый год для каждого из тикеров, чтобы выявить аномалии, если таковые имеются. Так как в этой работе проверяются данные с минутным интервалом, то найти котировки за много лет достаточно проблематично и полученные файлы будут исчисляться сотнями тысяч строк для каждой компании. По этой причине будем рассматривать только промежуток с 01.01.2021 до 31.12.2021.

В отличие от схожих работ прошлых лет, где анализировались дневные котировки, и соответственно в таблице данные распределялись по годам, в случае минутных котировок для одного целого года будет удобно распределить все данные по двенадцати месяцам и найти общее количество торговых минут в каждом из них.

Составим такую таблицу. Для этого используется программа «Подсчёт количества торговых минут.ipynb».

Таблица 2. Количество торговых минут по месяцам

	ЯНВ	ФЕВ	МАРТ	АПР	МАЙ	ИЮНЬ	ИЮЛЬ	АВГ	СЕН	ОКТ	НОЯ	ДЕК
GAZP	15402	16211	17834	17807	16212	17832	17841	17838	11111	17028	17031	21260
LKOH	15330	16061	17721	17581	15929	17477	17399	17438	17602	16898	16941	21183
NVTK	15069	15750	17512	17355	15583	17155	17090	17352	17653	16753	16824	20823
ROSN	15197	15931	17500	17199	15504	17324	16881	17184	17799	17008	16997	21188
SIBN	9781	10333	11342	11198	10008	11283	10866	10789	11097	10726	10730	11410
SNGS	14380	14897	16580	16180	14309	15424	15904	15555	15729	15206	16362	19555
TATN	15359	16164	17782	17718	16192	17715	17714	17757	17771	16931	16943	20993

Как видно из таблицы, котировки акции SIBN («Газпрнефть») имеют серьёзный недобор торговых минут по сравнению с другими. Поэтому целесообразно исключить котировки этой акции из рассмотрения для получения более точного результата.

Помимо аномалий в количество данных, могут выявиться аномалии непосредственно в самих данных. Для этого найдём максимальные отклонения (как в положительную, так и в отрицательную сторону) для каждого месяца. Проверим это с помощью «Максимальные отклонения цен.ipynb».

Таблица 3. Максимальные относительные скачки вверх (в %)

	ЯНВ	ФЕВ	МАРТ	АПР	МАЙ	ИЮНЬ	ИЮЛЬ	АВГ	СЕН	ОКТ	НОЯ	ДЕК
GAZP	1.844	1.201	0.899	1.251	0.735	0.45	0.738	1.095	1.278	1.358	3.387	1.339
LKOH	1.211	1.688	1.181	1.206	0.764	0.729	0.656	0.934	0.95	0.883	2.127	4.098
NVTK	2.938	1.937	0.913	0.941	0.696	1.058	0.819	0.873	1.517	0.63	1.54	1.382
ROSN	2.836	0.97	0.872	1.263	0.646	0.947	1.061	0.932	2.113	1.358	2.464	0.953
SNGS	1.445	1.033	0.609	1.137	0.666	0.589	0.608	0.532	1.181	0.892	3.4	2.216
TATN	2.385	1.909	0.969	1.02	0.45	0.801	0.742	0.686	1.081	1.58	2.653	1.124

Таблица 4. Максимальные относительные скачки вниз (в %)

	ЯНВ	ФЕВ	МАРТ	АПР	МАЙ	ИЮНЬ	ИЮЛЬ	АВГ	СЕН	ОКТ	НОЯ	ДЕК
GAZP	-1.1	-0.79	-0.76	-1.39	-1.33	-0.53	-3.4	-1.91	-1.51	-1.17	-4.27	-1.55
LKOH	-1.49	-0.71	-1.11	-1.06	-0.99	-0.68	-2.46	-0.8	-0.75	-0.85	-4.11	-5.69
NVTK	-2.17	-0.68	-1.31	-0.99	-1.09	-0.63	-0.51	-1.2	-1.67	-1.09	-3.68	-1.37
ROSN	-1.88	-0.86	-1.29	-0.82	-0.57	-0.66	-1.05	-0.74	-1.33	-1.96	-3.36	-1.19

SNGS	-0.97	-0.39	-0.59	-0.68	-0.54	-0.84	-1.57	-1.38	-0.87	-0.85	-2.41	-1.61
TATN	-1.09	-0.83	-1.05	-1.61	-0.54	-0.75	-1.84	-3.02	-0.96	-1.38	-2.19	-1.33

В Таблице 3 красным цветом выделен самый крупный положительный скачок за весь период, он наблюдается в декабре у акций компании «ЛУКОЙЛ». Аналогичным образом в Таблице 4 выделен самый крупный отрицательный скачок за все двенадцать месяцев, который произошёл в декабре у «ЛУКОЙЛА».

Для большей наглядности подкрепим вычисления графиками акций этой компании за тот месяц, в котором наблюдались наибольшие скачки курса ценной бумаги. Программа, выполняющая это, представлена в «Построение графика наибольших отклонений цены.ipynb».

Рисунок 1. График цены «LKOH» за декабрь

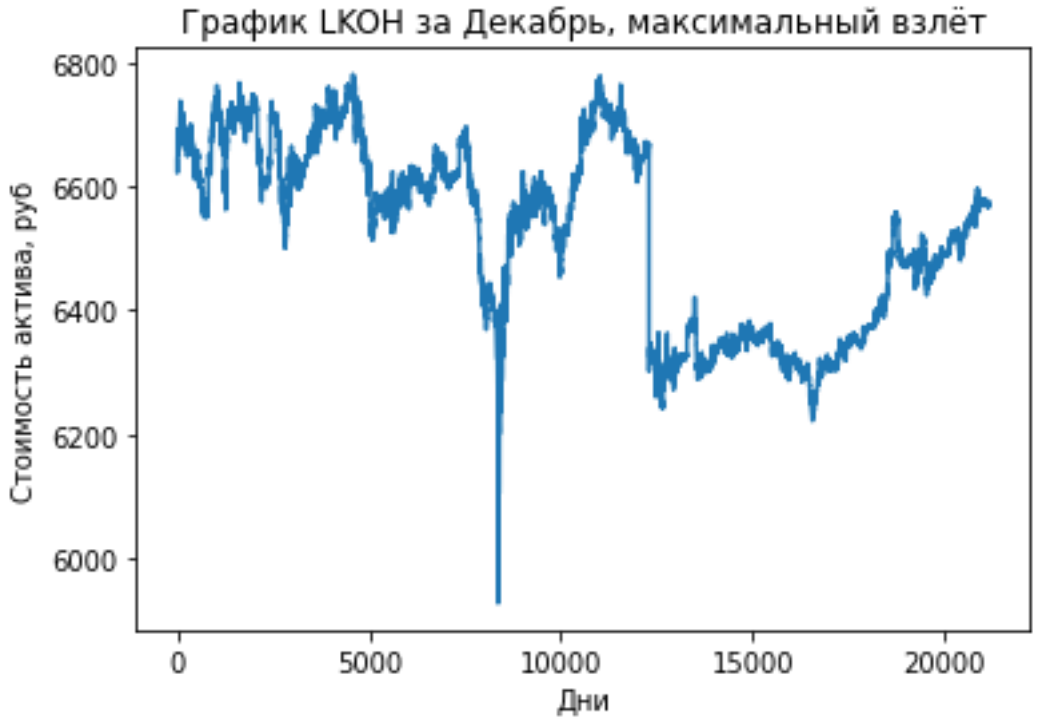
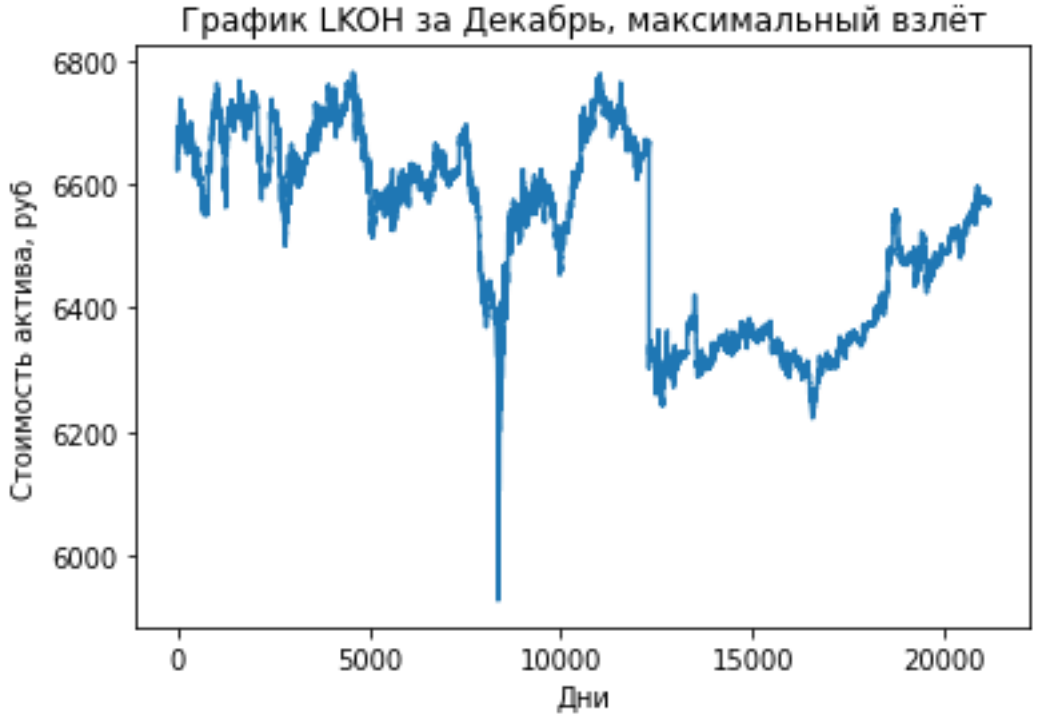


Рисунок 2. График цены «LKOH» за «декабрь»



Значения изменения курса акции вписываются в разумные пределы, поэтому из рассмотрения котировки данная акция не исключается.

Теоретическая справка по проверке гипотез

Критерий Пирсона

Разобьём нашу выборку на r групп так, чтобы i-ая группа принадлежала отрезку $(e_i - ½h; e_i + ½h)$, где $e_i = e_1 + (i - 1)h$, а $h = (x_{max}-x_{min})/m$, где $m = 1 + [\log_2 n]$, n – объём выборки. Для крайних групп (i=1, i=r) примем за интервалы разбиения: $(-\infty; e_1+½h)$ и $(e_r - ½h; +\infty)$. Тогда математическое ожидание и дисперсия будут выглядеть так:

$$m = \frac{1}{n}\sum_i \nu_i \frac{\int x g(x)\,dx}{\int g(x)\,dx}$$

$$\sigma^2 = \frac{1}{n}\sum_i \nu_i \frac{\int (x-m)^2 g(x)\,dx}{\int g(x)\,dx}$$

Найдём отсюда искомые показатели:

$$m^* = \frac{1}{n}\sum_i \nu_i \varepsilon_i$$

$$\sigma^{*2} = \frac{1}{n}\sum_i \nu_i (\varepsilon_i - m^*)^2$$

Собственно, статистика критерия Пирсона:

$$\chi^2 = \sum_i^r \frac{(\nu_i - np_i)^2}{np_i}$$

Где v_i – групповые вероятности выборки, а p_i – соответствующие значения заданной вероятностной функции. Для любой части разбиения S_i верно следующее:

$$p_i = P(S_i)$$

$$\sum_{1}^{r} p_i = 1$$

Нулевая гипотеза отвергается, если при заданном нами уровне значимости α выполняется следующее неравенство:

$$X^2 > X^2_{1-\alpha,r-m-1}$$

где r – кол-во разбиений, m – кол-во оцениваемых параметров, а X^2 – вычисленная статистика критерия.

Критерий Колмогорова-Смирнова

Вспомогательным критерием для проверки равномерности распределения p-values возьмём критерий Колмогорова-Смирнова. Он позволяет проверить согласие эмпирической функции распределения с теоретической.

$H_0: \hat{F}_n(x) = F(x)$

Статистика критерий Колмогорова определяется следующим образом: наибольший модуль разности между указанными двумя функциями распределения (теоретической и эмпирической):

$$D = \max |\hat{F}_n(x) - F(x)|.$$

Если объёмы данных не ограничены, то величина $\lambda = D\sqrt{n}$ стремится к случайной величине Q, имеющей распределение Колмогорова:

$$p(\lambda < x) \to p(\theta \leq x) = 1 + 2\sum_{k=1}^{+\infty}(-1)^{k-1} \cdot e^{2k^2x^2}, n \to \infty$$

Примем за уровень значимости α, тогда определять справедливость гипотезы H_0 будем так:

1) Если $\lambda < \lambda\alpha$, то нулевая гипотеза верна
2) Если $\lambda > \lambda\alpha$, то нулевая гипотеза не верна

Практическая часть

Гипотеза и выбранные критерии будут проверены на смоделированных методом Монте-Карло данных. Ко всему прочему проверим достоверность полученных результатов критерием Колмогорова.

В заключительном подпункте «Практической части» произведём проверку гипотезы и критерием на реальных данных, рассмотренных в Части 1 данной работы.

Проверка гипотезы на модельных данных

Проверка работоспособности наших критериев обязательно перед взаимодействием с реальными данными. Необходимо убедиться, что все программы отрабатывают идеально. Нужно проверить, что наш код будет корректно определять нормальность распределения наших модельных данных.

Программа «Генерация модельных данных.ipynb» случайным образом генерирует 1000 выборок, в каждой из которых по 16 тысяч чисел, полученных из нормального распределения. На основе полученных данных высчитываем критерий Пирсона. Объём выборок равен 16 тысячам, так как мы симулируем количество торговых минут за один месяц.

Далее мы находим значения 999 квантилей для значений критерия Пирсона.

Для примера приведём квантили от 0.1 до 0.9 с шагом 0.1

Таблица 5. Квантили

КВАНТИЛЬ	ЗНАЧЕНИЕ
0.1	9.968284

0.2	11.460526
0.3	12.685300
0.4	13.573182
0.5	14.907588
0.6	16.124294
0.7	17.575462
0.8	19.409848
0.9	22.343249

После формирования таблицы квантилей для критерия Пирсона нужно проверить равномерность, с которой p-values распределены на [0;1]. Создадим вначале эмпирический закон из полученных квантилей. Затем вычислим p-values критерия Хи-квадрат Пирсона, и проверим равномерность с помощью критерия Колмогорова-Смирнова.

Таблица 6. P-values, полученные вручную

	ЯНВ	ФЕВ	МАРТ	АПР	МАЙ	ИЮНЬ	ИЮЛЬ	АВГ	СЕН	ОКТ	НОЯ	ДЕК
GAZP	0.453	0.648	0.9	0.314	0.837	0.355	0.673	0.721	0.773	0.609	0.843	0.376
LKOH	0.507	0.193	0.172	0.302	0.446	0.839	0.416	0.732	0.534	0.776	0.995	0.078
NVTK	0.941	0.159	0.659	0.767	0.492	0.891	0.556	0.021	0.477	0.951	0.675	0.547
ROSN	0.330	0.147	0.498	0.596	0.079	0.070	0.220	0.546	0.512	0.937	0.007	0.597
SNGS	0.040	0.094	0.178	0.085	0.648	0.413	0.459	0.917	0.849	0.968	0.791	0.801
TATN	0.969	0.320	0.542	0.997	0.384	0.045	0.791	0.159	0.263	0.785	0.277	0.058

Таблица 7. P-values, полученные с помощью критерия Колмогорова

	ЯНВ	ФЕВ	МАРТ	АПР	МАЙ	ИЮНЬ	ИЮЛЬ	АВГ	СЕН	ОКТ	НОЯ	ДЕК
GAZP	0.179	0.323	0.153	0.394	0.791	0.711	0.308	0.979	0.365	0.407	0.933	0.341
LKOH	0.213	0.267	0.176	0.916	0.725	0.349	0.085	0.931	0.084	0.696	0.521	0.731
NVTK	0.825	0.732	0.679	0.013	0.407	0.184	0.446	0.154	0.891	0.694	0.698	0.648
ROSN	0.551	0.09	0.121	0.362	0.428	0.688	0.837	0.334	0.323	0.051	0.236	0.753
SNGS	0.520	0.136	0.338	0.207	0.184	0.786	0.568	0.35	0.096	0.859	0.839	0.033
TATN	0.902	0.354	0.111	0.295	0.825	0.076	0.520	0.555	0.063	0.151	0.571	0.123

Рисунок 3. Гистограмма 1

Рисунок 4. Гистограмма 2

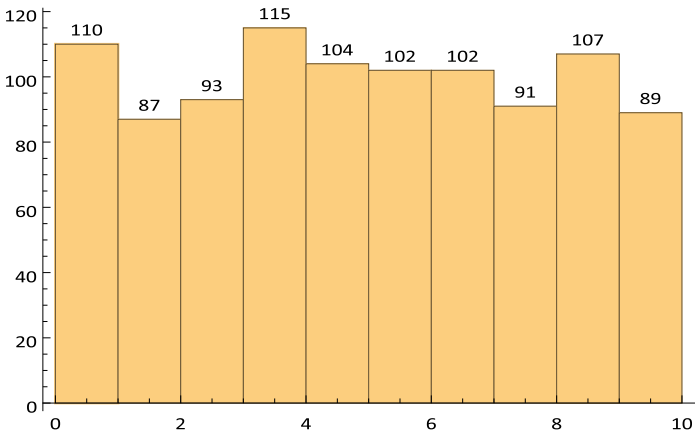
По гистограммам видно, что p-values, полученные обоими способами, распределены равномерно на отрезке от 0 до 1. P-value критерия Колмогорова-

Смирнова равняется 0.372. Как и предполагалось, на модельных данных гипотеза о нормальном распределении смоделированной выборки.

Для большей надёжности проверим равномерность распределения p-values через Wolfram Mathematica.

Также сгенерируем 1000 выборок по 16 тысяч элементов в каждой, найдём для каждой p-value и отрисуем гистограмму. Получим следующий результат:

Рисунок 5. Гистограмма через Wolfram Mathematica



Как видим, равномерность подтверждается и с помощью вычислений, полученных в среде Wolfram Mathematica. Код программы находится в «Генерация модельных данных.nb».

Проверка гипотезы на реальных данных

Приступим к главной части работы – проверке на реальных данных. На смоделированных данных гипотеза подтверждается, теперь выясним, какой результат мы получим на реальных, отобранных нами данных. Для этого мною была написана программа «Проверка гипотезы для реальных данных.ipynb». По завершении работы программы, получаем таблицу p-values для каждой компании

для каждого месяца. Аналогично пункту 3.1 строится гистограмма распределения p-values на отрезке [0;1].

Таблица 8. P-values для реальных данных

	ЯНВ	ФЕВ	МАРТ	АПР	МАЙ	ИЮНЬ	ИЮЛЬ	АВГ	СЕН	ОКТ	НОЯ	ДЕК
GAZP	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
LKOH	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
NVTK	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
ROSN	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
SNGS	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
TATN	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Изобразим гистограмму распределения p-values.

Рисунок 6. Гистограмма для реальных данных

Анализируя гистограмму явно видно, что распределение p-values не является равномерным и сосредоточенно в одном месте. Также о неравномерности говорит и p-value критерия Колмогорова, которое равняется 7.681e-29.

Заключение

В итоге можно сказать, что проверка, проделанная в курсовой работе с использованием среды «Jupyter Notebook» и «Wolfram Mathematica», а также критерия Хи-квадрат Пирсона и Колмогорова-Смирнова, не подтвердила исходную гипотезу о нормальном распределении логарифмической доходности на минутных котировках. Подобный результат наблюдается и в работах прошлых лет, рассматривающих котировки с интервалами большей длины.

Новизны в данную работу добавляют следующие пункты:

1) Работа с Wolfram Mathematica
2) Рассмотрение минутных котировок

В связи с этими требованиями мною не только дорабатывались программы из работ прошлых лет, но и создавались полностью новые.

Литература

1. Г. Крамер «Математические методы статистики» // Издательство «Мир». 1975.
2. Г. И. Ивченко, Ю. И. Медведев «Введение в математическую статистику» // Издательство «ЛКИ». 2009.
3. Громова М. С. «Проверка гипотезы о нормальном распределении логарифмической доходности по критерию Дэвида-Хартли-Пирсона». Вид исследуемых данных: «Котировки акций компаний, входящих в индекс ММВБ нефти и газа» - М., 2019
4. Гераськина Н. С. «Проверка гипотезы о нормальном распределении дневной логарифмической доходности при условии определённого объёма торгов накануне». Вид исследуемых данных: «Котировки акций компаний, входящих в индекс S&P 100» - М., 2021
5. Гмурман В. Е. Теория вероятностей и математическая статистика: Учеб. пособие для вузов / В. Е. Гмурман. — 9-е изд., стер. — М.: Высш. шк., 2003.
6. Браилов А. В. Лекции по математической статистике. — М.: Финакадемия, 2007.
7. Состав нефтегазового сектора: https://smart-lab.ru/q/shares/?sector_id%5B%5D=1
8. URL: https://statanaliz.info/statistica/proverka-gipotez/kriterij-soglasiya-pirsona-khi-kvadrat/
9. URL: https://habr.com/ru/company/skillfactory/blog/510688/

Приложение

Приложение 1 – характеристики компьютера

Процессор: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz

Random Access Memory (RAM): 8 GB

Название программы	Время выполнения, сек
Подсчёт количества торговых минут.ipynb	7
Максимальные отклонения цен.ipynb	16
Построение графика наибольших отклонений цен.ipynb	3
Генерация модельных данных.ipynb	148
Генерация модельных данных.nb	6
Проверка гипотезы для реальных данных.ipynb	20

Приложение 2 – код программ

Подсчёт количества торговых минут.ipynb

```
directory = 'C:/Users/Mark/Desktop/Данные/'
files = os.listdir(directory)

# ['GAZP', 'SIBN', ...]
tickers = [i.split('.')[0] for i in files if i[-3:] == 'csv']

# {'GAZP': 0, 'SIBN': 1, 'LKOH': 2, ...}
tickers = {ticker: index for index, ticker in enumerate(tickers)}

months = ['Январь', 'Февраль', 'Март', 'Апрель', 'Май', 'Июнь',
          'Июль', 'Август', 'Сентябрь', 'Октябрь', 'Ноябрь', 'Декабрь']

months_dict = {'Январь': 0, 'Февраль': 1, 'Март': 2, 'Апрель': 3,
               'Май': 4, 'Июнь': 5, 'Июль': 6, 'Август': 7, 'Сентябрь': 8,
               'Октябрь': 9, 'Ноябрь': 10, 'Декабрь': 11}

# порядковые номера месяцев
month_numbers = ['01', '02', '03', '04', '05', '06',
                 '07', '08', '09', '10', '11', '12']

# {'01': 0, '02': 1, '03': 2, ...}
month_numbers_dict = {name: i for i, name in enumerate(month_numbers)}
# датафрейм для количества торговых минут
data_df = pd.DataFrame(columns=months, index=tickers.keys())
```

```
for company in tickers.keys():
    # проходимся в цикле по всем компаниям
    with open(directory + f'{company}.csv') as file:
        df = pd.read_csv(file, delimiter=';')

        # количество минут для каждого месяца
        counts = [0]*12

        for i in month_numbers:
            for date in df['<DATE>']:
                if date[3:5] == i:
                    # если месяц совпал, увеличиваем счётчик
                    counts[month_numbers_dict[i]] += 1
                elif i != '12' and date[3:5] == month_numbers[month_numbers_di
ct[i]+1]:
                    # если попали на след месяц, выходим из цикла
                    break

        for i in range(len(counts)):
            # заполняем датафрейм
            data_df[months[i]][company] = counts[i]

# после анализа датафрейма удаляем лишние компании
data_df = data_df.drop(['SIBN'])
del tickers['SIBN']
```

Максимальные отклонения цен.ipynb

```
def calculate_diff_log(array):
    """
    Функция для расчёта логарифмической доходности
    Принимает на вход массив значений цен закрытия

    """
    diff = []
    for i in range(1, len(array)):
        diff.append(math.log(array[i] / array[i-1])*100)

    return diff
# датафрейм для макс скачков вверх
up_df = pd.DataFrame(columns=months, index=tickers.keys())
# датафрейм для макс скачков вверх
down_df = pd.DataFrame(columns=months, index=tickers.keys())

for company in tickers.keys():
    with open('C:/Users/Mark/Desktop/Данные/'+f'{company}.csv') as file:
        mylist = []
        # так как файл с данными очень велик, будем считывать
        # с помощью особого метода
        for chunk in pd.read_csv(file, sep=';', chunksize=20_000):
            mylist.append(chunk)
        df = pd.concat(mylist, axis= 0)
        del mylist

        # макс взлеты и падения для каждого месяца
        diff_up = [0]*12
        diff_down = [0]*12

        for i in month_numbers:
            count = -1
            value_array = []    # считываем сюда цены закрытия
```

```
            for date in df['<DATE>']:
                count += 1
                if date[3:5] == i:
                    value_array.append(df['<CLOSE>'][count])
                elif i != '12' and date[3:5] == month_numbers[month_numbers_di
ct[i]+1]:
                    break

            # вызываем функцию расчета лог доходности
            now_diff = calculate_diff_log(array=value_array)

            # определяем макс и мин значения
            diff_up[month_numbers_dict[i]] = max(now_diff)
            diff_down[month_numbers_dict[i]] = min(now_diff)


        for i in range(len(diff_up)):
            # заполняем наши датафреймы полученными данными
            up_df[months[i]][company] = round(diff_up[i], 3)
            down_df[months[i]][company] = round(diff_down[i], 2)
```

Построение графика наибольших отклонений цен.ipynb

```
def get_max_diff(data_frame_up, data_frame_down):
    """
    Возвращает месяц и компанию с самым крупным дневным взлётом
    На вход принимает параметр data_frame_up - датафрейм из pandas с положител
ьными скачками
    И data_frame_down - датафрейм из pandas с отрицательными скачками

    """
    maximum = -10**5
    minimum = 10**5

    month_max, month_min = 0, 0
    company_max, company_min = 0, 0

    # двойной цикл, перебираем каждый месяц и каждую компанию
    for month in months:
        for company in tickers.keys():
            if data_frame_up[month][company] > maximum:
                # проверка на максимум
                maximum = data_frame_up[month][company]
                month_max = month
                company_max = company
            if data_frame_down[month][company] < minimum:
                # проверка на минимум
                minimum = data_frame_down[month][company]
                month_min = month
                company_min = company

    return month_max, company_max, month_min, company_min

# сохраняем найденные значения в переменные
month_max, company_max, month_min, company_min = get_max_diff(data_frame_up=up
_df, data_frame_down=down_df)
def draw_maximum(mon_max, com_max, mon_min, com_min):
    """
    Функцию для отрисовки графика максимального взлёта и падения цены.
    Получает на вход четыре параметра:
    mon_max - месяц с макс взлетом
    com_max - компания с макс взлетом
```

```
    mon_min - месяц с макс падением
    com_min -компания с макс падением

    """
    com = [com_max, com_min]
    mon = [mon_max, mon_min]
    titles = ['максимальный взлёт', 'максимальное падение']

    for j, company in enumerate(com):
        close_value_array = []

        with open('C:/Users/Mark/Desktop/Данные/' + f'{company}.csv') as file:
            mylist = []
            for chunk in pd.read_csv(file, sep=';', chunksize=20_000):
                mylist.append(chunk)
            df = pd.concat(mylist, axis= 0)
            del mylist

            count = -1
            i = months_dict[mon[j]]
            number = month_numbers[i]

            for date in df['<DATE>']:
                count += 1
                if date[3:5] == number:
                    close_value_array.append(df['<CLOSE>'][count])

                elif number != '12' and date[3:5] == month_numbers[i+1]:
                    break

        close_value_array = np.array(close_value_array)

        # x и y координаты для графика
        x_coord = np.arange(0, len(close_value_array), 1)
        y_coord = close_value_array

        # рисуем график с помощью plt
        fig, ax = plt.subplots()
        ax.plot(x_coord, y_coord)
        ax.set_title(f'График {company} за {mon[j]}, {titles[j]}')
        ax.set_xlabel('Дни')
        ax.set_ylabel('Стоимость актива, руб')

        plt.show()


draw_maximum(mon_max=month_max, com_max=company_max, mon_min=month_min, com_mi
n=company_min)
```

Генерация модельных данных.ipynb

```
def random_tetta(data):
    """
    Функция оценивает такие параметры рандомной выборки как: мат ожидание и ди
сперсия

    """
    data = sorted(data)
    n = len(data)
    r = 1 + int(math.log2(n))    # находим кол-во интервалов разбиения
    max_el = max(data)
    min_el = min(data)
```

```
    h = (max_el-min_el) / r    # шаг

    middle_points = [min_el]    # храним середины интервалов
    intervals = []    # разбиение data на интервалы
    intervals.append([el for el in data if el < middle_points[-1]+h/2])

    for i in range(2, r):
        middle_points.append(middle_points[0]+(i-1)*h)
        intervals.append([el for el in data if middle_points[-1]-0.5*h < el <
middle_points[-1]+0.5*h])

    middle_points.append(middle_points[0]+(r-1)*h)
    intervals.append([el for el in data if el > middle_points[-1]-0.5*h])

    interval_len = [len(interval) for interval in intervals]

    m = []
    # находим мат ожидание по формуле
    for i in range(len(interval_len)):
        m.append(interval_len[i]*middle_points[i])
    m_ = (1/n) * sum(m)

    s = []
    # находим дисперсию по формуле
    for i in range(len(interval_len)):
        s.append(interval_len[i]*(middle_points[i]-m_)**2)
    sigma_ = (1/n) *sum(s)

    return m_, sigma_, interval_len, middle_points
def pirsons(n=16000):
    """
    Статистика критерия Пирсона для выборки размера n

    """
    data = scipy.stats.norm(0, 1).rvs(n)    # генерируем выборку

    r = 1 + int(math.log2(n))    # кол-во интервалов
    min_el = min(data)
    max_el = max(data)
    h = (max_el-min_el) / r    # шаг

    function = random_tetta(data)  # находим параметры выборки

    m, sigma, interval_len, middle_points = function
    expected = scipy.stats.norm(m, math.sqrt(sigma))  # ожидаемое распределени
e

    prob = [expected.cdf(middle_points[0]+0.5*h)]  # вероятности попасть в инт
ервал
    prob += [expected.cdf(middle_points[x]+0.5*h)-expected.cdf(middle_points[x
]-0.5*h) for x in range(1, len(middle_points)-1)]
    prob += [1-expected.cdf(middle_points[-1]-0.5*h)]

    z = []
    # находим хи-квадрат Пирсона
    for i in range(len(prob)):
        z.append( ((interval_len[i]-n*prob[i])**2)/(n*prob[i]) )

    chi_2 = sum(z)

    return chi_2
N = 1000    # количество испытаний
n = 16000    # размер выборки
```

```
pirs_stat = [pirsons(n) for _ in range(N)]    # 1000 значений хи-квадрат

# квантили для статистики Пирсона
chi2_999 = np.quantile(pirs_stat, np.arange(0.001, 1, 0.001))
chi2_9 = np.quantile(pirs_stat, np.arange(0.1, 1, 0.1))

df_quant_999 = pd.DataFrame()
df_quant_9 = pd.DataFrame()

# заполняем датафреймы для квантилей
df_quant_999['Месяц'] = np.round(chi2_999, 6)
df_quant_9['Месяц'] = np.round(chi2_9, 6)

index_999 = [quantile for quantile in list(np.arange(0.001, 1, 0.001))]
df_quant_999.index = index_999

index_9 = [quantile for quantile in list(np.arange(0.1, 1, 0.1))]
df_quant_9.index = index_9

df_quant_9
n_years = 12
n_tickers = len(tickers)
n_vb = 16000


def pvaluesVR(n, q):
    """
    Расчёт p-values вручную

    """
    u0 = pirsons(n)  # статистика Пирсона
    k = 0
    for i in range(len(q)): # расчёт P-value
        if q[i] > u0:
            k += 1
    p = round(k/len(q), 3)
    return p


pvalues_data_VR = pd.DataFrame()
pvalues_list_VR =[]

# двойной цикл, расчитываем p-values
for i in range(n_tickers):
    probs = []
    for j in range(n_years):
        probs.append(pvaluesVR(n_vb, chi2_999))
        pvalues_list_VR.append(int(probs[j]*10))
    probs = pd.DataFrame(probs).transpose()
    pvalues_data_VR = pvalues_data_VR.append(probs, ignore_index=True)

pvalues_data_VR.columns = months
pvalues_data_VR.index = tickers.keys()

VR_interval = [pvalues_list_VR.count(i) for i in range(11)]  # для гистограммы

pvalues_data_VR
def pvaluesKS(n):
    """
    p-values с помощью критерия Колмогорова-Смирнова

    """
```

```
    x = np.random.normal(loc=0, scale=1, size=n) # выборка
    p = round(scipy.stats.kstest(x,'norm')[1], 3) # p-value
    return p

pvalues_data_KS = pd.DataFrame()
pvalues_list_KS =[]

for i in range(n_tickers):
    probs = []
    for j in range(n_years):
        probs.append(pvaluesKS(n_vb))
        pvalues_list_KS.append(int(probs[j]*10))
    probs = pd.DataFrame(probs).transpose()
    pvalues_data_KS = pvalues_data_KS.append(probs, ignore_index=True)

pvalues_data_KS.columns = months
pvalues_data_KS.index = tickers.keys()

KS_interval = [pvalues_list_KS.count(i) for i in range(11)] # для гистограммы

pvalue = scipy.stats.ks_2samp(pvalues_list_VR, pvalues_list_KS) # значение кри
терия Колмогорова
print(pvalue)

# рисуем первую гистограмму - для значений, вычисленных вручную
plt.hist(pvalues_list_VR, bins=[_ for _ in range(0, 11)], color='#0504aa', alp
ha=0.7, rwidth=0.85)
plt.title('Гистограмма p-values, вычисленных вручную')
plt.xticks([_ for _ in range(0, 11)], [digit/10 for digit in range(0, 11)])
plt.show()

# рисуем вторую гистограмму - для значений, вычисленных критерием Колмогорова
plt.hist(pvalues_list_KS, bins=[_ for _ in range(0, 11)], color='#0504aa', alp
ha=0.7, rwidth=0.85)
plt.title('Гистограмма p-values, вычисленных с критерием Колмогорова')
plt.xticks([_ for _ in range(0, 11)], [digit/10 for digit in range(0, 11)])
plt.show()

pvalues_data_KS
```

Генерация модельных данных.nb

```
pointlist=Table[PearsonChiSquareTest[ RandomVariate[NormalDistribution[0, 1],
16000],Automatic, "PValue"],{i,1,1000}];
pointlist = pointlist*10;
a = Table[Count[pointlist, x_ /; i <= x < (i+1)], {i,0,9}];
Print[a];
Print[Histogram[pointlist, {1}, LabelingFunction -> Above]];
```

Проверка гипотезы для реальных данных.ipynb

```
def pirsons_real(array):
    """
    Статистика критерия Пирсона для выборки размера n
    На вход подаётся выборка

    """
    n = len(array)  # длина выборки
    data = array
```

```
    r = 1 + int(math.log2(n)) # интервалы
    min_el = min(data)
    max_el = max(data)
    h = (max_el-min_el) / r # шаг

    function = random_tetta(data)

    m, sigma, interval_len, middle_points = function # параметры выборки
    expected = scipy.stats.norm(m, math.sqrt(sigma))    # ожидаемое распределе
ние

    prob = [expected.cdf(middle_points[0]+0.5*h)]  # вероятности попадания в и
нтервал
    prob += [expected.cdf(middle_points[x]+0.5*h)-expected.cdf(middle_points[x
]-0.5*h) for x in range(1, len(middle_points)-1)]
    prob += [1-expected.cdf(middle_points[-1]-0.5*h)]

    z = []
    # расчёт хи-квадрат
    for i in range(len(prob)):
        if n*prob[i] == 0:
            z.append(0)
        else:
            z.append( ((interval_len[i]-n*prob[i])**2)/(n*prob[i]) )

    chi_2 = sum(z)

    return chi_2
def pvalue_real(x):
    """
    Нахождение значения p-value для месяца реальных данных

    """
    u0 = pirsons_real(array=x)
    k = 0
    for i in range(len(x)):
        if x[i] > u0:
            k += 1
    p = round(k/len(x), 10) # p-value
    return p


pvalue_data = pd.DataFrame(columns=months, index=tickers.keys())
all_pvalues = []

for company in tickers.keys():
    with open('C:/Users/Mark/Desktop/Данные/' + f'{company}.csv') as file:
        mylist = []
        for chunk in pd.read_csv(file, sep=';', chunksize=20_000):
            mylist.append(chunk)
        df = pd.concat(mylist, axis= 0)
        del mylist

        pvalue_list = []

        for i in month_numbers:
            count = -1
            value_array = []

            for date in df['<DATE>']:
                count += 1
                if date[3:5] == i:
```

```
                    value_array.append(df['<CLOSE>'][count])
                elif i != '12' and date[3:5] == month_numbers[month_numbers_di
ct[i]+1]:
                    break

            now_diff = calculate_diff_log(array=value_array)
            pvalue_list.append(pvalue_real(now_diff))

        all_pvalues.extend([i*10 for i in pvalue_list])

        for i in range(len(months)):
            pvalue_data[months[i]][company] = pvalue_list[i]


# кол-во значений в каждом интервале
p_interval = [all_pvalues.count(i) for i in range(11)]

# гистограмма на реальных данных
plt.hist(all_pvalues, bins=[i for i in range(11)],color='#0504aa', alpha=0.7,
rwidth=0.85)
plt.title('Гистограмма p-значений для реальных данных')
plt.xticks([i for i in range(11)], [i/10 for i in range(11)])
plt.show()


pvalue = scipy.stats.ks_2samp(all_pvalues, pvalues_list_KS)
print(pvalue)

pv001 = 0
pv005 = 0
pv010 = 0

for month in months:
    for ticker in tickers.keys():
        if pvalue_data[month][ticker] > 0.01:
            pv001 += 1
        if pvalue_data[month][ticker] > 0.05:
            pv005 += 1
        if pvalue_data[month][ticker] > 0.1:
            pv010 += 1

pv001 = round(pv001/len(tickers)/len(months), 5)
pv005 = round(pv005/len(tickers)/len(months), 5)
pv010 = round(pv010/len(tickers)/len(months), 5)

print('1% уровень значимости:', pv001)
print('5% уровень значимости:', pv005)
print('10% уровень значимости:', pv010)
```

Приложение 3 – список файлов

Название файла	**Программа**
Рисунок 1. График цены «LKOH» за декабрь	Построение графика наибольших отклонений цен.ipynb
Рисунок 2. График цены «LKOH» за декабрь	Построение графика наибольших отклонений цен.ipynb

Таблица 1. Тикеры компаний	Подсчёт количества торговых минут.ipynb
Таблица 2. Количество торговых минут по месяцам	Подсчёт количества торговых минут.ipynb
Таблица 3. Максимальные относительные скачки вверх (в %)	Максимальные отклонения цен.ipynb
Таблица 4. Максимальные относительные скачки вниз (в %)	Максимальные отклонения цен.ipynb
Таблица 5. Квантили	Генерация модельных данных.ipynb
Таблица 6. P-values, полученные вручную	Генерация модельных данных.ipynb
Таблица 7. P-values, полученные с помощью критерия Колмогорова	Генерация модельных данных.ipynb
Рисунок 3. Гистограмма 1	Генерация модельных данных.ipynb
Рисунок 4. Гистограмма 2	Генерация модельных данных.ipynb
Рисунок 5. Гистограмма через Wolfram Mathematica	Генерация модельных данных.nb
Рисунок 6. Гистограмма для реальных данных	Проверка гипотезы для реальных данных.ipynb
Таблица 8. P-values для реальных данных	Проверка гипотезы для реальных данных.ipynb