Федеральное государственное образовательное бюджетное учреждение высшего образования
«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»
(Финансовый университет)
ДЕПАРТАМЕНТ АНАЛИЗА ДАННЫХ И МАШИННОГО ОБУЧЕНИЯ

Факультет информационных технологий и анализа больших данных
Дисциплина: «**Теория вероятностей и математическая статистика**»
Направление подготовки: 01.03.02 «Прикладная математика и информатика»
профиль «Анализ данных и принятие решений в экономике и финансах»
Форма обучения очная
Учебный 2021/2022 год, 4 семестр

Курсовая работа на тему:

«Проверка гипотезы о нормальном распределении логарифмической доходности по критерию Шапиро-Уилка»

Вид исследуемых данных:

Котировки акций, входящих в немецкий индекс DAX

Выполнила:
студентка группы
ПМ20-1
Усенко Ксения
Сергеевна

Научный руководитель:
д. ф.-м. н., доц., проф,
Рябов Павел
Евгеньевич

Москва 2022г.

Оглавление

Раздел 1. Введение

1.1 Объяснение выбранной темы

Чтобы выполнить заданную курсовую работу, потребуется проверить гипотезу о нормальном распределении логарифмической доходности. В качестве статистического критерия используется критерий Шапиро-Уилка. Дополняющим критерием выбранн критерий Колмогорова, он применим в работе для проверки равномерности распределения P-значения выше описанного статистического критерия. С помощью Монте-Карло оценивается мощность рассматриваемых критериев.

1.2 Описание выборок – исследуемых данных

В моей курсовой используются 9 котировок компаний, которые входят в индекс DAX. Перечислю эти компании со следующими тикерами: BAYN, DTE, SAP, DPW, LIN, ALV, SIE, BAS и BMW. Исследовать данные было решено в период с 1 января 2016 года до 31 декабря 2020 года. Котировки экспортировались с ресурса https://www.finam.ru. В ходе изучения данных для проверки гипотезы придётся убрать часть данных, которые могли бы исказить результаты.

1.3 Планируемая новизна

Новизна моей работы заключается в том, что индекс DAX по данным с сайта Matcalc не анализировался с 2016 года. Более того, в отличие от предыдущих работ, связанных с данным немецким фондом, для анализа и визуализации используется язык Python.

Раздел 2. Предварительный анализ данных

Для анализа взяты 9 “голубых фишек” – самых надёжных компаний из списка немецкого индекса DAX (последнее обновление 5 апреля 2022) с источника https://dailypik.com/ger30-index-companies/

Названия компаний и тикеров приведены в таблице ниже.

Название компании	**Тикер**
Bayer	BAYN
Deutsche Telekom	DTE
SAP	SAP
Deutsche Post	DPW
Linde	LIN
Allianz	ALV
Siemens	SIE
BASF	BAS
BMW	BMW

Таблица 1. Отобранные компании

Для начала рассчитаем количество торговых дней у каждой компании по годам. Это делается для того, чтобы понять, подходят ли данные для анализа, и если нет – что надо исключить.

Тикер	2016	2017	2018	2019	2020
ALV	255	252	239	249	253
BAS	255	252	239	249	253
BAYN	255	252	239	249	253
BMW	255	252	249	249	253
DPW	255	252	241	249	252
DTE	255	252	241	249	252
LIN	0	0	0	0	90
SAP	255	252	241	249	252
SIE	255	252	241	249	252

Таблица 2. Число торговых дней.

С помощью визуализации этой таблицы мы видим, что у компании Linde котировки появились только с 2020 года, поэтому в дальнейшей исследовании мы не будем ее использовать – исключаем из выборки.

Теперь займёмся расчётом максимальных дневных относительных скачков цен вверх и вниз, потому что только количества торговых дней будет недостаточно для вывода по подходящим компаниям для дальнейшего анализа. Приведённые вычисления описаны в процентах, используемые поля в данных о компаниях – <TICKER>,<DATE>,<CLOSE>.

Ticker	2016	2017	2018	2019	2020
ALV	4,37%	3,31%	3,12%	3,39%	13,51%
BAS	3,62%	3,88%	3,94%	4,91%	9,56%
BAYN	5,16%	4,21%	5,22%	9,05%	7,77%
BMW	4,74%	2,99%	4,85%	4,07%	12,54%
DPW	5,21%	4,87%	4,31%	3,96%	11,08%
DTE	4,27%	4,60%	3,47%	2,77%	7,27%
SAP	5,68%	2,95%	5,02%	11,97%	8,04%
SIE	8,62%	5,63%	6,53%	5,08%	9,97%

Таблица 3. Максимальные дневные относительных скачков цен вверх

Ticker	2016	2017	2018	2019	2020
ALV	-10,39%	-2,20%	-4,97%	-5,52%	-14,23%
BAS	-6,68%	-2,68%	-4,42%	-5,33%	-10,59%
BAYN	-8,20%	-4,10%	-10,70%	-9,93%	-12,15%
BMW	-7,53%	-3,20%	-5,60%	-5,23%	-12,08%
DPW	-7,01%	-3,84%	-7,06%	-3,74%	-11,44%
DTE	-5,19%	-3,22%	-5,12%	-4,70%	-9,28%
SAP	-5,63%	-3,46%	-6,08%	-5,37%	-22,37%
SIE	-7,41%	-3,73%	-4,94%	-4,12%	-11,31%

Таблица 4. Максимальные дневные относительных скачков цен вниз

По таблице видно, что ни одна из рассматриваемых компаний не имеет в заданный период максимальное отклонение цен вниз и вверх> 50%, поэтому нет повода для исключения из выборки каких-то компаний. Однако можно рассмотреть графики двух компаний с максимальных отклонением вверх и вниз.

Ниже изображен график цен для акций с максимальным однодневным *снижением цены* – компания SAP и график цен для акций с максимальным однодневным *повышением цены* – компания ALV. Используемые поля для графиков в коде – <DATE>,<CLOSE>.

Рис 1. График цен закрытия компании SAP

Рис 1. График цен закрытия компании SAP

Рис 2. График цен закрытия компании ALV

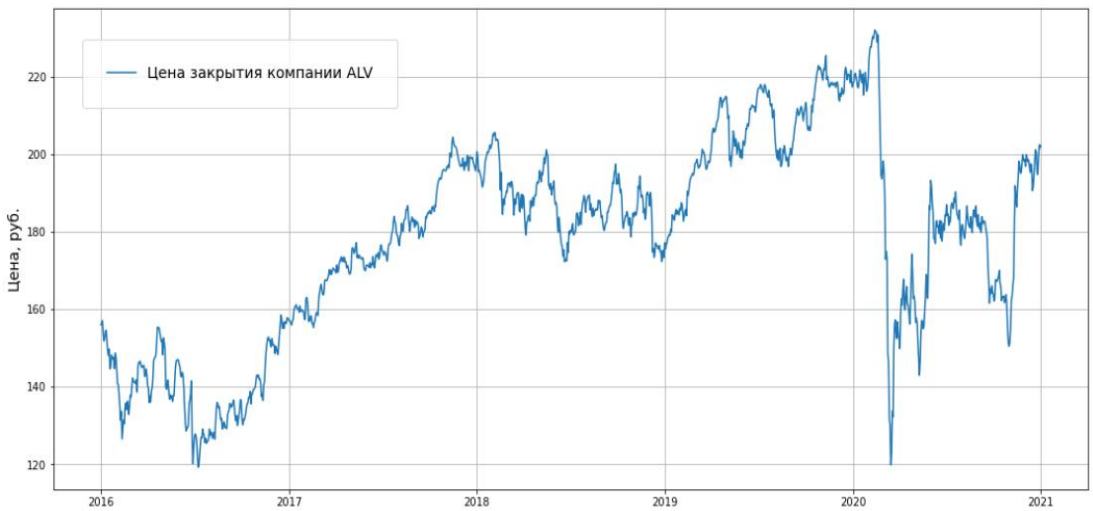


Рис 2. График цен закрытия компании ALV

На данном шаге завершается предварительный анализ, неподходящие для дальнейшего анализа компании удалены из выборки. У оставшихся компаний также можно удалить столбцы PER, TIME, так как они не несут смысловой нагрузки. Теперь можно приступать к теоретической справке.

Раздел 3. Теоретическая справка

1) Что такое статистическая гипотеза

Прежде чем осуществлять проверку гипотезы о нормальном распределении, нужно разобраться, что же такое статистическая гипотеза?

Статистическая гипотеза объясняется любым предположением о виде распределения исследуемых данных. Существуют две основные гипотезы, с которыми мы будем работать – нулевая и альтернативная. Нулевая гипотеза H0 используется для подтверждения нашего предположения, в которое мы верим. Однако, мы заранее должны продумать, во что верить, если нулевая гипотеза не подтвердится проверкой данных. Для этого и существует альтернативная гипотеза H1. H0 и H1 взаимоисключают друг друга.

2) Статистический критерий

Определим понятие статистического критерия. Статистический критерий – это такое строгое правило в математической статистике, по которому принимается либо же отвергается рассматриваемая гипотеза при заранее обозначенном уровне значимости. Большая часть статистических критериев указывается через подходящее множество и функцию t(x1,.., xn).

Основываясь на статистическом критерии, установление решений в пользу нулевой или альтернативной гипотезы может привести к появлению ошибок 1-го и 2-го рода. Ошибка 1-го рода выражается в том, что H0 – верная гипотеза и она отвергается. Аналогично ошибка 2-го рода выражается в том, что верная гипотеза H1 отвергается. Определим вероятность возникновения ошибки первого рода – α, второго рода – β.

Мы рассмотрим такие критерии, где α и β не зависят от распределения. Тогда вместо ошибки 1-го рода мы будем использовать коэффициент значимости, а при ошибке 2-го рода вводится понятие мощности критерия 1-β, которое показывает: чем выше мощность – тем менее вероятно, что верная гипотеза H1 отвергается.

Конечно, нам бы хотелось минимизировать одновременно как α, так и β. Однако это неосуществимо, так как при уменьшении α, β увеличивается. Поэтому мы определим небольшое значение α и выберем более мощный критерий.

3) P–value.

P-value рассматривается вместе с уровнем значимости α. Данная величина характеризует вероятность ошибки при отклонении H0 – нулевой гипотезы. P-value – такое минимальное значение уровня значимости, которое принадлежит критической области, способное отвергнуть гипотезу H0. Таким образом, если p-value меньше заданного уровня значимости – мы отклоняем нулевую гипотезу, так как если p-значение уже способно отвергнуть гипотезу и α больше него, то и α также отвергнет H0. Если p-value >= α – H0 верна.

4) Критерий Шапиро-Уилка

Данный критерий интересен для нас потому, что он эффективно проверяет нормальность на совокупности выборок, которые независимы между собой. Статистика критерия для вариационного ряда $x_1 \le x_2 \le \cdots \le x_n$ выборки выражается так:

$$W = \frac{1}{\sum_{i=1}^{n}(x_i-\bar{x})^2} * \left[\sum_{i=1}^{n/2} a_{n-i+1}(x_{n-i+1} - x_i)\right]^2,$$

где $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$.

a_{n-i+1} – это коэффициент и W(α) – критические значения статистики Шапиро-Уилка, которые вычисляются по изначально заданной таблице. Если при описанном уровне значимости будет W < W(α), то мы отвергаем H0 – нулевую гипотезу. Чтобы получить эмпирическое значение статистики при H0, найдём приближенную вероятность с помощью формулы, данной ниже. Коэффициенты ε, η и γ определяются по таблице.

$$z = \gamma + \eta \ln\left(\frac{W - \varepsilon}{1 - W}\right)$$

На самом деле, этот статистический критерий Шапиро-Уилка, появившийся в 1965 году, верно работает исключительно для выборок, у которых объем n⩽50. Поэтому вскоре он был модернизирован, так и появился критерий Шапиро-Франчиа. Далее Ж.П.Ройстоном в 1982 году был выдвинут более реалистичный способ подсчета P-значений, потому что он работал для выборок объема n⩽2000 выборок:

$$y = (1 - W)^{\lambda}, z = \frac{y - \mu_y}{\sigma_y},$$

z выступает нормальной случайной величиной, μ_y – её математическое ожидание, σ_y – её среднеквадратическое отклонение.

5) Критерий согласия

Используется критерий согласия Колмогорова для проверки того, правда ли распределение выборки X описывается выбранным нами распределением P0. В нашем случае он выступает в качестве вспомогательного критерия, чтобы проверить полученные P-значения на равномерность. Так выглядит функция статистики этого критерия:

$d = \sup_x \left|\hat{F}(x) - F(x)\right|$, где $\hat{F}(x)$ – эмпирическая функция распределения, а $F(x)$ – теоретическая. В чем идея? Если нулевая гипотеза верна – $d \to 0$, иначе – $d > 0$. Плюс критерия заключается в том, что что d не зависит от конкретного непрерывного распределения P0.

Колмогоров обнаружил, что если взять $\lambda = d\sqrt{n}$ и описать заранее уровень значимости α, есть возможность найти из соотношения $P(\lambda_\alpha) = \alpha$ соответствующее критическое значение λ_α, которое может дать нам ответ о справедливости гипотезы $H0$: если $\lambda < \lambda_\alpha$, то $H0$ верна, то есть функция распределения согласуется с нашими данным; если $\lambda > \lambda\alpha$, то гипотеза $H0$ отклоняется в пользу конкурирующей $H1$.

Выделив основные моменты из теории, которые пригодятся нам в дальнейшей работе, можно приступить к моделированию данных.

Раздел 4. Проверка гипотезы на смоделированных данных

Перед тем, как приступить к моделированию, попробуем самостоятельно написать функцию, которая считает статистику критерия Шапиро-Уилка по формуле из теоретической справки. Отметим, что в Python есть встроенная функция stats.shapiro, которая возвращает статистику и p-value этого же критерия. Конечно, хотелось бы сократить время работы и пользоваться имплементированным в программу критерием, но для этого нам понадобиться сравнить полученные значения от собственной и встроенной функции, чтобы быть уверенными в своих действиях.

Было решено рассмотреть выборку объёма n=10: [-1,0,1,2,3,5,6,7,10,15]. Для своей функции нам понадобится найти из таблицы значения коэффициентов a_{n-i+1}

	i				
	1	2	3	4	5
n=10	0.5739	0.3291	0.2141	0.1224	0.0399

Таблица 5. Коэффициенты

Теперь можно приступать к сравнению.

```
import time
sample=[-1,0,1,2,3,5,6,7,10,15]
start_time = time.time()
def stat_shapiro_mine(sample):
    koef_n10=[5739/10**4,3291/10**4,2141/10**4,1224/10**4,399/10**4]
    denominator=0
    numerator=0
    n=len(sample)
    sample=sorted(sample)
    sample_mean=np.mean(sample)
    for j in range(1,n//2+1):
        numerator+=((sample[n-j-1+1]-sample[j-1])*koef_n10[j-1])
    for j in range(n):
        denominator+=(sample[j]-sample_mean)**2
    w=numerator**2/denominator
    return w
W=stat_shapiro_mine(sample)
finish_time = time.time()
start_time1 = time.time()
W1=stats.shapiro(sample)[0]
finish_time1 = time.time()
print('Результат W моей функции=',np.round(W,5))
print('Результат W встроенной функции=', np.round(W1,5))
print("--- %s seconds in my function ---" % (finish_time - start_time))
print("--- %s seconds ---in stats.shapiro" % (finish_time1 - start_time1))
#sample=stats.norm.rvs(loc=0, scale=1, size=n) - если выше захотим проверить на другой выборке
```

```
Результат W моей функции= 0.93476
Результат W встроенной функции= 0.93466
--- 0.001003265380859375 seconds in my function ---
--- 0.0 seconds ---in stats.shapiro
```

Рис. 3 Функции критерия

Видно, что разница между полученными значениями составляет лишь 0.0001, а это значит, что мы имеем право теперь пользоваться stats.shapiro, более того, встроенная функция на выполняется быстрее по времени – оптимальный вариант для нас.

Для начала генерируем 100 000 выборок размером n=250 (примерное количество торговых дней в году) нормального распределения. На основе этого создаем таблицу 999 квантилей. Ниже здесь представлена таблица 9 квантилей.

Квантиль	Значение
0.1	0.990446
0.2	0.992083
0.3	0.993094
0.4	0.993837
0.5	0.994464
0.6	0.995029
0.7	0.995554
0.8	0.996100
0.9	0.996743

Таблица 6. 9 квантилей

Далее находим p-значения моих выборок тремя способами.

1 способ: вручную считаем 1000 значений по Шапиро-Уилка через процентиль. Чтобы выполнить данные вычисления, была взята таблица 999 квантилей как эмпирическая функция распределения.

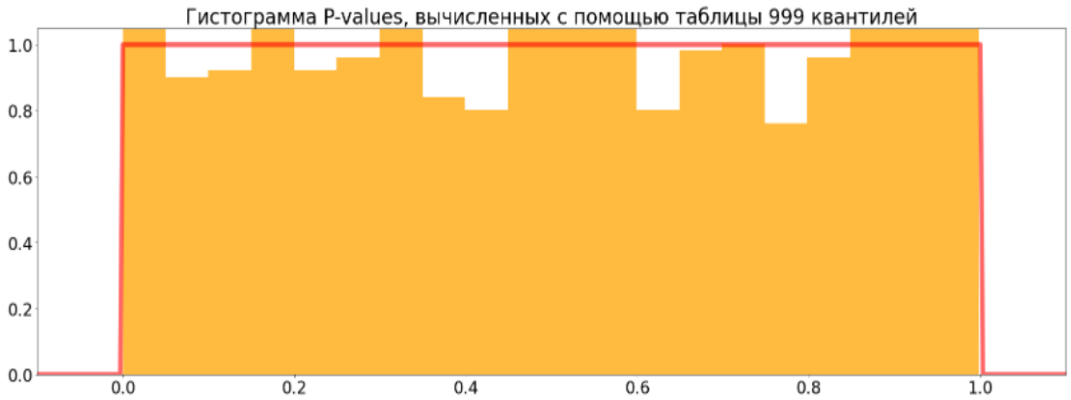


Рис. 4 Гистограмма P-значений 1 способом.

2 способ: используем встроенную функцию stats.shapiro для вычисления p-значений, чтобы сопоставить эту гистограмму с гистограммой, полученной из собственных вычислений способа выше.

Рис. 5 Гистограмма P-значений 2 способом.

3 способ: считаем p-value через критерий согласия Колмогорова, который проверяет данные выборки на нормальность. Это также возможно выполнить через встроенную функцию в Python stats.kstest.

Рис. 6 Гистограмма p-значений 3 способом.

Теперь проверим равномерность полученных p-значений. Нам это важно, так как если наша нулевая гипотеза H0 верна – то и p-value распределены равномерно на отрезке [0;1]. Нас интересует только список p-значений, посчитанный вручную, так как логично, что встроенная функция на заранее заданные нормальные выборки даёт верный ответ.

Мы получаем значение критерия Колмогорова для 1 способа приблизительно равное 0.60632. Это значит, что p-значения распределены равномерно и это подтверждается нашими расчётами. Для того, чтобы окончательно убедиться в правильности поиска наших

p-value, воспользуемся ещё одной встроенной функцией stats.ks_2samp(p_znach_shapiro,KS_p). К ней на вход подаются две выборки, и она проверяет, распределены ли они обе по одному и тому же закону. На выходе получено число 0.57980 – поэтому нулевая гипотеза H0 принимается.

Раздел 5. Выбор альтернативной гипотезы и оценка мощности критерия

Перед тем, как выбрать альтернативные гипотезы, сначала было изучено, какие распределения близки к нормальному, чтобы их можно было принять. По итогу нам подходят – распределение T (Стьюдента), распределение Лапласа и $\chi 2$ распределение. Например, T-распределение, как и нормальное, симметрично и имеет плавную форму. $\chi 2$ рассматриваем с двумя степенями свободы, а распределение Лапласа – с параметрами (0;1).

Вот так это выглядит:

```
data_chi=stats.chi2.rvs(df=2,loc=0, scale=1, size=n)
data_lap=stats.laplace.rvs(loc=0, scale=1, size=n)
data_t=stats.t.rvs(df=2,loc=0, scale=1, size=n)
```

Рис.7 Альтернативные распределения на Python

Тысячу раз генерируем выборки трёх распределений объёма n=250, 125, 30 при уровне значимости α=0.05. Это было сделано для того, чтобы посмотреть, как изменяется мощность критерия Шапиро-Уилка в зависимости от распределения и его объёма.

	Распределение	Мощность при n=250	Мощность при n=125	Мощность при n=30
0	Хи-квадрат	0.0	0.000	0.000
1	Стьюдент	0.0	0.000	0.103
2	Лапласа	0.0	0.014	0.217

Таблица 7. Мощность критерия при альтернативных гипотезах.

Чем выше мощность – тем больше вероятность принять альтернативную гипотезу. Проанализировав эту таблицу, становится ясно, что мощность критерия для всех трёх распределений мала при выборках объема n=250 и n=125 – значит очень мала вероятность принять H0, когда она неверна, то есть совершить ошибку второго рода

почти нереально (отклонение верной H1). Однако для Лапласа при n=30 больше вероятность принятия его распределения по критерию Шапиро-Уилка.

Раздел 6. Проверка гипотезы на реальных данных

После того, как мы убедились в адекватности наших выводов и вычислений на модельных данных, можно перейти к реальным – исследуем дневные логарифмические доходности выбранных мною компаний из индекса DAX.

Временной промежуток: год

Начну с нахождения p-value всем компаниям за каждый год с 2016 по 2020г. Алгоритм работы схож с работой над сгенерированными ранее данными.

	2016	2017	2018	2019	2020
ALV	0.00000	0.00033	0.00027	0.00000	0.0
BAS	0.00027	0.00968	0.61563	0.00029	0.0
BAYN	0.00000	0.00070	0.00000	0.00000	0.0
BMW	0.03062	0.00027	0.00028	0.00028	0.0
DPW	0.00000	0.00000	0.00000	0.01702	0.0
DTE	0.00035	0.00000	0.00505	0.00000	0.0
SAP	0.00002	0.00096	0.00025	0.00000	0.0
SIE	0.00000	0.00000	0.00002	0.00015	0.0

Таблица 8. P-значения за год

Гипотеза принимается лишь в 8% при 1% уровне значимости и в 2% при 5% уровне значимости. Теперь давайте посмотрим, как выглядит гистограмма.

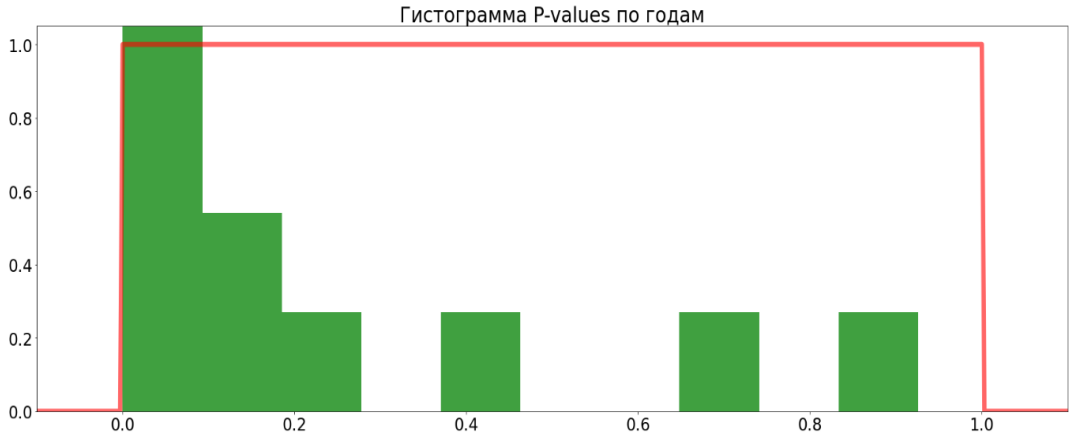


Рис 8. Гистограмма p-значений по годам.

По критерию Колмогорова-Смирнова получено очень маленькое число, близкое к нулю: 4.7793625063075226e-09. Поэтому придётся уменьшать объём выборки.

Временной промежуток: полгода

	2016	2017	2018	2019	2020
ALV	0.00000	0.00016	0.02771	0.00000	0.00000
BAS	0.03920	0.00447	0.92571	0.00102	0.00663
BAYN	0.00006	0.00078	0.03656	0.00001	0.00051
BMW	0.46173	0.16264	0.06438	0.00076	0.00001
DPW	0.02243	0.00003	0.00000	0.68933	0.00008
DTE	0.20697	0.00000	0.07121	0.00000	0.00077
SAP	0.01159	0.00006	0.11121	0.00000	0.00012
SIE	0.00002	0.00000	0.00158	0.04118	0.00077

Таблица 9. P-значения за полгода.

Пока что почти все p-value также близки к нулю. Гипотеза принимается лишь в 35% при 1% уровне значимости и в 20% при 5% уровне значимости. Так выглядит гистограмма:

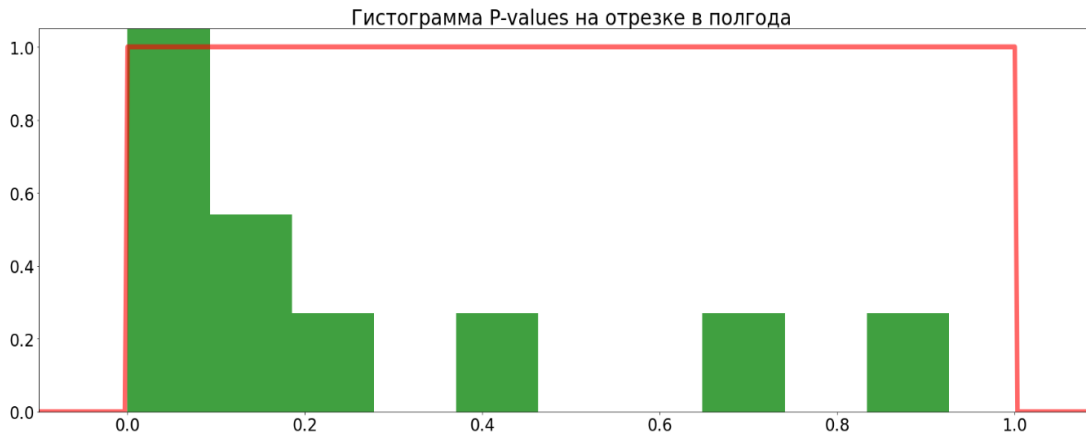


Рис 9. Гистограмма P-значений за полгода

Значение критерия К.С всё ещё мало – 3.769667186512969e-26.

Временной промежуток: один квартал

Рассмотрим данные по компаниям на каждый год с 1 января по 30 марта.

	2016	2017	2018	2019	2020
ALV	0.63161	0.00570	0.00832	0.00064	0.00000
BAS	0.58922	0.31570	0.54158	0.00038	0.00001
BAYN	0.14370	0.38698	0.26983	0.00008	0.00027
BMW	0.12009	0.49366	0.40503	0.10166	0.00000
DPW	0.80293	0.97103	0.12681	0.89547	0.00044
DTE	0.28332	0.01205	0.99146	0.00004	0.00201
SAP	0.14120	0.84513	0.35158	0.11130	0.00110
SIE	0.00118	0.00022	0.07087	0.04621	0.00001

Таблица 10. P-значения за квартал

Теперь значения понемногу становятся приятнее. Гипотеза принимается в 60% при 1% уровне значимости и в 55% при 5% уровне значимости.

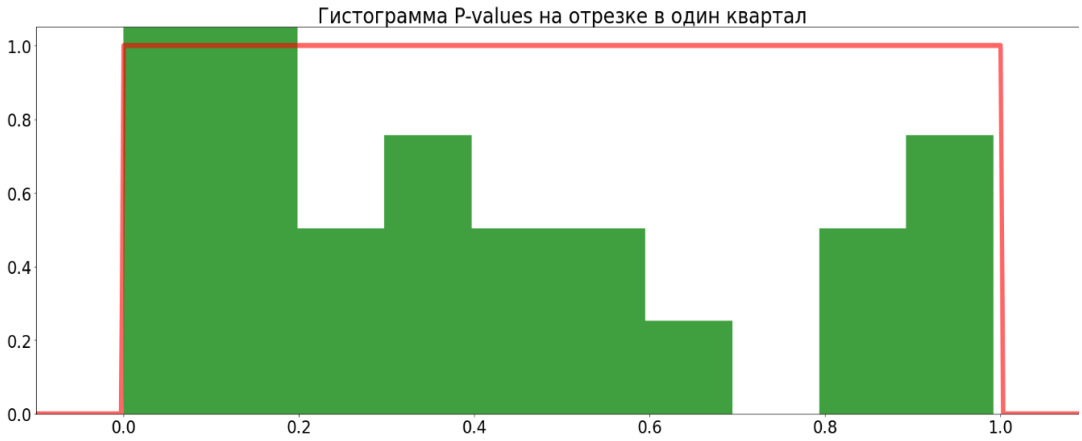


Рис 10. Гистограмма P-значений за квартал.

Гистограмма и критерий КС=4.7793625063075226e-09 дают понять, что нам надо продолжать урезать объем выборок.

Временной промежуток: один месяц

Мы взяли на рассмотрение март месяц у компаний из DAX на все исследуемые года.

	2016	2017	2018	2019	2020
ALV	0.05014	0.84382	0.14729	0.93683	0.15385
BAS	0.44318	0.15685	0.47627	0.52246	0.33255
BAYN	0.99148	0.37145	0.14963	0.00168	0.57819
BMW	0.79613	0.39255	0.22171	0.23044	0.03613
DPW	0.12440	0.97144	0.65925	0.03588	0.71256
DTE	0.50307	0.56302	0.27058	0.00005	0.99288
SAP	0.48781	0.64899	0.81732	0.82715	0.48789
SIE	0.46452	0.48733	0.18221	0.00228	0.48151

Таблица 11. P-значения за месяц

Теперь гипотеза принимается в 92% при 1% уровне значимости и в 88% при 5% уровне значимости. Значение критерия К.С=0.28018 нас устраивает. Следовательно, только после уменьшения начального объёма выборок в 8 раз мы получили подтверждение нулевой гипотезы. Это имеет место быть, потому что исследуемый мной

критерий Шапиро-Уилка эффективен только для выборок с небольшим объёмом. К сожалению, в практике не имеет смысла исследовать столь короткие сроки котировок акций, поэтому придётся воспользоваться другими критериями для возможного анализа.

Заключение

Суммируя, в ходе моей работы была рассмотрена гипотеза о нормальном распределении логарифмических доходностей акций немецких компаний индекса DAX за 5 лет с 2016 по 2020 год. В качестве статистического критерия для анализа мною был выбран критерий Шапиро-Уилка.

Иследование началось с предварительного анализа компаний, во время которого пришлось удалить организации, неподходящие для дальнейшей обработки. Далее мы удостоверились в работоспособности подсчёта p-значений на модельных данных и после этого приступили к реальным данным. Чтобы было проще делать выводы по моим действиям, были визуализированы гистограммы и таблицы p-values.

На модельных данных нулевая гипотеза подтвердилась, однако в 3 из 4 случаях на реальных данных H0 отвергалась по критерию согласия Колмогорова-Смирнова на нормальность и полученной гистограмме p-значений. Для выборки длинной в месяц гипотеза принялась в 88% при 5% уровне значимости, это обусловлено особенностью самого критерия.

Таким образом, мы не сможем сделать прогноз на изменение цен акций в долгосрочной перспективе. Аналогичный вывод был выдвинут и в предыдущих курсовых по критерию Шапиро-Уилка.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ИНТЕРНЕТ-РЕСУРСОВ

- Браилов А.В. Лекции по математической статистике. – М.: Финакадемия, 2007. – 172 с. (Дата обращения: 27.04.2022).
- Кобзарь А.И. Прикладная математическая статистика. Для инженеров и научных работников. – М.: ФИЗМАТЛИТ, 2006. – 816 с. (Дата обращения: 27.04.2022).
- Shapiro S. S., Wilk M. B. An analysis of variance test for normality. — Biometrika, 1965, 52, No3 — p. 591-611. (Дата обращения: 18.05.2022).
- Солодовников А.С., Бабайцев В.А., Браилов А.В. Математика в экономике: учебник в 3-х ч. Ч. 3. Теория вероятностей и математическая статистика. – М.: Финансы и статистика, 2008. – Ы464 с.: ил. (Дата обращения: 05.05.2022).

Интернет-ресурсы:

- https://www.finam.ru/profile/
- https://docs.scipy.org/doc/scipy-1.8.0/html-scipyorg/reference/
- https://dailypik.com/ger30-index-companies/

ПРИЛОЖЕНИЯ

Приложение 1

Компьютерные характеристики:

1. Тип процессора – AMD A9-9425
2. Тактовая частота – 3.1GHz
3. Частота системной шины – 99.8 MHz
4. Объём кэш-памяти второго уровня (L2) – 2.0Мб.

Приложение 3

Список приложенных файлов:

- весь_код.ipynb
- Таблица 2. Количество торговых дней_до_изменений.csv
- Таблица 2. Количество торговых дней_изменённая.csv
- Таблица 3. Max дневные относительные скачки цен вниз.csv
- Таблица 4. Max дневные относительные скачки цен вверх.csv
- макс сдвиги по ценам вверх и вниз.xlsx
- Таблица 9 квантилей и их значений.csv
- Таблица 999 квантилей и их значений.csv
- 9 файлов формата csv – тикеры компаний

Приложение 2

In [256]:

```
import time
import pandas as pd
import numpy as np
import plotly.graph_objects as go
import plotly.figure_factory as ff
from scipy.stats import norm
import matplotlib.pyplot as plt
```

In [257]:

```
data_ALV=pd.read_csv('ALV.csv', parse_dates=['DATE'])
data_BAS=pd.read_csv('BAS.csv', parse_dates=['DATE'])
data_BAYN=pd.read_csv('BAYN.csv',parse_dates=['DATE'])
data_BMW=pd.read_csv('BMW.csv',parse_dates=['DATE'])
data_DPW=pd.read_csv('DPW.csv',parse_dates=['DATE'])
data_DTE=pd.read_csv('DTE.csv',parse_dates=['DATE'])
data_LIN=pd.read_csv('LIN.csv',parse_dates=['DATE'])
data_SAP=pd.read_csv('SAP.csv',parse_dates=['DATE'])
data_SIE=pd.read_csv('SIE.csv',parse_dates=['DATE'])
```

In [258]:

```
tickers=['ALV', 'BAS', 'BAYN', 'BMW', 'DPW', 'DTE', 'LIN', 'SAP', 'SIE']
companies=[data_ALV,data_BAS,data_BAYN,data_BMW,data_DPW,data_DTE,data_LIN,data_SAP,data_SI
years=[2016,2017,2018,2019,2020]
SaleDays = pd.DataFrame()
SaleDays['Тикер'] = tickers
```

1. Торговые дни

In [259]:

```
for year in years:
    rigth_year_count=0
    list_year=[]
    for com in companies:
        right_year_count=com.query(f'DATE >= "{year}-01-01" & DATE <= "{year}-12-31" ')['DA
        list_year.append(right_year_count)
    SaleDays[str(year)] = list_year
```

In [260]:

```
SaleDays
```

Out[260]:

	Тикер	2016	2017	2018	2019	2020
0	ALV	255	252	239	249	253
1	BAS	255	252	239	249	253
2	BAYN	255	252	239	249	253
3	BMW	255	252	249	249	253
4	DPW	255	252	241	249	252
5	DTE	255	252	241	249	252
6	LIN	0	0	0	0	90
7	SAP	255	252	241	249	252
8	SIE	255	252	241	249	252

In [261]:

```
SaleDays.to_csv ('Таблица 2. Количество торговых дней.csv')
```

In [111]:

```
#так как мы не сможем исследовать компанию Linde, удалим ее из полученной таблицы
```

In [262]:

```
SaleDays_new=SaleDays.drop(index=6)
```

In [263]:

```
SaleDays_new
```

Out[263]:

	Тикер	2016	2017	2018	2019	2020
0	ALV	255	252	239	249	253
1	BAS	255	252	239	249	253
2	BAYN	255	252	239	249	253
3	BMW	255	252	249	249	253
4	DPW	255	252	241	249	252
5	DTE	255	252	241	249	252
7	SAP	255	252	241	249	252
8	SIE	255	252	241	249	252

In [264]:

```
SaleDays_new.to_csv ('Таблица 2. Количество торговых дней_изменённая.csv')
```

In [265]:

```
#обновляем переменную companies и tickers, так как не используем больше Linde
```

In [266]:

```
tickers=['ALV', 'BAS', 'BAYN', 'BMW', 'DPW', 'DTE', 'SAP', 'SIE']
companies=[data_ALV,data_BAS,data_BAYN,data_BMW,data_DPW,data_DTE,data_SAP,data_SIE]
```

2.1 Максимальные дневные относительные скачки цен вверх (по годам и акциям)

In [267]:

```
max_up=pd.DataFrame()
max_up['Ticker'] = tickers

for year in years:
    d_right=[]
    for com in companies:
        com['LEAP']=com['CLOSE'].pct_change().round(4)
        diap=(com['DATE'] >= pd.to_datetime(f'{year}-01-01')) & (com['DATE'] <= pd.to_datet
        d_znach=com[diap]['LEAP'].max()
        d_right.append(d_znach)
    max_up[str(year)]=d_right

max_up.to_csv ('Таблица 3. Max дневные относительные скачки цен вверх.csv')
```

In [268]:

```
max_up
```

Out[268]:

	Ticker	2016	2017	2018	2019	2020
0	ALV	0.0437	0.0331	0.0312	0.0339	0.1351
1	BAS	0.0362	0.0388	0.0394	0.0491	0.0956
2	BAYN	0.0516	0.0421	0.0522	0.0905	0.0777
3	BMW	0.0474	0.0299	0.0485	0.0407	0.1254
4	DPW	0.0521	0.0487	0.0431	0.0396	0.1108
5	DTE	0.0427	0.0460	0.0347	0.0277	0.0727
6	SAP	0.0568	0.0295	0.0502	0.1197	0.0804
7	SIE	0.0862	0.0563	0.0653	0.0508	0.0997

2.2 Максимальные дневные относительные скачки цен вниз (по годам и акциям)

In [269]:

```
max_down=pd.DataFrame()
max_down['Ticker'] = tickers

for year in years:
    d_right=[]
    for com in companies:
        com['LEAP']=com['CLOSE'].pct_change().round(4)
        diap=(com['DATE'] >= pd.to_datetime(f'{year}-01-01')) & (com['DATE'] <= pd.to_datet
        d_znach=com[diap]['LEAP'].min()
        d_right.append(d_znach)
    max_down[str(year)]=d_right

max_down.to_csv ('Таблица 4. Max дневные относительные скачки цен вниз.csv')
```

In [270]:

```
max_down
```

Out[270]:

	Ticker	2016	2017	2018	2019	2020
0	ALV	-0.1039	-0.0220	-0.0497	-0.0552	-0.1423
1	BAS	-0.0668	-0.0268	-0.0442	-0.0533	-0.1059
2	BAYN	-0.0820	-0.0410	-0.1070	-0.0993	-0.1215
3	BMW	-0.0753	-0.0320	-0.0560	-0.0523	-0.1208
4	DPW	-0.0701	-0.0384	-0.0706	-0.0374	-0.1144
5	DTE	-0.0519	-0.0322	-0.0512	-0.0470	-0.0928
6	SAP	-0.0563	-0.0346	-0.0608	-0.0537	-0.2237
7	SIE	-0.0741	-0.0373	-0.0494	-0.0412	-0.1131

3.1 График цен для акций с максимальным однодневным снижением цены - SAP

In [271]:

```
data_SAP
```

Out[271]:

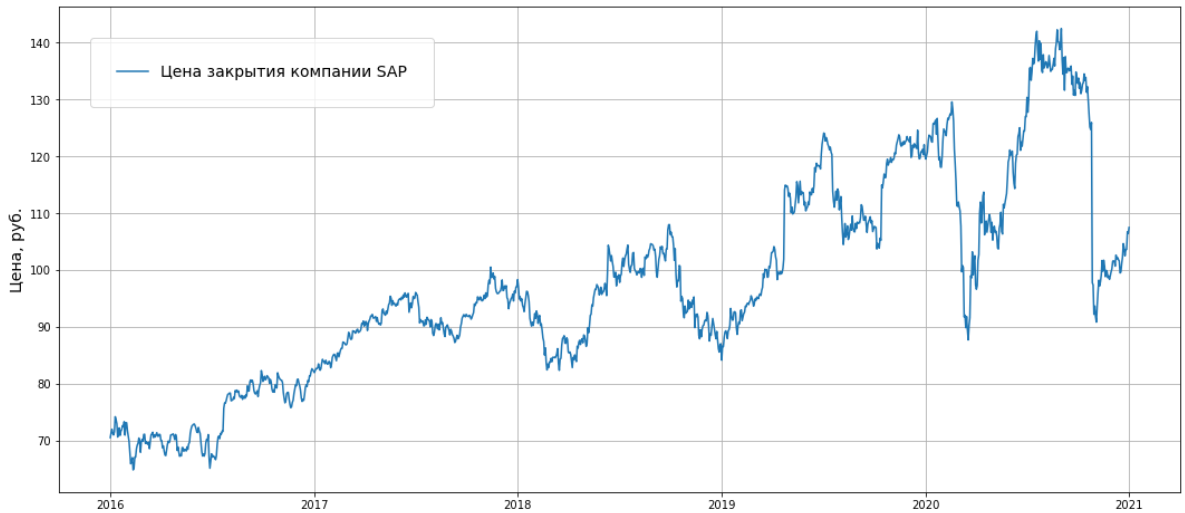
	TICKER	PER	DATE	TIME	OPEN	HIGH	LOW	CLOSE	VOL	LEAP
0	SAP	D	2016-01-04	0	71.50	71.76	70.10	70.58	4570793	NaN
1	SAP	D	2016-01-05	0	71.74	71.74	70.17	71.43	2771848	0.0120
2	SAP	D	2016-01-06	0	73.50	73.78	71.24	72.05	3767568	0.0087
3	SAP	D	2016-01-07	0	69.60	71.94	69.50	71.34	4636705	-0.0099
4	SAP	D	2016-01-08	0	71.90	72.48	71.03	71.05	4104441	-0.0041
...	...	...	...	...	...	...	...	...	...	...
1244	SAP	D	2020-12-22	0	101.54	103.98	101.50	103.62	164428	0.0105
1245	SAP	D	2020-12-23	0	103.30	104.16	103.02	103.66	147954	0.0004
1246	SAP	D	2020-12-28	0	104.98	107.00	104.34	106.76	318651	0.0299
1247	SAP	D	2020-12-29	0	107.50	108.00	106.00	106.46	362539	-0.0028
1248	SAP	D	2020-12-30	0	107.42	107.66	106.66	107.52	131912	0.0100

1249 rows × 10 columns

In [272]:

```
y=data_SAP['CLOSE']
x=np.linspace(2016,2021,len(y))
fig, ax = plt.subplots(figsize=(18, 8))
plt.grid(True)
plt.plot(x, y, label='Цена закрытия компании SAP')
plt.legend(borderaxespad=2, fontsize=14, borderpad = 1.7)
plt.ylabel('Цена, руб.', size=14)
plt.title('Рис 1. График цен закрытия компании SAP', size=16, pad = 19, fontfamily = 'serif
plt.show()
```

Рис 1. График цен закрытия компании SAP



3.2 График цен для акций с максимальным однодневным повышением цены - Allianz

In [273]:

```
y=data_ALV['CLOSE']
x=np.linspace(2016,2021,len(y))
fig, ax = plt.subplots(figsize=(18, 8))
plt.grid(True)
plt.plot(x, y, label='Цена закрытия компании ALV')
plt.legend(borderaxespad=2, fontsize=14, borderpad = 1.7)
plt.ylabel('Цена, руб.', size=14)
plt.title('Рис 2. График цен закрытия компании ALV', size=16, pad = 19, fontfamily = 'serif
plt.show()
```

Рис 2. График цен закрытия компании ALV

давайте удалим все ненужные для дальнейшего анализы столбцы по каждой компании - PER, Time

In [274]:

```
data_ALV=data_ALV.drop(["PER","TIME"], axis=1)
data_BAS=data_BAS.drop(["PER","TIME"], axis=1)
data_BAYN=data_BAYN.drop(["PER","TIME"], axis=1)
data_BMW=data_BMW.drop(["PER","TIME"], axis=1)
data_DPW=data_DPW.drop(["PER","TIME"], axis=1)
data_DTE=data_DTE.drop(["PER","TIME"], axis=1)
data_SAP=data_SAP.drop(["PER","TIME"], axis=1)
data_SIE=data_SIE.drop(["PER","TIME"], axis=1)
```

In [275]:

```
data_SIE
```

Out[275]:

	TICKER	DATE	OPEN	HIGH	LOW	CLOSE	VOL	LEAP
0	SIE	2016-01-04	88.09	88.09	85.21	85.53	4326103	NaN
1	SIE	2016-01-05	85.50	86.35	85.00	86.07	2649739	0.0063
2	SIE	2016-01-06	85.51	85.78	84.45	85.73	2509210	-0.0040
3	SIE	2016-01-07	83.68	84.61	82.75	84.09	4010217	-0.0191
4	SIE	2016-01-08	84.75	85.66	83.33	83.51	3476413	-0.0069
...	...	...	...	...	...	...	...	...
1244	SIE	2020-12-22	113.68	114.90	113.50	114.34	40812	-0.0054
1245	SIE	2020-12-23	114.28	117.08	114.28	116.74	59747	0.0210
1246	SIE	2020-12-28	117.20	119.30	117.20	119.00	106596	0.0194
1247	SIE	2020-12-29	119.10	119.74	116.64	116.78	89969	-0.0187
1248	SIE	2020-12-30	117.24	118.26	117.14	117.90	56513	0.0096

1249 rows × 8 columns

Раздел 4. Проверка гипотезы на модельных данных

In [295]:

```
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as stats
import matplotlib.ticker as ticker
from scipy.special import *
```

3.2.2.18.1. Применение критерия Шапиро–Уилка [1)]

Пусть имеется k независимых выборок объема n_i каждая $(i = 1, \ldots, k)$ и $x_{i1} \leqslant x_{i2} \leqslant \ldots \leqslant x_{in_i}$ для каждой i-й выборки. Для каждой выборки вычислим статистику

$$W_i = \frac{\left\{ \sum_{j=1}^{\left[\frac{n}{2}\right]} a_j (x_{i,n_i-j+1} - x_{ij}) \right\}^2}{\sum_{j=1}^{n_i} (x_{ij} - \bar{x}_i)^2}.$$

In [286]:

```
import time
sample=[-1,0,1,2,3,5,6,7,10,15]
start_time = time.time()
def stat_shapiro_mine(sample):
    koef_n10=[5739/10**4,3291/10**4,2141/10**4,1224/10**4,399/10**4]
    denominator=0
    numerator=0
    n=len(sample)
    sample=sorted(sample)
    sample_mean=np.mean(sample)
    for j in range(1,n//2+1):
        numerator+=((sample[n-j-1+1]-sample[j-1])*koef_n10[j-1])
    for j in range(n):
        denominator+=(sample[j]-sample_mean)**2
    w=numerator**2/denominator
    return w
W=stat_shapiro_mine(sample)
finish_time = time.time()
start_time1 = time.time()
W1=stats.shapiro(sample)[0]
finish_time1 = time.time()
print('Результат W моей функции=',np.round(W,5))
print('Результат W встроенной функции=', np.round(W1,5))
print("--- %s seconds in my function ---" % (finish_time - start_time))
print("--- %s seconds ---in stats.shapiro" % (finish_time1 - start_time1))
#sample=stats.norm.rvs(loc=0, scale=1, size=n) - если выше захотим проверить на другой выбо
```

```
Результат W моей функции= 0.93476
Результат W встроенной функции= 0.93466
--- 0.001003265380859375 seconds in my function ---
--- 0.0 seconds ---in stats.shapiro
```

1) строим гистограмму p-значений по Шапиро-Уилка с помощью метода Монте-Карло

для начала рассчитаем статистику каждой выборки и значения квантилей

In [134]:

```
N=10**5 #количество испытаний
n=249 #объем каждой из выборок, соотвт. среднему количеству торговых дней за год
list_for_q9=np.arange(0.1,1,0.1) #будущие 9 квантилей
list_for_q999=np.arange(0.001,1,0.001) #будущие 999 квантилей
list_q9=[] #здесь будут записаны значения 9 квантилей
list_q999=[] #здесь будут записаны значения 999 квантилей
stat_shapiro=[] #список статистик каждой из N выборок
stat_shapiro_p=[]
KS_p=[]
for i in range(N):
    data_norm=stats.norm.rvs(loc=0, scale=1, size=n) #генерируем выборку из нормального рас
    stat_shapiro.append(stats.shapiro(data_norm)[0]) #находим значение статистики критерия
    stat_shapiro_p.append(stats.shapiro(data_norm)[1]) #через встроенную функцию находим p-
    KS_p.append(stats.kstest(data_norm,'norm')[1])


list_q9=np.quantile(stat_shapiro,list_for_q9) #заполняем значения 9 квантилей
list_q999=np.quantile(stat_shapiro,list_for_q999)  #заполняем значения 999 квантилей

#создадим таблицу 9 и 999 квантилей

q9= pd.DataFrame({'Квантиль':list_for_q9, 'Значение':list_q9})
q999= pd.DataFrame({'Квантиль':list_for_q999, 'Значение':list_q999})

#сохраним таблицы в csv
```

In [161]:

```
q9.to_csv ('Таблица 9 квантилей и их значений')
q999.to_csv ('Таблица 999 квантилей и их значений')
```

In [162]:

```
q9
```

Out[162]:

	Квантиль	Значение
0	0.1	0.990446
1	0.2	0.992083
2	0.3	0.993094
3	0.4	0.993837
4	0.5	0.994464
5	0.6	0.995029
6	0.7	0.995554
7	0.8	0.996100
8	0.9	0.996743

In [136]:

```
q999
```

Out[136]:

	Квантиль	Значение
0	0.001	0.979107
1	0.002	0.980682
2	0.003	0.981564
3	0.004	0.982257
4	0.005	0.982897
...	...	...
994	0.995	0.998077
995	0.996	0.998139
996	0.997	0.998204
997	0.998	0.998286
998	0.999	0.998430

999 rows × 2 columns

теперь перейдём к вычислению 1000 p-значений выборок аналитически

In [137]:

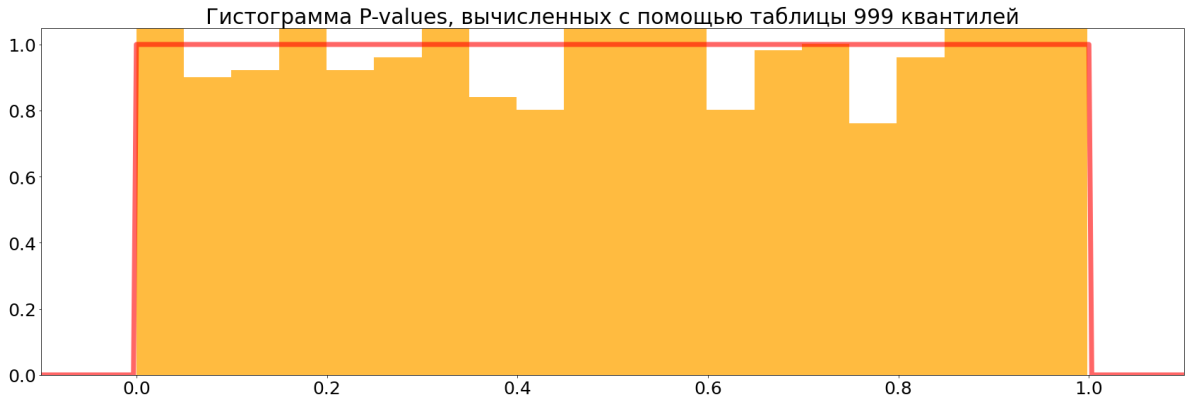
```
p_znach_shapiro=[stats.percentileofscore(list_q999,stat_shapiro[i])/100 for i in range(1000
```

In [288]:

```
U=stats.uniform()
u = np.linspace(-1, 2, 1000)
y=U.pdf(u)
```

In [145]:

```
fig,ax =plt.subplots(figsize=(29, 9))
plt.xlim(-0.1,1.1)
plt.ylim(0.0,1.05)
plt.tick_params(labelsize = 25)
plt.plot(u, y, 'r-', lw=7, alpha=0.6)
plt.hist(p_znach_shapiro, density=True,bins=20,alpha=0.75, color='orange')
plt.title('Гистограмма P-values, вычисленных с помощью таблицы 999 квантилей', fontsize=30)
plt.show()
```



In [146]:

```
pvalue111 = stats.kstest(p_znach_shapiro,'uniform')
```

In [147]:

```
pvalue111
```

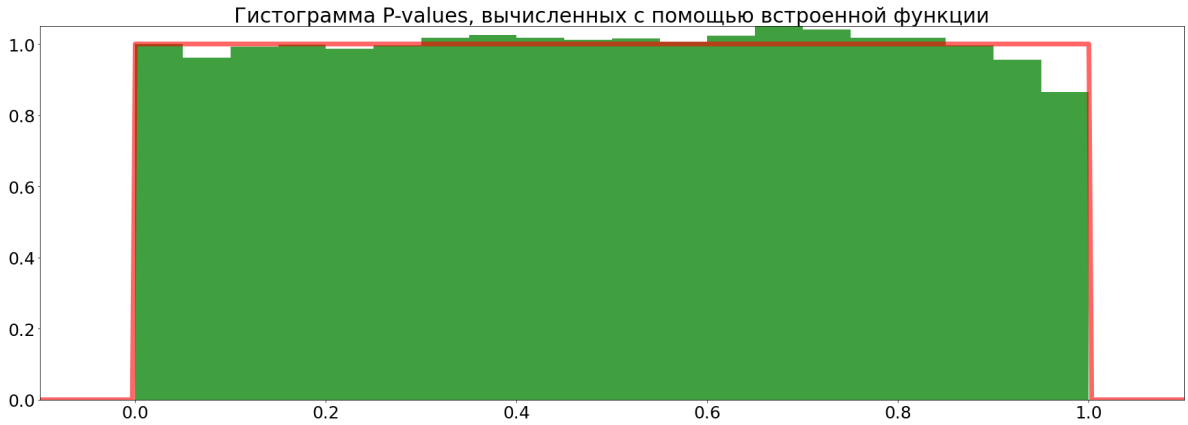
Out[147]:

```
KstestResult(statistic=0.023944944944944944, pvalue=0.6063268920229192)
```

2) строим гистограмму p-значений по Шапиро-Уилка, полученных из встроенной функции на ЭТИХ же данных

In [149]:

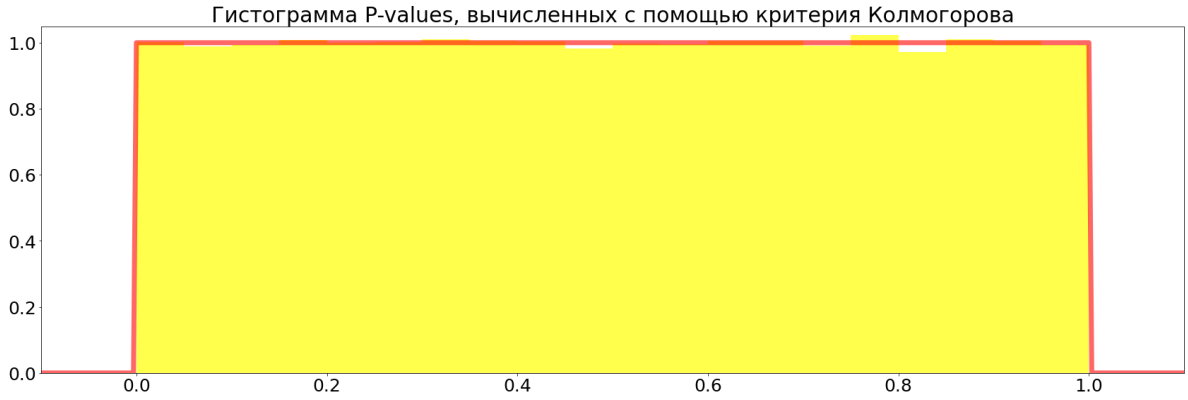
```
fig,ax =plt.subplots(figsize=(30, 10))
plt.xlim(-0.1,1.1)
plt.ylim(0.0,1.05)
plt.tick_params(labelsize = 25)
plt.plot(u, y, 'r-', lw=7, alpha=0.6)
plt.hist(stat_shapiro_p,density=True,bins=20,alpha=0.75, color='green')
plt.title('Гистограмма P-values, вычисленных с помощью встроенной функции',fontsize=30)
plt.show()
```



3) строим гистограмму p-значений по критерию Колмогорова

In [153]:

```
fig,ax =plt.subplots(figsize=(29, 9))
plt.tick_params(labelsize = 25)
plt.xlim(-0.1,1.1)
plt.ylim(0.0,1.05)
plt.plot(u, y, 'r-', lw=7, alpha=0.6)
plt.hist(KS_p,bins=20,alpha=0.7, density=True, color='yellow')
plt.title('Гистограмма P-values, вычисленных с помощью критерия Колмогорова',fontsize=30)
plt.show()
```



In [154]:

```
#проверка, что распределения p-значений идентичны
pvalueend= stats.ks_2samp(p_znach_shapiro,KS_p)
```

In [84]:

```
pvalueend
```

Out[84]:

```
KstestResult(statistic=0.02457000000000001, pvalue=0.579806403842505)
```

Раздел 5. Выбор альтернативной гипотезы и оценка мощности критерия

In [ ]:

```
#распределение T (Стьюдента), распределение Лапласа и X2 распределение.
```

In [303]:

```
#### n=250
alpha=0.5
stat_shapiro_p_chi=[]
stat_shapiro_p_laplace=[]
stat_shapiro_p_t=[]

n=250
for i in range(1000):
    data_chi=stats.chi2.rvs(df=2,loc=0, scale=1, size=n)
    data_lap=stats.laplace.rvs(loc=0, scale=1, size=n)
    data_t=stats.t.rvs(df=2,loc=0, scale=1, size=n)
    stat_shapiro_p_chi.append(stats.shapiro(data_chi)[1])
    stat_shapiro_p_laplace.append(stats.shapiro(data_lap)[1])
    stat_shapiro_p_t.append(stats.shapiro(data_t)[1])

pow_chi_250=0
pow_t_250=0
pow_lap_250=0

for i in stat_shapiro_p_chi:
    if i>alpha:
        pow_chi_250+=1

for i in stat_shapiro_p_laplace:
    if i>alpha:
        pow_lap_250+=1

for i in stat_shapiro_p_t:
    if i>alpha:
        pow_t_250+=1

pow_250=[pow_chi_250/1000,pow_t_250/1000,pow_lap_250/1000 ]

########n=125
stat_shapiro_p_chi=[]
stat_shapiro_p_laplace=[]
stat_shapiro_p_t=[]
n=125
for i in range(1000):
    data_chi=stats.chi2.rvs(df=2,loc=0, scale=1, size=n)
    data_lap=stats.laplace.rvs(loc=0, scale=1, size=n)
    data_t=stats.t.rvs(df=2,loc=0, scale=1, size=n)
    stat_shapiro_p_chi.append(stats.shapiro(data_chi)[1])
    stat_shapiro_p_laplace.append(stats.shapiro(data_lap)[1])
    stat_shapiro_p_t.append(stats.shapiro(data_t)[1])

pow_chi_125=0
pow_t_125=0
pow_lap_125=0

for i in stat_shapiro_p_chi:
    if i>alpha:
        pow_chi_125+=1

for i in stat_shapiro_p_laplace:
    if i>alpha:
        pow_lap_125+=1

for i in stat_shapiro_p_t:
```

```
    if i>alpha:
        pow_t_125+=1

pow_125=[pow_chi_125/1000,pow_t_125/1000,pow_lap_125/1000 ]

#######n=30
stat_shapiro_p_chi=[]
stat_shapiro_p_laplace=[]
stat_shapiro_p_t=[]
n=30
for i in range(1000):
    data_chi=stats.chi2.rvs(df=2,loc=0, scale=1, size=n)
    data_lap=stats.laplace.rvs(loc=0, scale=1, size=n)
    data_t=stats.t.rvs(df=2,loc=0, scale=1, size=n)
    stat_shapiro_p_chi.append(stats.shapiro(data_chi)[1])
    stat_shapiro_p_laplace.append(stats.shapiro(data_lap)[1])
    stat_shapiro_p_t.append(stats.shapiro(data_t)[1])

pow_chi_30=0
pow_t_30=0
pow_lap_30=0

for i in stat_shapiro_p_chi:
    if i>alpha:
        pow_chi_30+=1

for i in stat_shapiro_p_laplace:
    if i>alpha:
        pow_lap_30+=1

for i in stat_shapiro_p_t:
    if i>alpha:
        pow_t_30+=1

pow_30=[pow_chi_30/1000,pow_t_30/1000,pow_lap_30/1000 ]

list_raspr=['Хи-квадрат','Стьюдент','Лапласа']
power= pd.DataFrame({'Распределение':list_raspr, 'Мощность при n=250':pow_250,'Мощность при
```

In [304]:

```
power
```

Out[304]:

	Распределение	Мощность при n=250	Мощность при n=125	Мощность при n=30
0	Хи-квадрат	0.0	0.000	0.000
1	Стьюдент	0.0	0.000	0.103
2	Лапласа	0.0	0.014	0.217

Раздел 6. Реальные данные

In [203]:

```
#найдем логарифмические доходности на каждый день по всем компаниям
```

In [279]:

```
tickers=['ALV', 'BAS', 'BAYN', 'BMW', 'DPW', 'DTE', 'SAP', 'SIE']
companies=[data_ALV,data_BAS,data_BAYN,data_BMW,data_DPW,data_DTE,data_SAP,data_SIE]
def log_kek(com): #функция для подсчета дневной логарифмической доходности
    log_com=[np.round(np.log(com['CLOSE'][i+1]/com['CLOSE'][i]),4) for i in range(len(com)-
    log_com.insert(0,0)
    return log_com
data_ALV['LOG LEAP']=log_kek(data_ALV)
data_BAS['LOG LEAP']=log_kek(data_BAS)
data_BAYN['LOG LEAP']=log_kek(data_BAYN)
data_BMW['LOG LEAP']=log_kek(data_BMW)
data_DPW['LOG LEAP']=log_kek(data_DPW)
data_DTE['LOG LEAP']=log_kek(data_DTE)
data_SAP['LOG LEAP']=log_kek(data_SAP)
data_SIE['LOG LEAP']=log_kek(data_SIE)
```

In [ ]:

```
#ПРОМЕЖУТОК ГОД
```

In [158]:

```
years=[2016,2017,2018,2019,2020]
p_real=pd.DataFrame(index=tickers)
p_okrugl=pd.DataFrame(index=tickers)
p_all=[]
for year in years:
    p_per_all=[]
    p_per_all_okrug=[]
    for com in companies:
        diap=(com['DATE'] >= pd.to_datetime(f'{year}-01-01')) & (com['DATE'] <= pd.to_datet
        p_znach=stats.shapiro(com[diap]['LOG LEAP'])[1]
        p_znach_okrug=np.round(stats.shapiro(com[diap]['LOG LEAP'])[1],5)
        p_per_all.append(p_znach)
        p_per_all_okrug.append(p_znach_okrug)
        p_all.append(p_znach)
    p_real[str(year)]=p_per_all
    p_okrugl[str(year)]=p_per_all_okrug

k1,k5=0,0
for i in range(len(p_all)):
    if p_all[i]>=0.01:
        k1+=1
    if p_all[i]>=0.05:
        k5+=1

print(f"Гипотеза принимается лишь в {k1/len(p_all):.0%} при 1% уровне значимости")
print(f"Гипотеза принимается лишь в {k5/len(p_all):.0%} при 5% уровне значимости")
```

```
Гипотеза принимается лишь в 8% при 1% уровне значимости
Гипотеза принимается лишь в 2% при 5% уровне значимости
```

In [96]:

```
p_real
```

Out[96]:

	2016	2017	2018	2019	2020
ALV	1.583984e-10	3.304577e-04	2.662125e-04	2.397460e-09	3.231395e-13
BAS	2.730699e-04	9.677988e-03	6.156262e-01	2.922494e-04	3.428819e-07
BAYN	1.150762e-07	6.951123e-04	2.892991e-10	2.498536e-08	6.687219e-08
BMW	3.061720e-02	2.708930e-04	2.758724e-04	2.806764e-04	1.054436e-10
DPW	1.570336e-07	2.127672e-06	2.697140e-07	1.701912e-02	1.658740e-11
DTE	3.493496e-04	1.985800e-06	5.050438e-03	3.852072e-08	5.650786e-09
SAP	1.736952e-05	9.640772e-04	2.535483e-04	1.381971e-15	6.952844e-19
SIE	6.639803e-08	3.570996e-08	1.958802e-05	1.486460e-04	1.188475e-08

In [97]:

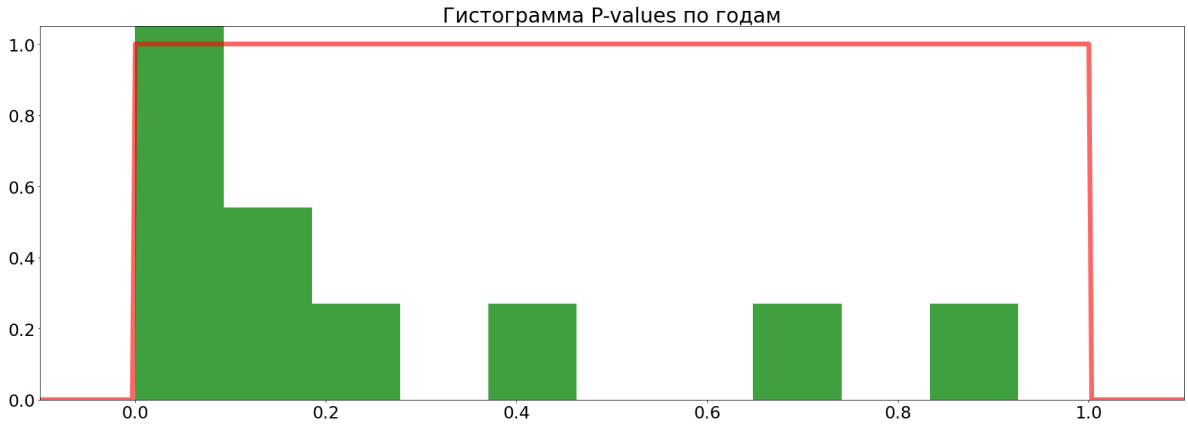
```
p_okrugl
```

Out[97]:

	2016	2017	2018	2019	2020
ALV	0.00000	0.00033	0.00027	0.00000	0.0
BAS	0.00027	0.00968	0.61563	0.00029	0.0
BAYN	0.00000	0.00070	0.00000	0.00000	0.0
BMW	0.03062	0.00027	0.00028	0.00028	0.0
DPW	0.00000	0.00000	0.00000	0.01702	0.0
DTE	0.00035	0.00000	0.00505	0.00000	0.0
SAP	0.00002	0.00096	0.00025	0.00000	0.0
SIE	0.00000	0.00000	0.00002	0.00015	0.0

In [112]:

```
#построим гистограмму всех p-значений
fig,ax =plt.subplots(figsize=(30, 10))
plt.tick_params(labelsize = 25)
plt.xlim(-0.1,1.1)
plt.ylim(0.0,1.05)
plt.plot(u, y, 'r-', lw=7, alpha=0.6)
plt.hist(p_all,alpha=0.75, density=True, color='green')
plt.title('Гистограмма P-values по годам',fontsize=30)
plt.show()
```



In [122]:

```
pv = stats.kstest(p_all,'uniform')
pv[1]
```

Out[122]:

```
4.7793625063075226e-09
```

In [ ]:

```
# ПРОМЕЖУТОК ПОЛГОДА
```

In [159]:

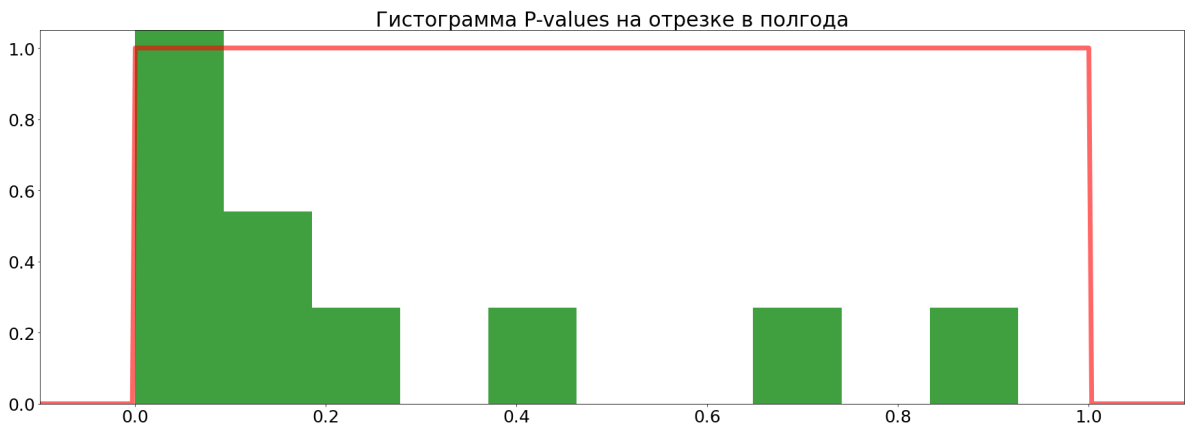
```
years=[2016,2017,2018,2019,2020]
p_real=pd.DataFrame(index=tickers)
p_okrugl=pd.DataFrame(index=tickers)
p_all=[]
for year in years:
    p_per_all=[]
    p_per_all_okrug=[]
    for com in companies:
        diap=(com['DATE'] >= pd.to_datetime(f'{year}-01-01')) & (com['DATE'] <= pd.to_datet
        p_znach=stats.shapiro(com[diap]['LOG LEAP'])[1]
        p_znach_okrug=np.round(stats.shapiro(com[diap]['LOG LEAP'])[1],5)
        p_per_all.append(p_znach)
        p_per_all_okrug.append(p_znach_okrug)
        p_all.append(p_znach)
    p_real[str(year)]=p_per_all
    p_okrugl[str(year)]=p_per_all_okrug

k1,k5=0,0
for i in range(len(p_all)):
    if p_all[i]>=0.01:
        k1+=1
    if p_all[i]>=0.05:
        k5+=1

print(f"Гипотеза принимается лишь в {k1/len(p_all):.0%} при 1% уровне значимости")
print(f"Гипотеза принимается лишь в {k5/len(p_all):.0%} при 5% уровне значимости")

#построим гистограмму p-значений промежутка в полгода
fig,ax =plt.subplots(figsize=(30, 10))
plt.tick_params(labelsize = 25)
plt.xlim(-0.1,1.1)
plt.ylim(0.0,1.05)
plt.plot(u, y, 'r-', lw=7, alpha=0.6)
plt.hist(p_all,density=True,alpha=0.75, color='green')
plt.title('Гистограмма P-values на отрезке в полгода',fontsize=30)
plt.show()
pv = stats.kstest(p_all,'uniform')
print('значение проверки на равномерность',pv[1])
```

```
Гипотеза принимается лишь в 35% при 1% уровне значимости
Гипотеза принимается лишь в 20% при 5% уровне значимости
```




```
значение проверки на равномерность 3.769667186512969e-26
```

In [103]:

```
p_okrugl
```

Out[103]:

	2016	2017	2018	2019	2020
ALV	0.00000	0.00016	0.02771	0.00000	0.00000
BAS	0.03920	0.00447	0.92571	0.00102	0.00663
BAYN	0.00006	0.00078	0.03656	0.00001	0.00051
BMW	0.46173	0.16264	0.06438	0.00076	0.00001
DPW	0.02243	0.00003	0.00000	0.68933	0.00008
DTE	0.20697	0.00000	0.07121	0.00000	0.00077
SAP	0.01159	0.00006	0.11121	0.00000	0.00012
SIE	0.00002	0.00000	0.00158	0.04118	0.00077

In [ ]:

```
# ПРОМЕЖУТОК ОДИН КВАРТАЛ (3 МЕСЯЦА)
```

In [289]:

```
years=[2016,2017,2018,2019,2020]
p_real=pd.DataFrame(index=tickers)
p_okrugl=pd.DataFrame(index=tickers)
p_all=[]
for year in years:
    p_per_all=[]
    p_per_all_okrug=[]
    for com in companies:
        diap=(com['DATE'] >= pd.to_datetime(f'{year}-01-01')) & (com['DATE'] <= pd.to_datet
        p_znach=stats.shapiro(com[diap]['LOG LEAP'])[1]
        p_znach_okrug=np.round(stats.shapiro(com[diap]['LOG LEAP'])[1],5)
        p_per_all.append(p_znach)
        p_per_all_okrug.append(p_znach_okrug)
        p_all.append(p_znach)
    p_real[str(year)]=p_per_all
    p_okrugl[str(year)]=p_per_all_okrug

k1,k5=0,0
for i in range(len(p_all)):
    if p_all[i]>=0.01:
        k1+=1
    if p_all[i]>=0.05:
        k5+=1

print(f"Гипотеза принимается лишь в {k1/len(p_all):.0%} при 1% уровне значимости")
print(f"Гипотеза принимается лишь в {k5/len(p_all):.0%} при 5% уровне значимости")

#построим гистограмму p-значений промежутка в полгода
fig,ax =plt.subplots(figsize=(30, 10))
plt.tick_params(labelsize = 25)
plt.xlim(-0.1,1.1)
plt.ylim(0.0,1.05)
plt.plot(u, y, 'r-', lw=7, alpha=0.6)
plt.hist(p_all,density=True,alpha=0.75, color='green')
plt.title('Гистограмма P-values на отрезке в один квартал',fontsize=30)
plt.show()
pv = stats.kstest(p_all,'uniform')
print('значение проверки на равномерность',pv[1])
```

```
Гипотеза принимается лишь в 60% при 1% уровне значимости
Гипотеза принимается лишь в 55% при 5% уровне значимости
```


```
значение проверки на равномерность 4.7793625063075226e-09
```

In [290]:

```
p_okrugl
```

Out[290]:

	2016	2017	2018	2019	2020
ALV	0.63161	0.00570	0.00832	0.00064	0.00000
BAS	0.58922	0.31570	0.54158	0.00038	0.00001
BAYN	0.14370	0.38698	0.26983	0.00008	0.00027
BMW	0.12009	0.49366	0.40503	0.10166	0.00000
DPW	0.80293	0.97103	0.12681	0.89547	0.00044
DTE	0.28332	0.01205	0.99146	0.00004	0.00201
SAP	0.14120	0.84513	0.35158	0.11130	0.00110
SIE	0.00118	0.00022	0.07087	0.04621	0.00001

In [291]:

```
p_okrugl.to_csv('p-values за квартал')
```

In [ ]:

```
# ПРОМЕЖУТОК 1 МЕСЯЦ
```

In [292]:

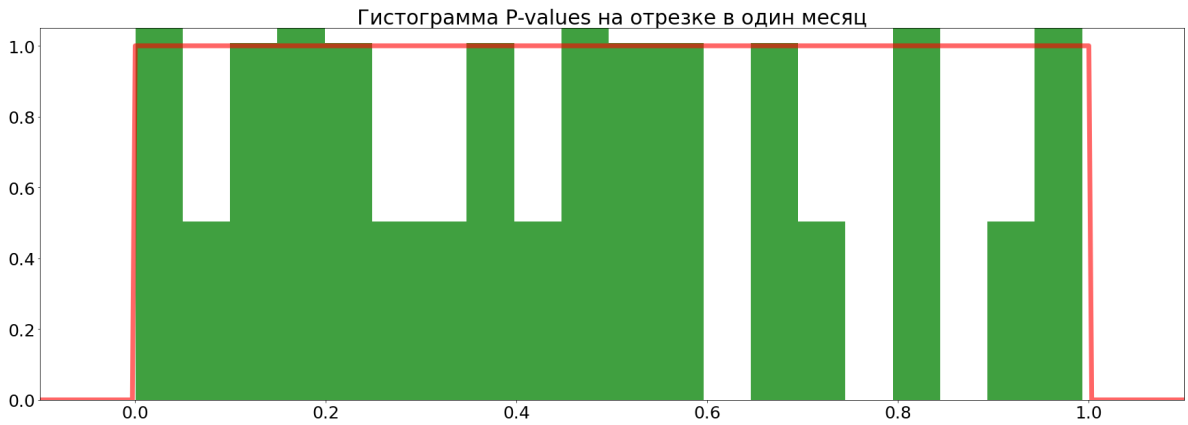
```
years=[2016,2017,2018,2019,2020]
p_real=pd.DataFrame(index=tickers)
p_okrugl=pd.DataFrame(index=tickers)
p_all=[]
for year in years:
    p_per_all=[]
    p_per_all_okrug=[]
    for com in companies:
        diap=(com['DATE'] >= pd.to_datetime(f'{year}-03-01')) & (com['DATE'] <= pd.to_datet
        p_znach=stats.shapiro(com[diap]['LOG LEAP'])[1]
        p_znach_okrug=np.round(stats.shapiro(com[diap]['LOG LEAP'])[1],5)
        p_per_all.append(p_znach)
        p_per_all_okrug.append(p_znach_okrug)
        p_all.append(p_znach)
    p_real[str(year)]=p_per_all
    p_okrugl[str(year)]=p_per_all_okrug

k1,k5=0,0
for i in range(len(p_all)):
    if p_all[i]>=0.01:
        k1+=1
    if p_all[i]>=0.05:
        k5+=1

print(f"Гипотеза принимается в {k1/len(p_all):.0%} при 1% уровне значимости")
print(f"Гипотеза принимается в {k5/len(p_all):.0%} при 5% уровне значимости")

#построим гистограмму p-значений промежутка в полгода
fig,ax =plt.subplots(figsize=(30, 10))
plt.tick_params(labelsize = 25)
plt.xlim(-0.1,1.1)
plt.ylim(0.0,1.05)
plt.plot(u, y, 'r-', lw=7, alpha=0.6)
plt.hist(p_all,bins=20,alpha=0.75, density=True, color='green')
plt.title('Гистограмма P-values на отрезке в один месяц',fontsize=30)
plt.show()
pv = stats.kstest(p_all,'uniform')
print('значение проверки на равномерность',np.round(pv[1],5))
```

```
Гипотеза принимается в 92% при 1% уровне значимости
Гипотеза принимается в 88% при 5% уровне значимости
```




```
значение проверки на равномерность 0.28018
```

In [293]:

```
p_okrugl
```

Out[293]:

	2016	2017	2018	2019	2020
ALV	0.05014	0.84382	0.14729	0.93683	0.15385
BAS	0.44318	0.15685	0.47627	0.52246	0.33255
BAYN	0.99148	0.37145	0.14963	0.00168	0.57819
BMW	0.79613	0.39255	0.22171	0.23044	0.03613
DPW	0.12440	0.97144	0.65925	0.03588	0.71256
DTE	0.50307	0.56302	0.27058	0.00005	0.99288
SAP	0.48781	0.64899	0.81732	0.82715	0.48789
SIE	0.46452	0.48733	0.18221	0.00228	0.48151

In [294]:

```
p_okrugl.to_csv('p-values за месяц')
```

In [ ]:

```

```