Домашнее задание по Математической Статистике

Выполнил

Семенов Всеволод Алексеевич
БЭК215

Создание параметров

Для дальнейшего использования я сгенерировал следующий ряд параметров:

a_1 = 19; b_1 = 3; c_1 = 1
a_2 = 6; b_2 = 19; c_2 = 13
a_3 = 14; b_3 = 6; c_3 = 6

Подбирались по алфавиту в соответствии с условием

Соответствующие строки кода, это важно, поскольку они использовались во всех номерах работы

```
[31] # Параметры в код
     a_1 = 19
     b_1 = 3
     c_1 = 1
     a_2 = 6
     b_2 = 19
     c_2 = 13
     a_3 = 14
     b_3 = 6
     c_3 = 6
```

https://github.com/ExuberantSebastian/HW_MS/tree/main

Задание 1.1

Во всех номерах я буду использовать одни и те же аббревиатуры для распределений

N - Нормальное

Exp или E - показательное

U - равномерное

При генерации выборок использовались функции из scipy.stats , для поиска значений использовались функции из numpy

https://github.com/ExuberantSebastian/HW_MS/tree/main

Примеры кода-генератора и кода, для решения первого пункта

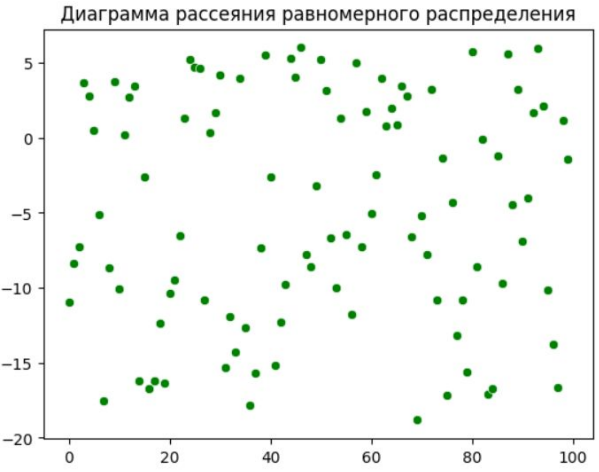
```
# Нормальное
norm_rv = sts.norm(loc=b_1, scale=np.sqrt(b_2))
N = norm_rv.rvs(100)
```

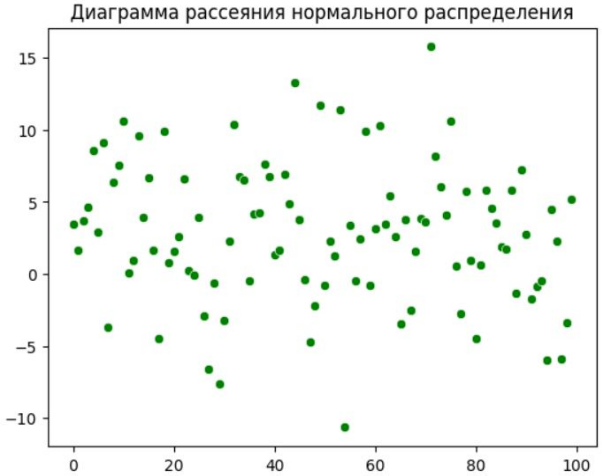
```
# Пункт 4 для нормального

max_N = np.array(N).max()
min_N = np.array(N).min()
mean_N = np.array(N).mean()
std_N = np.array(N).std()
median_N = np.median(np.array(N))

(max_N, min_N), mean_N, std_N, median_N
```

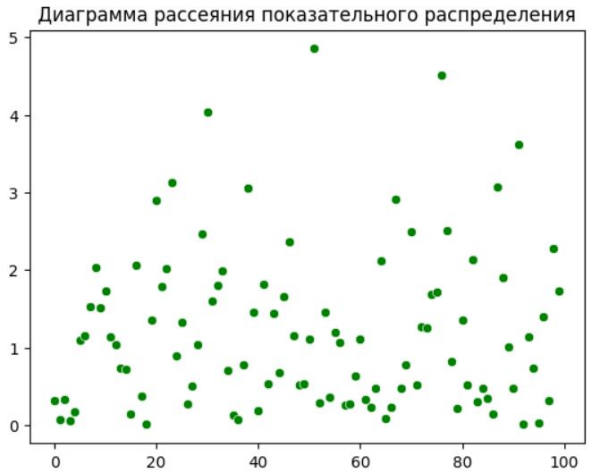
```
((15.76486566573288, -10.652249698801265),
 2.8269636449033704,
 4.8172301362244525,
 2.8151513311460823)
```

Задание 1.1 графики

Для создания использовались библиотеки matplotlib.pyplot и seaborn





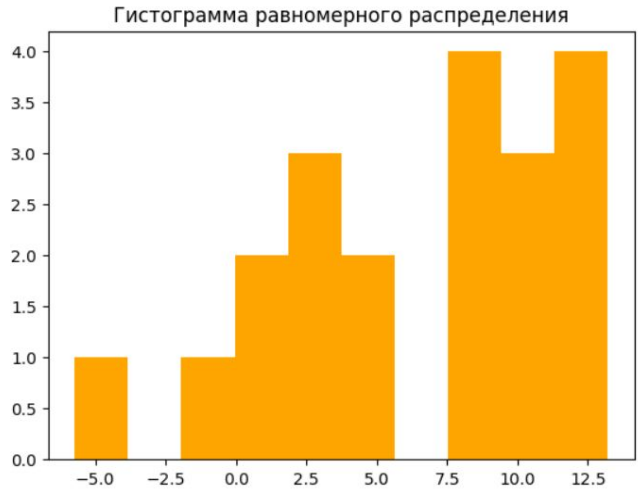


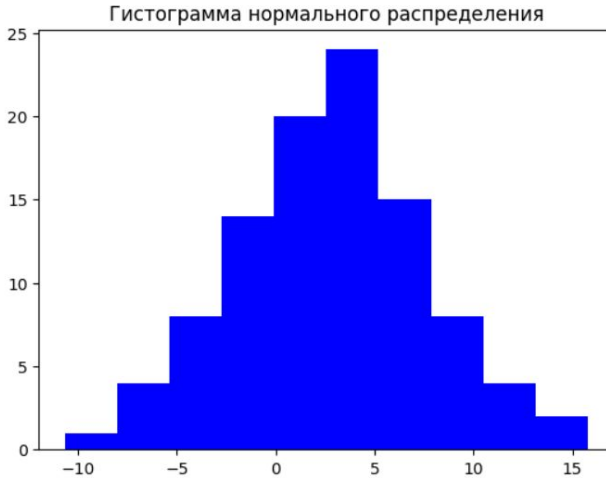
Как мы видим, равномерное распределение распыляется в произвольном порядке, нормальное скапливается около матожидания, а экспоненциальное - около нуля

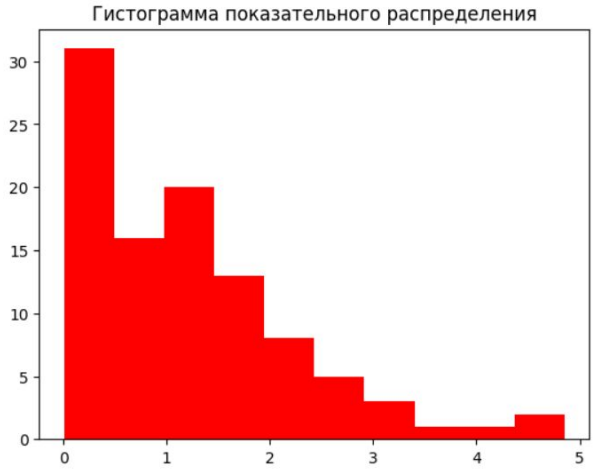
https://github.com/ExuberantSebastian/HW_MS/tree/main

Задание 1.1 графики

Для создания использовались библиотеки matplotlib.pyplot и seaborn





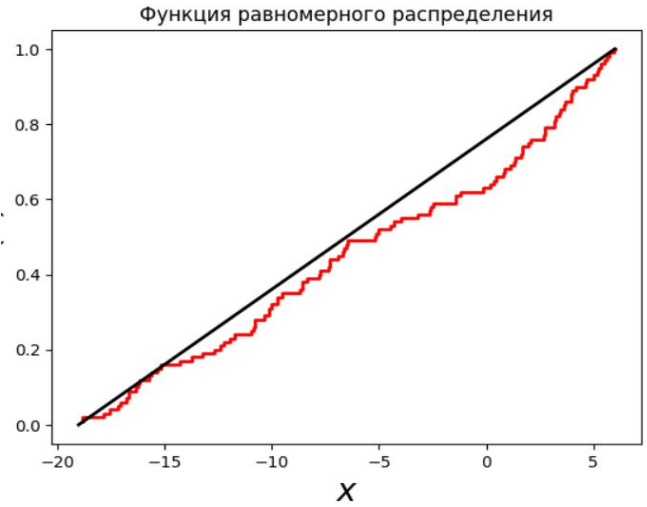


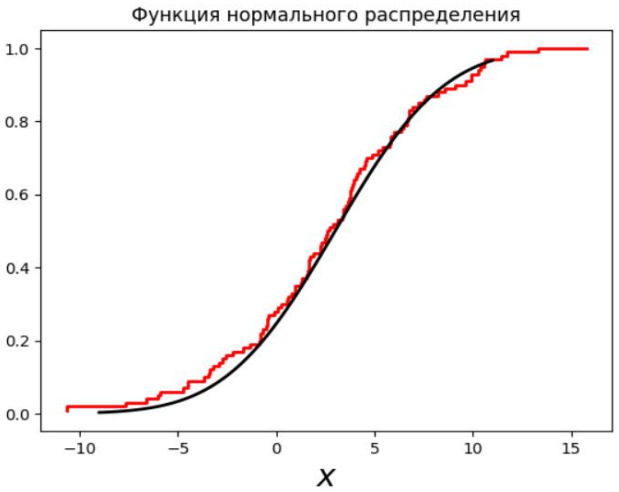
Гистограмма равномерного распределения распределяется в около-произвольном порядке, нормального и экспоненциального - напоминают свои функции плотности

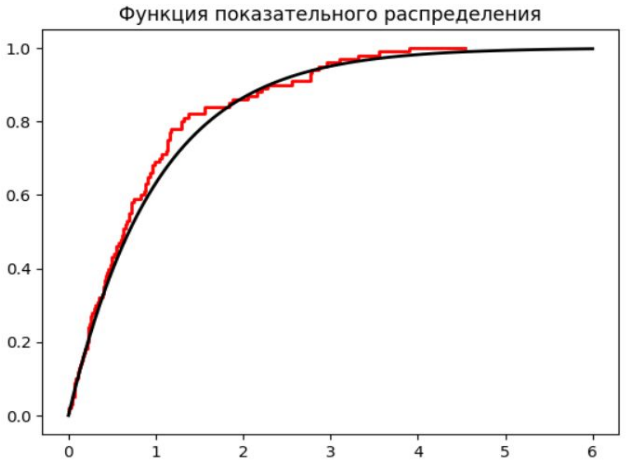
https://github.com/ExuberantSebastian/HW_MS/tree/main

Задание 1.1 графики

Для создания использовались библиотеки matplotlib.pyplot и seaborn





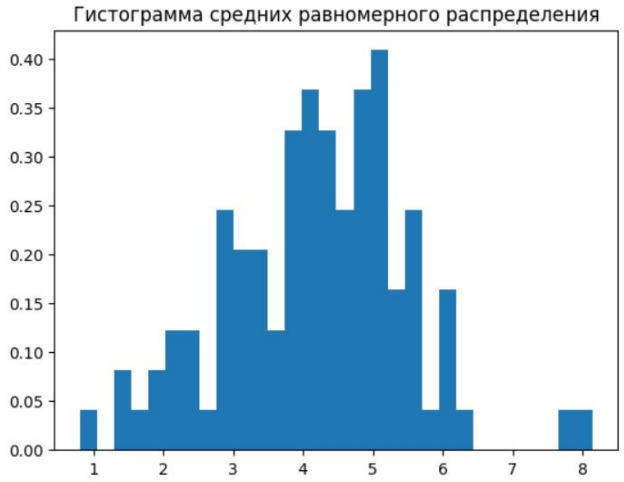


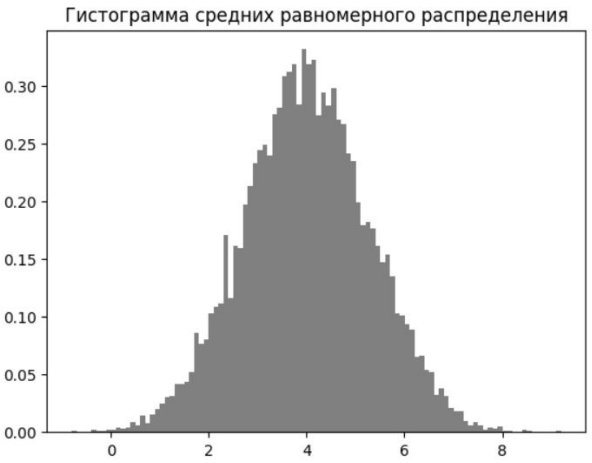
Красная - выборочная, чёрная - теоретическая
Видно, что они пытаются сойтись, на бесконечности сработает асимптотика и они совпадут

https://github.com/ExuberantSebastian/HW_MS/tree/main

Задание 1.2 графики

Для создания использовались библиотеки matplotlib.pyplot и seaborn





Синяя n = 100, Фиолетовая n = 1000, Серая n = 10000
Наглядный пример работы ЦПТ

https://github.com/ExuberantSebastian/HW_MS/tree/main

Задание 1.3 матрица

Сделано в экселе

Было проанализировано 3 столбца
Корреляция между средним по нефти и “Круд Дубай” почти равна единице
С газом у нефти корреляция слабее, но тоже довольно большая

К-Матрица	Crude oil, average	Crude oil, Dubai	Natural gas, US
Crude oil, average	1	0,998654	0,691115
Crude oil, Dubai	0,998654	1	0,663743
Natural gas, US	0,691115	0,663743	1

Хоть мы и можем видеть эту взаимосвязь, чтобы подтвердить зависимость нужно выполнить хотя бы хи-2 тест

https://github.com/ExuberantSebastian/HW_MS/tree/main

Задание 2.1 генерация

На данном этапе было создано 9 случайных выборок из трёх распределений с размерностью в 100, 300 и 1000

Пример кода генерации

```
# Равномерное
unif_rv = sts.uniform(loc=c_3, scale=c_2)
U_1 = unif_rv.rvs(100)
U_2 = unif_rv.rvs(300)
U_3 = unif_rv.rvs(1000)
```

https://github.com/ExuberantSebastian/HW_MS/tree/main

Задание 2.2 одиночные интервалы

В данном пункте было сгенерировано около 90 доверительных интервалов с различными параметрами

После решётки прописан уровень доверия, смысл интервала понятен по заданию и переменной, подробнее в файлике с кодом (см. ipynb файл)

Общие тенденции - с увеличением количества наблюдений в выборке интервал сужается и становится точнее 90% интервал “уже” 95% , что в принципе очевидно

https://github.com/ExuberantSebastian/HW_MS/tree/main

Для создания использовались библиотеки numpy и scipy.stats

Пример интервального кода

```
# 90
(np.array(N_1).mean() - z_crit_1*np.sqrt(a_3)/np.sqrt(100),
 np.array(N_1).mean() + z_crit_1*np.sqrt(a_3)/np.sqrt(100))
```

```
(17.891598874317726, 19.122494619006705)
```

```
# 95
(np.array(N_1).mean() - z_crit_2*np.sqrt(a_3)/np.sqrt(100),
 np.array(N_1).mean() + z_crit_2*np.sqrt(a_3)/np.sqrt(100))
```

```
(17.773695374605698, 19.240398118718733)
```

```
# 90
(np.array(N_2).mean() - z_crit_1*np.sqrt(a_3)/np.sqrt(300),
 np.array(N_2).mean() + z_crit_1*np.sqrt(a_3)/np.sqrt(300))
```

```
(18.624176606313352, 19.334834595853895)
```

```
# 95
(np.array(N_2).mean() - z_crit_2*np.sqrt(a_3)/np.sqrt(300),
 np.array(N_2).mean() + z_crit_2*np.sqrt(a_3)/np.sqrt(300))
```

```
(18.556104989016212, 19.402906213151034)
```

Задание 2.3 разностные интервалы

В данном пункте также было сгенерировано около 90 доверительных интервалов с различными параметрами

После решётки прописаны распределения и объём выборок, для удобства пункт разделён на 90% и 95% интервалы подробнее в файле с кодом
(см. ipynb файл)

Общие тенденции те же - с увеличением количества наблюдений в выборке интервал сужается и становится точнее 90% интервал “уже” 95% , что в принципе очевидно

https://github.com/ExuberantSebastian/HW_MS/tree/main

Для создания использовались библиотеки numpy и scipy.stats

Пример интервального кода

```
# N U 100

(m_N_1 - m_U_1 - z_crit*np.sqrt((var_N + var_U)/100),
 m_N_1 - m_U_1 + z_crit*np.sqrt((var_N + var_U)/100))
```

```
(5.33261659293224, 7.075954525396046)
```

```
# N U 300

(m_N_2 - m_U_2 - z_crit*np.sqrt((var_N + var_U)/300),
 m_N_2 - m_U_2 + z_crit*np.sqrt((var_N + var_U)/300))
```

```
(6.134600113758424, 7.141116738354889)
```

```
# N U 1000

(m_N_3 - m_U_3 - z_crit*np.sqrt((var_N + var_U)/1000),
 m_N_3 - m_U_3 + z_crit*np.sqrt((var_N + var_U)/1000))
```

```
(6.30390808875327, 6.855199948548713)
```

Задание 3.1 гипотезы по симуляции

В данном пункте было сгенерировано 100 нормальных выборок объёмом в 100 с заданными по заданию параметрами

После решётки прописан уровень значимости, цикл проходится по выборке и подбирает попавшие в ДИ значения подробнее в файле с кодом
(см. ipynb файл)

Общие тенденции - с увеличением уровня значимости, интервал расширяется и становится менее точным
Доля верных значений сокращается

https://github.com/ExuberantSebastian/HW_MS/tree/main

Для создания использовались библиотеки numpy и scipy.stats

Пример кода гипотезы

```
# 1% уровень значимости

alpha = 0.01
z_crit = sts.norm.ppf(1 - alpha / 2)
ans_list_1 = []

for i in range(len(N_list)):
    z_obs = ((N_list[i].mean() - a_2) / np.sqrt(a_1)) * np.sqrt(n)
    if (z_obs > -z_crit) and (z_obs < z_crit):
        a = 1
        ans_list_1.append(a)
    else:
        a = 0
        ans_list_1.append(a)

'Доля верных ответов', np.array(ans_list_1).mean()
```

```
('Доля верных ответов', 0.97)
```

Задание 3.2 гипотезы по реальным данным

В данном пункте были подобраны три колонки из предложенных данных (нефть, газ и кофе), переведены в формат DataFrame и обработаны с помощью библиотеки pandas

При работе с данными использовалась гипотеза t-распределения, поскольку нам неизвестны параметры и меняется количество степеней свободы + данные малых объёмов (однако цены не могут быть нормальными в условиях рынка)

Общие тенденции - у стабильных активов вроде нефти и газа гипотезы в основном не имели основания для отвержения, с кофе иная ситуация, из-за провала в цене все гипотезы отверглись, подробнее в файлике (см. ipynb)

https://github.com/ExuberantSebastian/HW_MS/tree/main

Для создания использовались библиотеки numpy, pandas и scipy.stats

Пример кода поиска

```
# Генерация нужных интервалов и значений

Oil_36 = df['Crude oil, Dubai'][733-36:733]
Oil_72 = df['Crude oil, Dubai'][733-72:733]
Oil_108 = df['Crude oil, Dubai'][733-108:733]
last_oil = df['Crude oil, Dubai'][733]
```

Пример кода гипотезы

```
alpha = 0.05
n = 36

t_crit = sts.t.ppf(1 - alpha / 2, 35)
t_obs = ((Oil_36.mean() - last_oil) / Oil_36.var(ddof=1)) * np.sqrt(n)

if (t_obs > -t_crit) and (t_obs < t_crit):
    print('Нет оснований отвергать H0')
else:
    print('H0 отвергается')

t_obs, (-t_crit, t_crit)
```

```
Нет оснований отвергать H0
(-0.08001469462716156, (-2.0301079282503425, 2.0301079282503425))
```

Спасибо за внимание