In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import pandahouse
from scipy import stats
from tqdm import tqdm_notebook
import ipywidgets
```

A/B тест для ленты новостей с новым алгоритмом рекомендаций

Экспериментальные группы:

exp_group = 1 - всё по-старому

exp_group = 2 - рекомендации «похожих на лайкнутые постов»

Загружаем данные

In [4]:
```
connection = {
    'host': 'https://clickhouse.lab.karpov.courses',
    'password': 
    'user': 
    'database': 'simulator_20231220'
}
```

In [44]:
```
query = '''SELECT
    exp_group,
    user_id,
    sum(action = 'like') AS likes,
    sum(action = 'view') AS views,
    likes/views AS ctr
FROM simulator_20231220.feed_actions
WHERE
    toDate(time) BETWEEN '2023-11-18' and '2023-11-24'
AND
    exp_group IN (1,2)
GROUP BY
    exp_group,
    user_id'''
```

In [45]:
```
df = pandahouse.read_clickhouse(query, connection = connection)
```

In [46]:
```
df.groupby('exp_group').count()
```

Out[46]:

exp_group	user_id	likes	views	ctr
1	10020	10020	10020	10020
2	9877	9877	9877	9877

Проводим статистические тесты

T-тест

In [20]:
```
# t-test
stats.ttest_ind(
    df[df.exp_group == 1].ctr,
    df[df.exp_group == 2].ctr,
    equal_var = False
)
```

Out[20]:
```
Ttest_indResult(statistic=0.4051491913112757, pvalue=0.685373331140751)
```

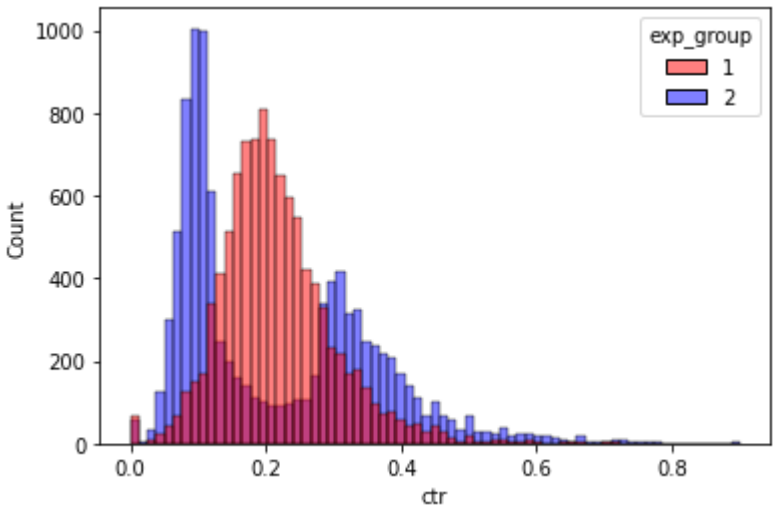
Тест Манна-Уитни

In [21]:
```
stats.mannwhitneyu(
    df[df.exp_group == 1].ctr,
    df[df.exp_group == 2].ctr,
    alternative = 'two-sided'
)
```

Out[21]:
```
MannwhitneyuResult(statistic=55189913.0, pvalue=4.632205841806026e-45)
```

Визуализация распределений значений CTR

In [22]:
```
plot = sns.histplot(
    data = df,
    x = 'ctr',
    hue = 'exp_group',
    palette = ['r', 'b'],
    alpha=0.5,
    kde=False)
```



Промежуточные выводы

Итак, результаты наших тестов разошлись — тест Манна-Уитни показал различие между нашими группами, в то время как Т-тест не дал нам достаточных оснований для отклонения гипотезы о равенстве средних. Объясняется данное различие формами распределений.

Из-за того, что пик относительно нормального распределения значений CTR 1 группы лежит между горбами распределения CTR 2 группы T-тест видит сходство в их средних.

Критерий Манна-Уитни же даёт нам более интересную информацию: полученный p-уровень значимости показывает нам, что вероятности того, что случайно выбранное значение из одного распределения окажется больше значения из другого неравны. Сопоставив этот вывод с графиком распределений можно утверждать, что в большинстве случаев CTR случайного пользователя из 2 группы окажется меньше, чем CTR пользователя из 1 группы, что уже указывает нам на то, что новый алгоритм рекомендаций **возможно** оказывает негативное влияние на CTR.

Проведём исследование другими методами.

Сравниваем распределения при помощи сглаженного CTR

Расчёт распределения

In [23]:
```
def smoothed_ctr(likes, views, global_ctr, alpha):
    smoothed_ctr = (likes + alpha * global_ctr) / (views + alpha)
    return smoothed_ctr
```

In [24]:
```
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()
```

In [25]:
```
print(smoothed_ctr(df['likes'][1], df['views'][1], global_ctr_1, 5))
```

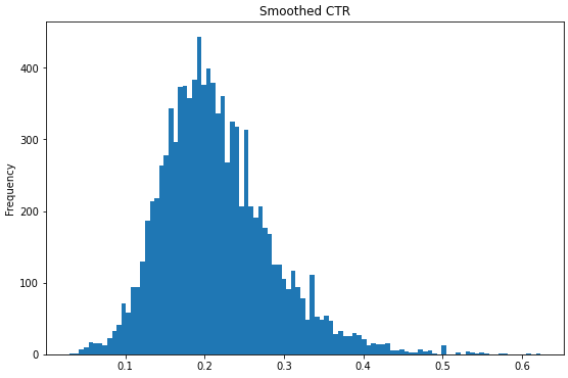
```
0.22635630694655579
```

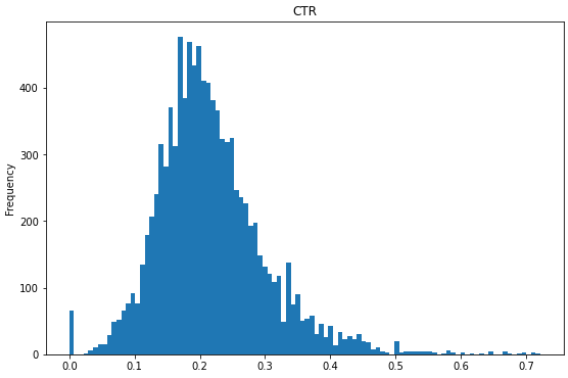
In [26]:
```
group1 = df[df.exp_group == 1].copy()
group2 = df[df.exp_group == 2].copy()
```

In [27]:
```
group1['smoothed_ctr'] = df.apply(
    lambda x: smoothed_ctr(x['likes'], x['views'], global_ctr_1, 5), axis = 1)

group2['smoothed_ctr'] = df.apply(
    lambda x: smoothed_ctr(x['likes'], x['views'], global_ctr_2, 5), axis = 1)
```

Визуализация распределений CTR

In [30]:
```
fig, axes = plt.subplots(nrows=1,ncols=2, figsize = (20, 6))
axes[0].set_title('Smoothed CTR')
axes[1].set_title('CTR')


group1['smoothed_ctr'].plot(kind = 'hist', ax = axes[0], bins = 100, subplots=True)
group1['ctr'].plot(kind = 'hist', bins = 100, ax = axes[1], subplots=True)

fig.show()
```

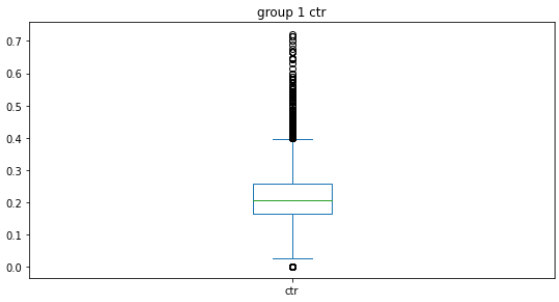


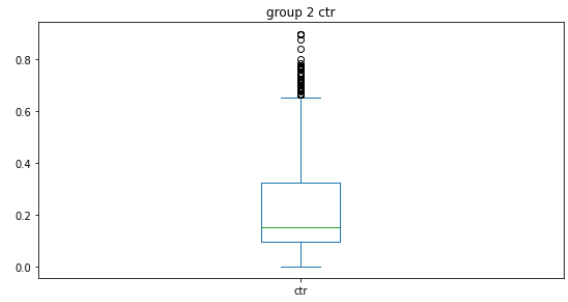


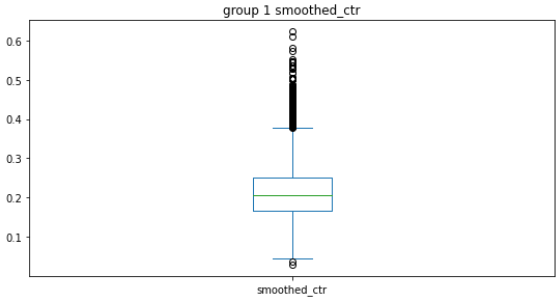
In [31]:
```
stats.ttest_ind(
    group1.smoothed_ctr,
    group2.smoothed_ctr,
    equal_var = False
)
```

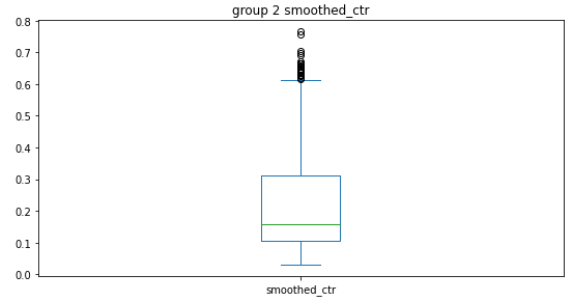
Out[31]:
```
Ttest_indResult(statistic=1.9460491517027683, pvalue=0.05166679015318526)
```

In [82]:
```
fig, axes = plt.subplots(nrows=2,ncols=2, figsize = (20, 10))
axes[0,0].set_title('group 1 ctr')
axes[0,1].set_title('group 2 ctr')
axes[1,0].set_title('group 1 smoothed_ctr')
axes[1,1].set_title('group 2 smoothed_ctr')

df.loc[df['exp_group'] == 1].ctr.plot(kind = 'box', ax = axes[0,0], subplots=True)
df.loc[df['exp_group'] == 2].ctr.plot(kind = 'box', ax = axes[0,1], subplots=True)
group1['smoothed_ctr'].plot(kind = 'box', ax = axes[1,0], subplots=True)
group2['smoothed_ctr'].plot(kind = 'box', ax = axes[1,1], subplots=True)

fig.show()
```









Вывод по сглаженному CTR

Сгладив CTR, мы смогли повысить чувствительность нашей метрики, однако T-тест всё ещё не позволяет нам отклонить гипотезу о расвенстве средних. Причины этого видны на боксплотах — сглаженный CTR едва ли как-то повлиял на наши распределения.

Сравниваем распределения при помощи бутстрепа

Строим распределения

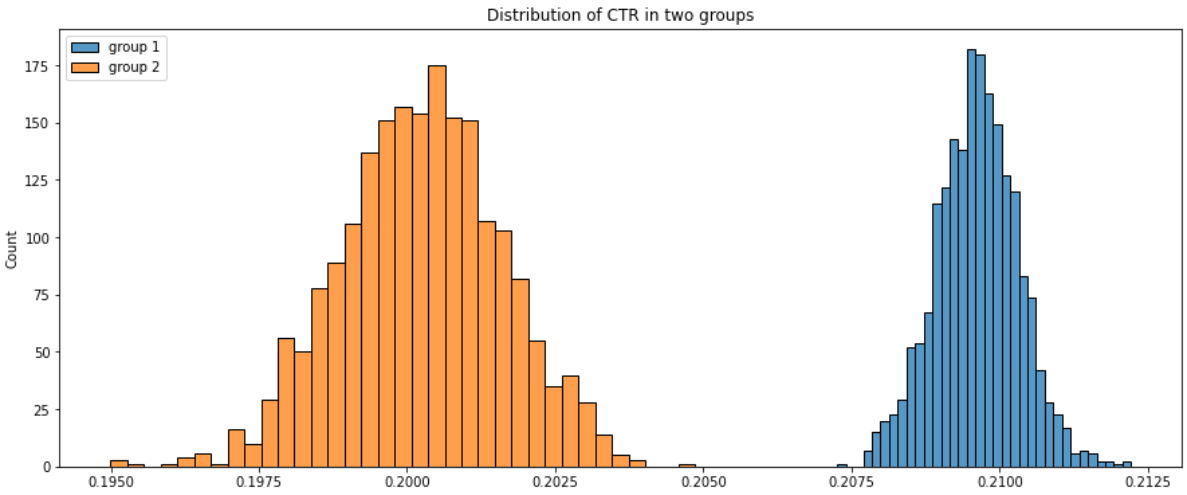
In [83]:
```
poisson_bootstraps1 = stats.poisson(1).rvs(
        (2000, len(df[df.exp_group == 1].likes.to_numpy()))).astype(np.int64)
```

In [84]:
```
len(poisson_bootstraps1[620])
```

Out[84]:
```
10020
```

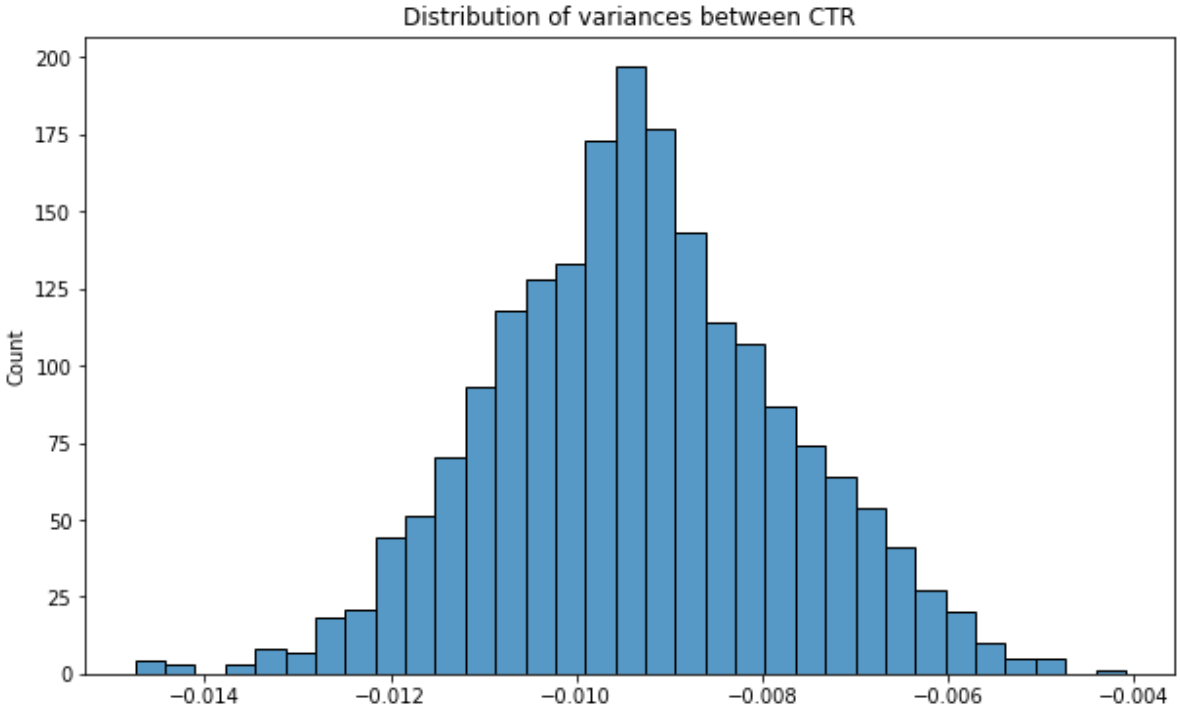
In [85]:
```
def bootstrap(likes1, views1, likes2, views2, n_bootstrap=2000):

    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*viev

    globalCTR2 = (poisson_bootstraps2*likes2).sum(axis=1)/(poisson_bootstraps2*viev

    return globalCTR1, globalCTR2
```

In [117…
```
likes_1 = df[df.exp_group == 1].likes.to_numpy()
views_1 = df[df.exp_group == 1].views.to_numpy()
likes_2 = df[df.exp_group == 2].likes.to_numpy()
views_2 = df[df.exp_group == 2].views.to_numpy()

ctr1, ctr2 = bootstrap(likes_1, views_1, likes_2, views_2)

plt.figure(figsize=(15,6))
sns.histplot(ctr1, label = 'group 1').set(title = 'Distribution of CTR in two group
sns.histplot(ctr2, label = 'group 2')

plt.legend(loc="upper left")


plt.show()
```



In [103…

```
plt.figure(figsize=(10,6))
sns.histplot(ctr2-ctr1).set(title = 'Distribution of variances between CTR')

plt.show()
```



Выводы по бутстрепу

Применение бутстрепа даёт нам вполне явную интерпретацию эффекта от внедрения новой системы рекомендаций — CTR становится меньше.

Такие значения распределения прежде всего обусловлены тем, что при генерации псевдовыборок из 2 распределения, в них попадает большое количество низких значений CTR (это видно по форме распределения). Более того, наши наблюдения сходятся с результатом, полученным после применения теста Манна-Уитни — вероятность «достать» из распределения CTR 1 группы значение, большее значения этой метрики в 2 группе, выше, чем вероятность обратного события.

Сравниваем распределения при помощи бакетного преобразования

In [92]:

```
query2 = """

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 {db}.feed_actions
    WHERE toDate(time) between '2023-11-18' and '2023-11-24'
        and exp_group in (1,2)
```

```
    GROUP BY exp_group, bucket, user_id)
GROUP BY exp_group, bucket
"""

df2 = pandahouse.read_clickhouse(query2, connection=connection)
```

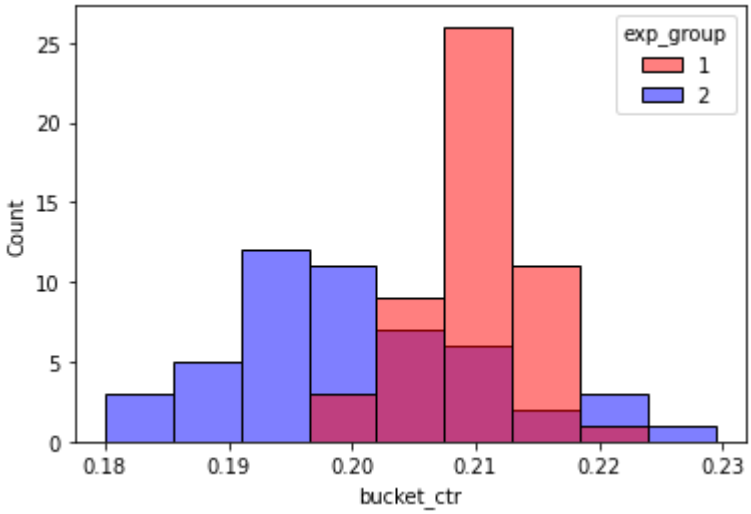
In [49]:
```
stats.mannwhitneyu(
    df2[df2.exp_group == 1].bucket_ctr,
    df2[df2.exp_group == 2].bucket_ctr,
    alternative = 'two-sided'
)
```

Out[49]:
```
MannwhitneyuResult(statistic=1997.0, pvalue=2.6576427804010095e-07)
```

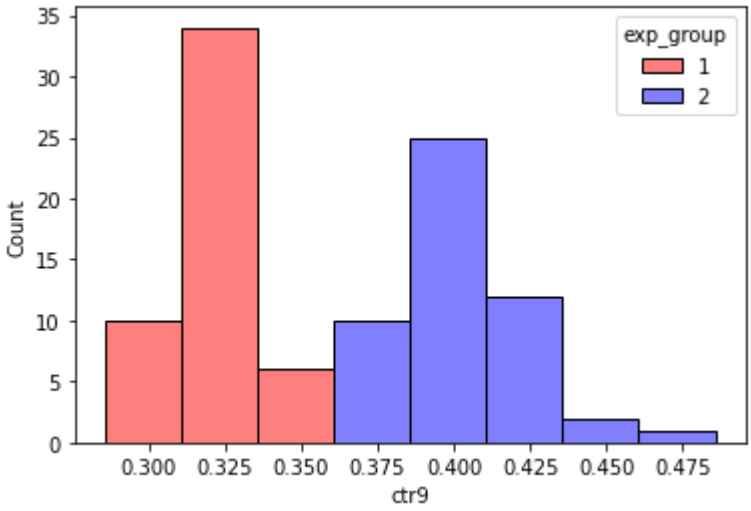
In [50]:
```
stats.ttest_ind(
    df2[df2.exp_group == 1].bucket_ctr,
    df2[df2.exp_group == 2].bucket_ctr,
    equal_var = False
)
```

Out[50]:
```
Ttest_indResult(statistic=5.614819358149381, pvalue=4.592644937473873e-07)
```

In [51]:
```
plot = sns.histplot(
    data = df2,
    x = 'bucket_ctr',
    hue = 'exp_group',
    palette = ['r', 'b'],
    alpha=0.5,
    kde=False)
```



In [93]:
```
plot = sns.histplot(
    data = df2,
    x = 'ctr9',
    hue = 'exp_group',
    palette = ['r', 'b'],
    alpha=0.5,
    kde=False)
```



Выводы по бакетному преобразованию

Распределение пользователей по бакетам позволило нам уйти от исходной формы распределения данных. Теперь на графике видно чёткое смещение распределения тестовой группы, а T-тест видит различия в средних благодаря тому, что мы изменили представление наших данных.

Итог

В результате всех проведённых тестов можно утверждать, что у нас недостаточно обоснования для развёртывания новой системы на всех пользователей. Несмотря на то, что у части пользователей приложения CTR увеличился, другая, достоточно большая часть аудитории стала менее положительно воспринимать новости в ленте.

В нашем случае самая интресная часть результатов для анализа — двугорбая форма распределения. Напомню, что новая система рекомендаций подразумевает выдачу новостей похожих на те, что уже лайкнул пользователь. Причины того, почему CTR значительной части аудитории падает могут быть самыми разными, например:

1. Нашим пользователям рекомендуют посты, которые они уже лайкнули. Из-за этого активные пользователи видят больше постов, но не могут их лайкнуть, поэтому их CTR снижается.
2. Система возможно рекомендует пользователям те посты, которые они уже видили, но ещё не лайкнули. Таким образом, часть пользователей всё-таки оставляет реакцию под постом, а у других пользователей просто растёт количество просмотров.
3. Наши алгоритмы подбора похожих постов нестабильно работают, и часть пользователей видит некачественные рекомендации.
4. Пользователи вполне могут уставать от большого количества однотипного контента и переставать его лайкать, хотя он всё ещё будет появляться у них в ленте, пока система на это не отреагирует.
5. На результат данных также может полвиять длительность эксперимента — возможно, наш алгоритм ещё не успел подстроиться под всех пользователей.

Таким образом, мы можем попробовать провести новый эксперимент с большей длительнотью или отправить систему на доработку, а лучше — запараллелить оба этих процесса (при этом стоит помнить, что нам нельзя обновлять систему **во время** нового эксперимента).