**A/B – тесты, код**

**Хеширование с солью**

def ab\_split(id, salt='exp\_mess\_1', n\_groups=5):

test\_id = str(id) + '-' + str(salt)

test\_id\_digest = hashlib.md5(test\_id.encode('ascii')).hexdigest()

test\_id\_final\_int = int(test\_id\_digest, 16)

return test\_id\_final\_int % n\_groups

Основная «фишка» в том, что какие бы id'шки к нам на вход ни пришли, разбиение их по соленому хэшу гарантирует, что получившиеся группы будут случайны и приблизительно одинакового размера! Продемонстрируем на примере:

connection = {

'host': 'https://clickhouse.lab.karpov.courses',

'password': 'dpo\_python\_2020',

'user': 'student',

'database': 'simulator'

}

q = """

SELECT distinct user\_id

FROM {db}.feed\_actions

WHERE toDate(time) >= '2021-11-15' and toDate(time) <= '2021-11-21'

"""

# Вытащили пользователей

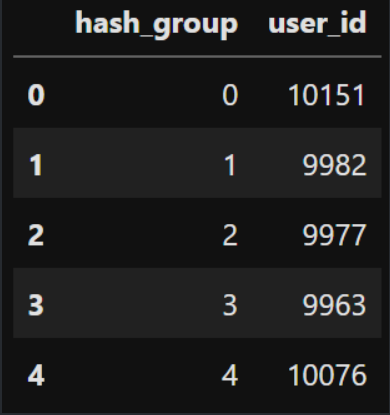
users\_df = pandahouse.read\_clickhouse(q, connection=connection)

# Сделали группы

users\_df['hash\_group'] = users\_df.user\_id.swifter.apply(ab\_split)

# Смотрим на число пользователей в каждой группе

users\_df.groupby('hash\_group').user\_id.nunique().reset\_index()



Хэшировать можно и прямо в кликхаусе:

q = """

SELECT xxHash64(toString(user\_id)||'my\_salt')%5 as group,

uniqExact(user\_id)

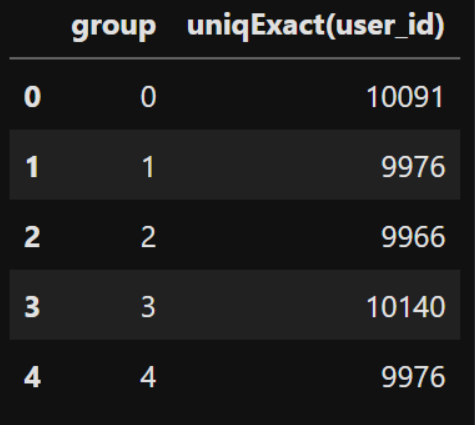
FROM {db}.feed\_actions

WHERE toDate(time) >= '2021-11-15' and toDate(time) <= '2021-11-21'

group by group

"""

pandahouse.read\_clickhouse(q, connection=connection)



**Проверка корректности работы системы сплитования**

# А/А-тест проходил с 8 по 14 ноября 2021.

# А/Б-тест с нашими системами рекомендаций проходил с 15 по 21 ноября 2021.

q = """

SELECT exp\_group,

user\_id,

sum(action = 'like') as likes,

sum(action = 'view') as views,

likes/views as ctr

FROM {db}.feed\_actions

WHERE toDate(time) between '2021-11-08' and '2021-11-14'

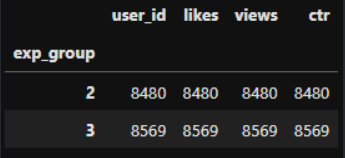
and exp\_group in (2,3)

GROUP BY exp\_group, user\_id

"""

df = pandahouse.read\_clickhouse(q, connection=connection)

df.groupby('exp\_group').count()



# Сделаем графики в seaborn покрупнее

sns.set(rc={'figure.figsize':(11.7,8.27)})

groups = sns.histplot(data = df,

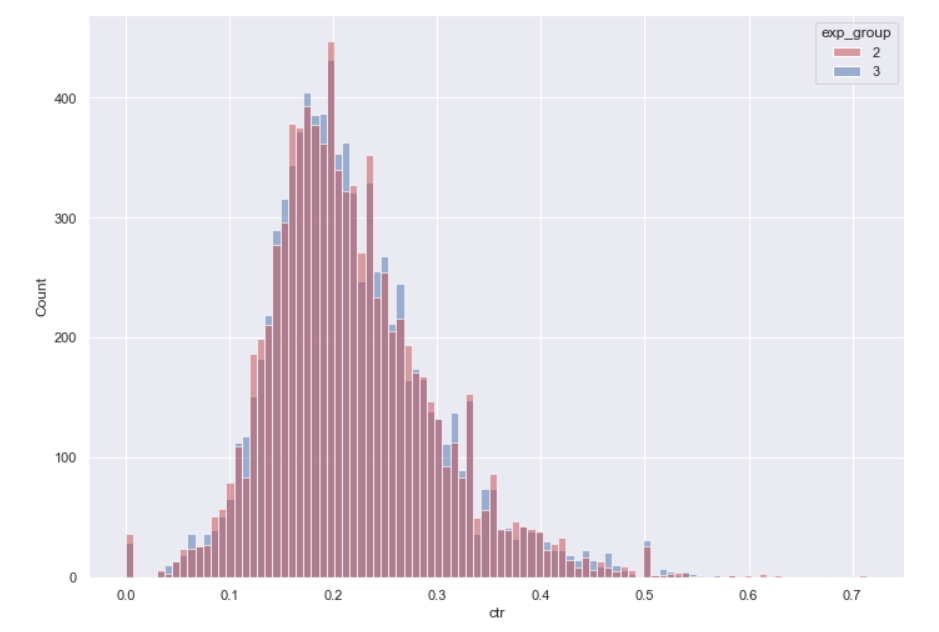
x='ctr',

hue='exp\_group',

palette = ['r', 'b'],

alpha=0.5,

kde=False)



**t-критерий Стьюдента**

stats.ttest\_ind(df[df.exp\_group == 1].ctr,

df[df.exp\_group == 3].ctr,

equal\_var=False)

**тест Манна-Уитни**

print(stats.mannwhitneyu(df[df.exp\_group == 1].ctr,

df[df.exp\_group == 3].ctr))

print('Отличие не прокрасилось')

A\_gt\_B = 0

for \_ in range(10000):

A\_gt\_B+= df[df.exp\_group == 1].ctr.sample().values[0] > df[df.exp\_group == 3].ctr.sample().values[0]

print('В', A\_gt\_B/100, '% случаев A > B. Должно получиться около 50%')

**Полный пример**

# Импорты

import hashlib #хэширование

import pandahouse #вытащить из кликхаоса

import swifter #ускорение

import seaborn as sns #сиборн

import scipy.stats as stats #скайпи статс

# Функция хэширования с солью

def ab\_split(id, salt='exp\_mess\_1', n\_groups=5):

test\_id = str(id) + '-' + str(salt)

test\_id\_digest = hashlib.md5(test\_id.encode('ascii')).hexdigest()

test\_id\_final\_int = int(test\_id\_digest, 16)

return test\_id\_final\_int % n\_groups

# Подключаемся

connection = {

'host': 'https://clickhouse.lab.karpov.courses',

'password': 'dpo\_python\_2020',

'user': 'student',

'database': 'simulator'

}

#Выбираем пользователей

q = """

SELECT distinct user\_id

FROM simulator\_20250520.feed\_actions

WHERE toDate(time) >= '2025-04-25' and toDate(time) <= '2025-05-01'

"""

# Вытащили пользователей

users\_df = pandahouse.read\_clickhouse(q, connection=connection)

# Сделали группы

users\_df['hash\_group'] = users\_df.user\_id.swifter.apply(ab\_split)

# Смотрим на число пользователей в каждой группе

users\_df.groupby('hash\_group').user\_id.nunique().reset\_index()

****

# Вытаскиваем по группам лайки, просмотры и CTR

q = """

SELECT exp\_group,

user\_id,

sum(action = 'like') as likes,

sum(action = 'view') as views,

likes/views as ctr

FROM simulator\_20250520.feed\_actions

WHERE toDate(time) between '2025-04-25' and '2025-05-01'

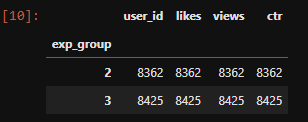
and exp\_group in (2,3)

GROUP BY exp\_group, user\_id

"""

df = pandahouse.read\_clickhouse(q, connection=connection)

df.groupby('exp\_group').count()



# Сделаем графики в seaborn покрупнее

sns.set(rc={'figure.figsize':(11.7,8.27)})

groups = sns.histplot(data = df,

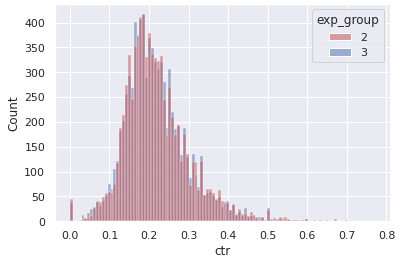
x='ctr',

hue='exp\_group',

palette = ['r', 'b'],

alpha=0.5,

kde=False)



**A/A-тест «по классике»**

**t-критерий Стьюдента**

stats.ttest\_ind(df[df.exp\_group == 2].ctr,

df[df.exp\_group == 3].ctr,

equal\_var=False)



**тест Манна-Уитни**

print(stats.mannwhitneyu(df[df.exp\_group == 2].ctr,

df[df.exp\_group == 3].ctr))

print('Отличие не прокрасилось')

A\_gt\_B = 0

for \_ in range(10000):

A\_gt\_B+= df[df.exp\_group == 2].ctr.sample().values[0] > df[df.exp\_group == 3].ctr.sample().values[0]

print('В', A\_gt\_B/100, '% случаев A > B. Должно получиться около 50%')



**A/B-тест**

# Импорты

import hashlib

import pandahouse

import swifter

import seaborn as sn

import scipy.stats as stats

# Подключение

connection = {

'host': 'https://clickhouse.lab.karpov.courses',

'password': 'dpo\_python\_2020',

'user': 'student',

'database': 'simulator'

}

# Выбор данных

q = """

SELECT exp\_group,

user\_id,

sum(action = 'like') as likes,

sum(action = 'view') as views,

likes/views as ctr

FROM simulator\_20250520.feed\_actions

WHERE toDate(time) between '2025-05-02' and '2025-05-08'

and exp\_group in (1,2)

GROUP BY exp\_group, user\_id

"""

df = pandahouse.read\_clickhouse(q, connection=connection)

#T-тест

stats.ttest\_ind(df[df.exp\_group == 1].ctr,

df[df.exp\_group == 2].ctr,

equal\_var=False)



# Строим график

groups = sns.histplot(data = df,

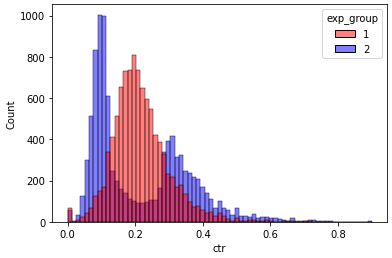
x='ctr',

hue='exp\_group',

palette = ['r', 'b'],

alpha=0.5,

kde=False)



# Манна-Уитни

stats.mannwhitneyu(df[df.exp\_group == 1].ctr,

df[df.exp\_group == 2].ctr,

alternative = 'two-sided')



**В Кликхаусе:**

--Для t-теста

WITH welchTTest(ctr, exp\_index) as ttest

select tupleElement(ttest,1) as statistic,

tupleElement(ttest,2) as p\_value

FROM (SELECT exp\_group,

case when exp\_group = 1 then 0 else 1 end as exp\_index,

user\_id,

sum(action = 'like') as likes,

sum(action = 'view') as views,

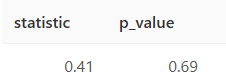
likes/views as ctr

FROM simulator\_20250520.feed\_actions

WHERE toDate(time) between '2025-05-02' and '2025-05-08'

and exp\_group in (1,2)

GROUP BY exp\_group, user\_id)



-- То есть t-тест говорит: «обе группы имеют очень близкие средние CTR → p≈0.685 → формально нет разницы»

--Для Манна-Уитни

WITH mannWhitneyUTest('two-sided')(ctr, exp\_index) as mw

select tupleElement(mw,1) as statistic,

tupleElement(mw,2) as p\_value

FROM (SELECT exp\_group,

case when exp\_group = 1 then 0 else 1 end as exp\_index,

user\_id,

sum(action = 'like') as likes,

sum(action = 'view') as views,

likes/views as ctr

FROM simulator\_20250520.feed\_actions

WHERE toDate(time) between '2025-05-02' and '2025-05-08'

and exp\_group in (1,2)

GROUP BY exp\_group, user\_id)



“Распределения сдвинуты (одна кривая лежит левее/правее другой) → p≈4·10⁻⁴⁵ или NaN или 0 → распределения не совпадают”.

**Критический момент**: если почти все (или все) значения в выборке одинаковы, либо в обеих группах одинаковые ранги, то алгоритм не может корректно “разложить” ранги и часто сводится к делению на ноль или бесконечным ранговым суммам. В ClickHouse в этом случае MannWhitneyUTest возвращает NaN.

Когда именно появляется NaN? Если все значения в объединённой выборке ровно одинаковы (ранги полностью совпадают). Или если слишком большой процент “связок” (ties), и ClickHouse не может разложить U-статистику без деления на ноль.

# Смотрим

A\_gt\_B = 0

for \_ in range(10000):

A\_gt\_B+= df[df.exp\_group == 1].ctr.sample().values[0] > df[df.exp\_group == 2].ctr.sample().values[0]

print('В', A\_gt\_B/100, '% случаев A > B. Должно получиться около 50%')



Итог: t-тест - p примерно 0.685, значит согласно t-тесту средние не различаются. Распределение по CTR сдвинуто, то есть в группе 2 пользователей с низким CTR больше, хотя среднее получилось почти таким же благодаря «хвостам». Если взять по одному пользователю из каждой группы, примерно в 54.5 % случаев CTR пользователя из группы 1 будет выше, чем у пользователя из группы 2, следовательно распределения отличаются.

**Сглаженный CTR**

# Импорты

import pandahouse

import seaborn as sns

# Подключение, взятие данных

connection = {

'host': 'https://clickhouse.lab.karpov.courses',

'password': 'dpo\_python\_2020',

'user': 'student',

'database': 'simulator'

}

q = """

SELECT exp\_group,

user\_id,

sum(action = 'like') as likes,

sum(action = 'view') as views,

likes/views as ctr

FROM simulator\_20250520.feed\_actions

WHERE toDate(time) between '2025-04-25' and '2025-05-01'

and exp\_group in (1,2)

GROUP BY exp\_group, user\_id

"""

df = pandahouse.read\_clickhouse(q, connection=connection)

# Глобальные CTR

global\_ctr\_1 = df[df.exp\_group == 1].likes.sum()/df[df.exp\_group == 1].views.sum()

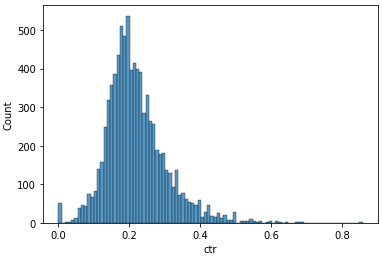
global\_ctr\_2 = df[df.exp\_group == 2].likes.sum()/df[df.exp\_group == 2].views.sum()

# График без сглаженного

group1 = df[df.exp\_group == 1].copy()

sns.histplot(group1.ctr,

kde = False)



# Применение сглаживания

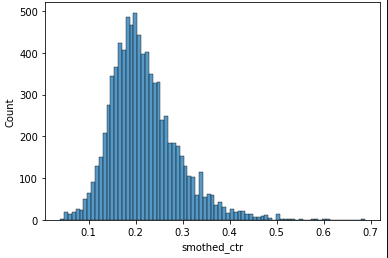
group1['smothed\_ctr'] = df.apply(

lambda x: get\_smothed\_ctr(x['likes'], x['views'], global\_ctr\_1, 5), axis=1)

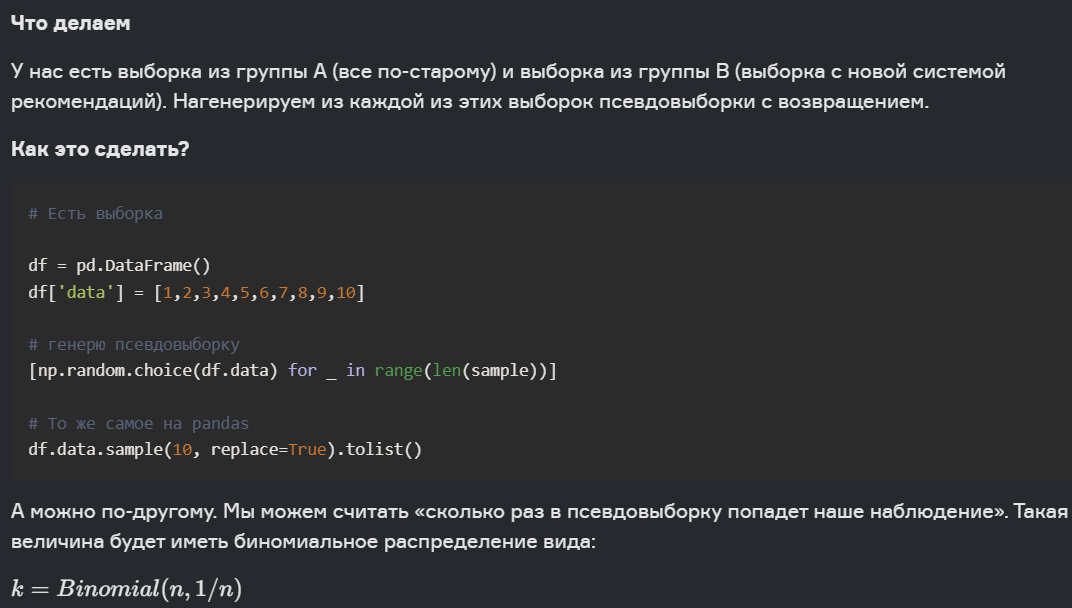
# Сглаженный график

sns.histplot(group1.smothed\_ctr,

kde = False)



**Бутстреп и CTR**



df = pd.DataFrame()

df['data'] = np.random.randint(1,100000,10000)

sample\_length = 10000

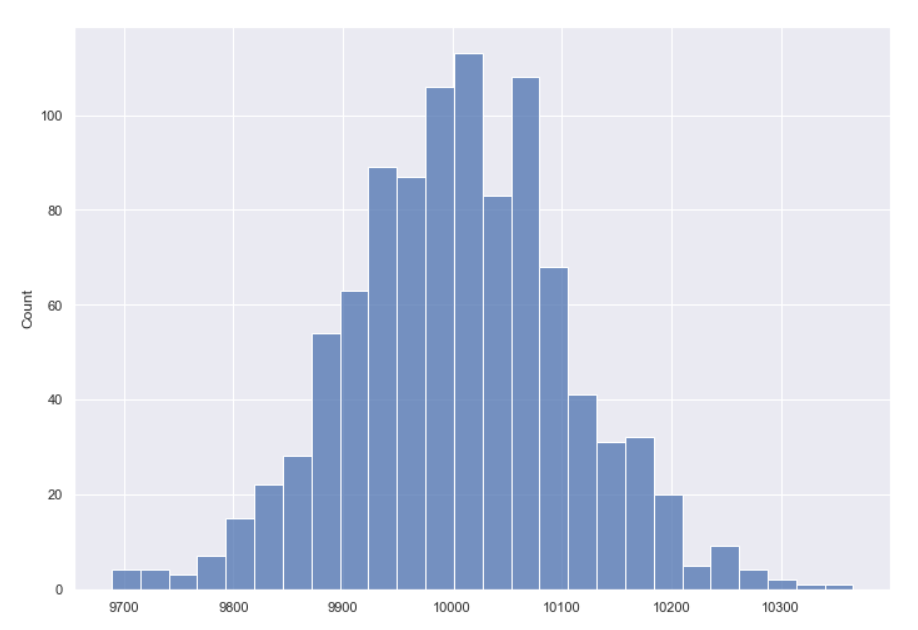
p\_to\_be\_taken = 1/sample\_length

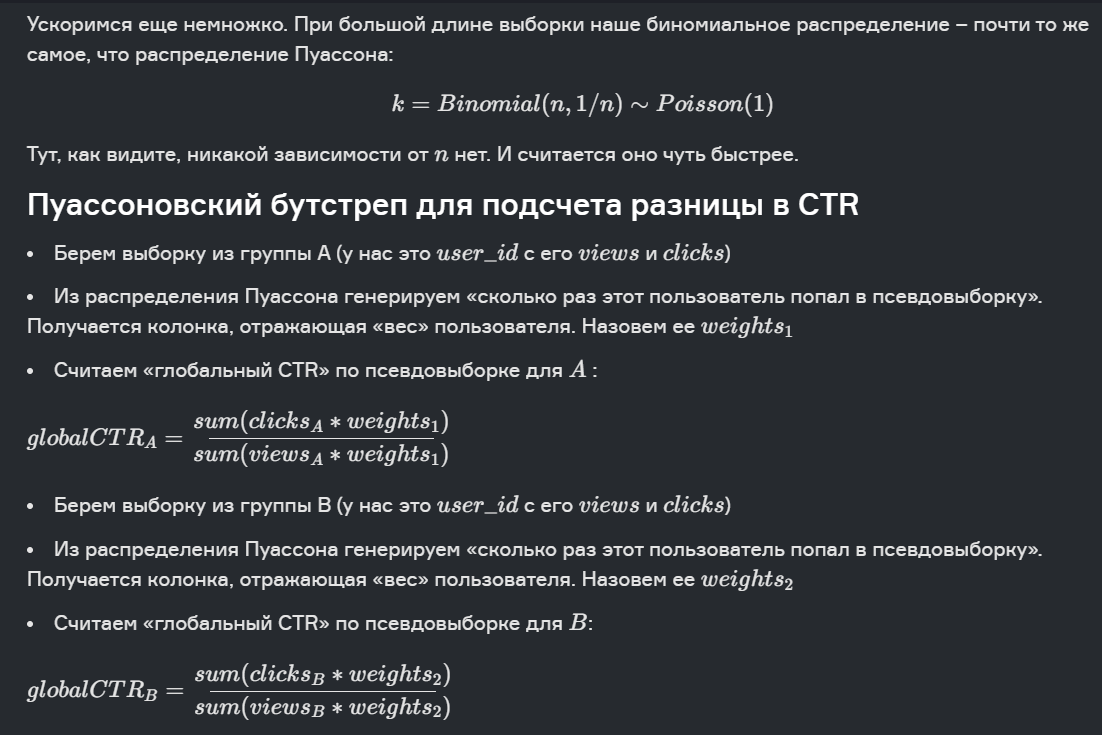
number\_of\_samples = 1000

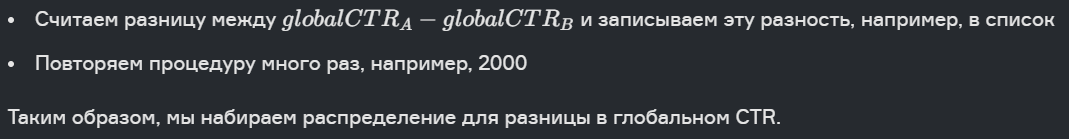
sums = [sum(stats.binom(sample\_length,

p\_to\_be\_taken).rvs(sample\_length)) for \_ in range(number\_of\_samples)]

sns.histplot(sums)







**Пример**

# Импорты

import pandahouse

import seaborn as sns

import scipy.stats as stats

import numpy as np

# Функция Бутстреп

def bootstrap(likes1, views1, likes2, views2, n\_bootstrap=2000):

poisson\_bootstraps1 = stats.poisson(1).rvs(

(n\_bootstrap, len(likes1))).astype(np.int64)

poisson\_bootstraps2 = stats.poisson(1).rvs(

(n\_bootstrap, len(likes2))).astype(np.int64)

globalCTR1 = (poisson\_bootstraps1\*likes1).sum(axis=1)/(poisson\_bootstraps1\*views1).sum(axis=1)

globalCTR2 = (poisson\_bootstraps2\*likes2).sum(axis=1)/(poisson\_bootstraps2\*views2).sum(axis=1)

return globalCTR1, globalCTR2

# Получение данных

connection = {

'host': 'https://clickhouse.lab.karpov.courses',

'password': 'dpo\_python\_2020',

'user': 'student',

'database': 'simulator'

}

q = """

SELECT exp\_group,

user\_id,

sum(action = 'like') as likes,

sum(action = 'view') as views,

likes/views as ctr

FROM simulator\_20250520.feed\_actions

WHERE toDate(time) between '2025-05-02' and '2025-05-08'

and exp\_group in (1,2)

GROUP BY exp\_group, user\_id

"""

df = pandahouse.read\_clickhouse(q, connection=connection)

# Выносится из датафрейма по-отдельности информация о лайках и просмотрах для двух групп эксперимента, и всё это конвертируется в NumPy-массивы

likes1 = df[df.exp\_group == 1].likes.to\_numpy()

views1 = df[df.exp\_group == 1].views.to\_numpy()

likes2 = df[df.exp\_group == 2].likes.to\_numpy()

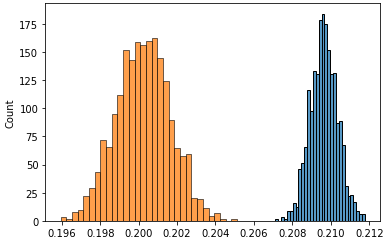
views2 = df[df.exp\_group == 2].views.to\_numpy()

# Бутстреп анализ и графики

ctr1, ctr2 = bootstrap(likes1, views1, likes2, views2)

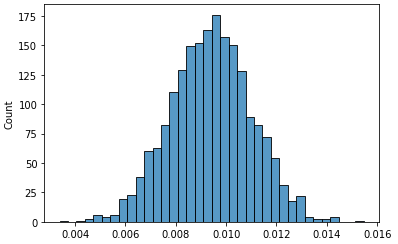
sns.histplot(ctr1)

sns.histplot(ctr2)



# Разница между глобальными CTR

sns.histplot(ctr1 – ctr2)

 Итог: Поскольку гистограмма разности не пересекает 0, на уровне, на котором мы сравниваем группы, 1 группа строго больше другой.

**Бакетное преобразование**

# Импорт

import pandahouse

import seaborn as sns

import scipy.stats as stats

# Подключение и считывание данных

connection = {

'host': 'https://clickhouse.lab.karpov.courses',

'password': 'dpo\_python\_2020',

'user': 'student',

'database': 'simulator'

}

q = """

SELECT exp\_group, bucket,

sum(likes)/sum(views) as bucket\_ctr,

quantileExact(0.9)(ctr) as ctr9

FROM (SELECT exp\_group,

xxHash64(user\_id)%50 as bucket,

user\_id,

sum(action = 'like') as likes,

sum(action = 'view') as views,

likes/views as ctr

FROM simulator\_20250520.feed\_actions

WHERE toDate(time) between '2025-05-02' and '2025-05-08'

and exp\_group in (1,2)

GROUP BY exp\_group, bucket, user\_id)

GROUP BY exp\_group, bucket

"""

df = pandahouse.read\_clickhouse(q, connection=connection)

#тест Манна-Уитни видит отличие

stats.mannwhitneyu(df[df.exp\_group == 1].bucket\_ctr,

df[df.exp\_group == 2].bucket\_ctr,

alternative = 'two-sided')



#и t-тест тоже

stats.ttest\_ind(df[df.exp\_group == 1].bucket\_ctr,

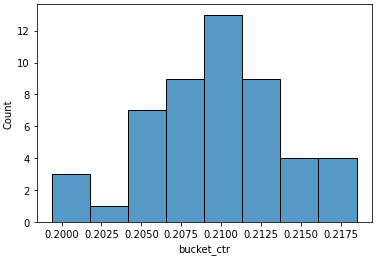
df[df.exp\_group == 2].bucket\_ctr,

equal\_var = False)



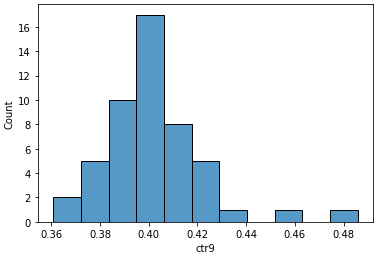
# Гистограмма по бакетам

sns.histplot(df[df.exp\_group == 1].bucket\_ctr)



# Гистограмма по квантилям (А вот с квантилями пользовательских CTR стоит быть осторожнее, t-тест тут может работать не очень хорошо)

sns.histplot(df[df.exp\_group == 2].ctr9)



**Задания**

Задание 1

В лекции мы выяснили, что при проведении А/А-теста было бы здорово убедиться в том, что наша система сплитования работает корректно, и ключевая метрика не отличается между группами не только в конкретно нашем А/А-тесте, но и в целом.

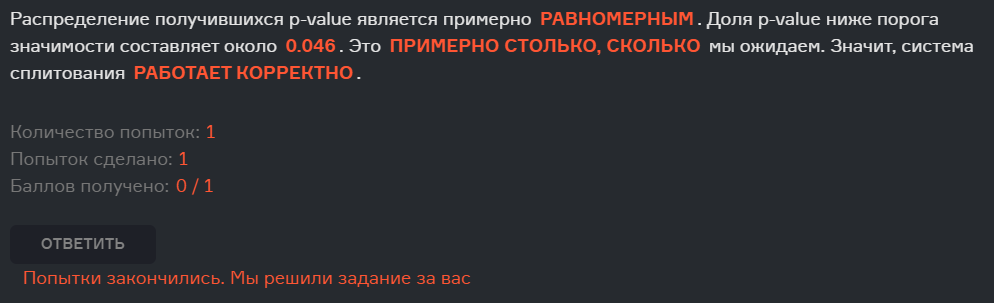
В идеале было бы здорово провести бесконечное количество А/A-тестов и посмотреть, в каком количестве случаев нам удалось отклонить нулевую гипотезу. Если система сплитования работает корректно, то статистически значимые различия между двумя группами встречались бы только в результате случайного ложного срабатывания. Например, если мы отвергаем нулевую гипотезу при условии, что p\_value < 0.05, то только приблизительно в 5% случаев у нас бы получались статистические значимые различия между 0 и 1 группой.

Понятное дело, что на практике провести бесконечное число тестов у нас вряд ли получится, поэтому используется небольшой трюк. Мы будем многократно извлекать подвыборки из наших данных, проводить t-test, а в конце посмотрим, в каком проценте случаев нам удалось отклонить нулевую гипотезу.

Сделаем следующее:

1. Берём данные АА-теста из следующего диапазона: с '2025-04-25' по '2025-05-01'
2. Из групп 2 и 3 берём подвыборки **без возвращения**размером в 500 юзеров
3. Сравниваем их t-тестом и сохраняем p-value (здесь и далее используем аргумент equal\_var=False)
4. Повторяем это 10000 раз

Нарисуйте гистограмму получившихся p-value и посчитайте долю p-value, оказавшихся ниже порога значимости в 0.05. Что мы можем сказать по этому результату?



**Верно сделанное**

import hashlib

import pandahouse

import swifter

import scipy.stats as stats

import numpy as np

import pandas as pd

import random

import matplotlib.pyplot as plt

import seaborn as sns

connection = {

'host': 'https://clickhouse.lab.karpov.courses',

'password': 'dpo\_python\_2020',

'user': 'student',

'database': 'simulator'

}

def ab\_split(id, salt='exp\_mess\_1', n\_groups=5):

test\_id = str(id) + '-' + str(salt)

test\_id\_digest = hashlib.md5(test\_id.encode('ascii')).hexdigest()

test\_id\_final\_int = int(test\_id\_digest, 16)

return test\_id\_final\_int % n\_groups

query\_users = """

SELECT DISTINCT user\_id

FROM simulator\_20250520.feed\_actions

WHERE toDate(time) >= '2025-04-25'

AND toDate(time) <= '2025-05-01'

"""

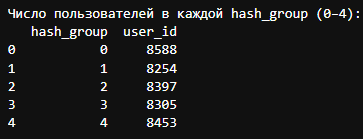
users\_df = pandahouse.read\_clickhouse(query\_users, connection=connection)

users\_df['hash\_group'] = users\_df['user\_id'].swifter.apply(ab\_split)

counts\_by\_group = users\_df.groupby('hash\_group')['user\_id'].nunique().reset\_index()

print("Число пользователей в каждой hash\_group (0–4):")

print(counts\_by\_group)



query\_ctr = """

SELECT

user\_id,

SUM(if(action = 'view', 1, 0)) AS views,

SUM(if(action = 'like', 1, 0)) AS likes,

CASE

WHEN SUM(if(action = 'view', 1, 0)) = 0 THEN 0.0

ELSE toFloat64(SUM(if(action = 'like', 1, 0)))

/ toFloat64(SUM(if(action = 'view', 1, 0)))

END AS ctr

FROM simulator\_20250520.feed\_actions

WHERE toDate(time) BETWEEN '2025-04-25' AND '2025-05-01'

GROUP BY user\_id

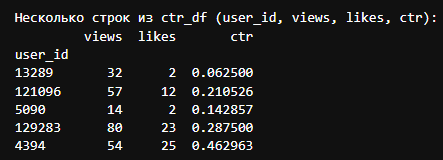
"""

ctr\_df = pandahouse.read\_clickhouse(query\_ctr, connection=connection)

ctr\_df = ctr\_df.set\_index('user\_id')

print("\nНесколько строк из ctr\_df (user\_id, views, likes, ctr):")

print(ctr\_df.head())



group2\_ids\_all = users\_df.loc[users\_df['hash\_group'] == 2, 'user\_id'].values

group3\_ids\_all = users\_df.loc[users\_df['hash\_group'] == 3, 'user\_id'].values

print(f"\nВсего в группе 2 пользователей: {len(group2\_ids\_all)}")

print(f"Всего в группе 3 пользователей: {len(group3\_ids\_all)}")



n\_iterations = 10000

all\_pvalues = []

for i in range(n\_iterations):

sampled\_2 = np.random.choice(group2\_ids\_all, size=500, replace=False)

sampled\_3 = np.random.choice(group3\_ids\_all, size=500, replace=False)

tmp2 = pd.DataFrame({'user\_id': sampled\_2})

tmp3 = pd.DataFrame({'user\_id': sampled\_3})

ctr\_group2 = (tmp2.join(ctr\_df[['ctr']], on='user\_id', how='left').fillna(0.0)['ctr'])

ctr\_group3 = (tmp3.join(ctr\_df[['ctr']], on='user\_id', how='left').fillna(0.0)['ctr'])

assert len(ctr\_group2) == 500

assert len(ctr\_group3) == 500

t\_stat, p\_val = stats.ttest\_ind(ctr\_group2, ctr\_group3, equal\_var=False)

all\_pvalues.append(p\_val)

all\_pvalues = np.array(all\_pvalues)

share\_below\_005 = np.mean(all\_pvalues < 0.05)

print(f"\nИз {n\_iterations} A/A-тестов доля случаев, когда p-value < 0.05: {share\_below\_005:.4f}")



plt.figure(figsize=(6, 4))

plt.hist(all\_pvalues, bins=50, edgecolor='k')

plt.axvline(0.05, color='red', linestyle='--', label='p = 0.05')

plt.title("Распределение p-value в 10000 A/A-тестах")

plt.xlabel("p-value")

plt.ylabel("Количество итераций")

plt.legend()

plt.tight\_layout()

plt.show()

