

Вводные данные: В период с 08.01.18 по 09.01.18 был проведен А/В тест. Тестовой группе был предложен товар со скидкой. Задача: Оцените эффект данной скидки с точки зрения изменения дохода для пользователей RU региона. Оформить в виде выводов с визуализацией, кратко пояснить методологию расчетов. Написать SQL запрос. Данные необходимо забрать с помощью SQL запроса: Имеются 2 таблицы с данными. В первой таблице «Users» содержатся идентификационный номер пользователей «SPA_ID», регион «Realm» и тип группы «Type_of_group». Во второй таблице «Revenue» содержатся идентификационный номер пользователей «SPA_ID», дата «DT» и «Revenue» (запись отсутствует, если пользователь не платил).

```
In [1]: import pandas as pd
import numpy as np
from scipy import stats
import seaborn as sns
sns.set()
import matplotlib.pyplot as plt
from scipy.stats import normaltest

In [2]: # настройки отображения графиков
%matplotlib inline
plt.style.use('ggplot')
plt.rcParams['figure.figsize'] = (15, 5)
plt.rcParams['font.family'] = 'sans-serif'

In [3]: df = pd.read_csv(
        'opt/kate_repo/real_data_analysis/wargaming_task/second_task.csv')

In [4]: df['Realm'] = df['Realm'].fillna('NA')
df.head()
```

Out[4]:

	SPA_ID	Realm	Type_of_group
0	1	RU	Target
1	2	EU	Target
2	3	NA	Target
3	4	SEA	Target
4	5	RU	Control

```
In [5]: df_second = pd.read_csv(
        'opt/kate_repo/real_data_analysis/wargaming_task/second_task_table2.csv')
df_second.head()

Out[5]:
```

	SPA_ID	dt	Revenue
0	2	1/8/2018	15.8
1	3	1/2/2018	24.0
2	5	1/9/2018	8.6
3	6	1/14/2018	5.8
4	9	1/8/2018	21.6

```
In [6]: from sqlalchemy import create_engine

In [7]: # Instantiate sqlalchemy.create_engine object
engine = create_engine('postgres://postgres:postgres@localhost/sql_example')
```

Save the data from dataframe to postgres table "second_task" df.to_sql('wargaming', engine, index=False # Not copying over the index) # Save the data from dataframe to postgres table "second_task_table2" # df_second.to_sql('wargaming_second', engine, index=False # Not copying over the index)

```
In [8]: conn = engine.connect()
total_pivot = pd.read_sql("""
        CREATE TEMPORARY TABLE wargaming_all("SPA_ID" int8 NULL,
        "Realm" text NULL,
        "Type_of_group" text NULL,
        "dt" text NULL,
        "Revenue" float8 NULL
    );

    INSERT INTO wargaming_all
    SELECT
        *
    FROM
        wargaming
    left JOIN wargaming_second
    USING ("SPA_ID")
    WHERE
        wargaming."Realm" = 'RU'
    and
        (wargaming_second."dt" = '1/8/2018'
        OR wargaming_second."dt" = '1/9/2018')
    ORDER BY
        wargaming."SPA_ID";

--средняя выручка, общая выручка и количество операций в группax (wargaming_agg)
CREATE TEMPORARY TABLE wargaming_agg("Type_of_group" text NULL,
    "summ" float8 NULL,
    "avg_revenue" float8 NULL,
    "count_transaction" int8 NULL
);

INSERT INTO wargaming_agg
SELECT
    DISTINCT "Type_of_group",
    sum(wargaming_all."Revenue") OVER (PARTITION BY wargaming_all."Type_of_group") AS summ,
    avg(wargaming_all."Revenue") OVER (PARTITION BY wargaming_all."Type_of_group") as avg_revenue,
    count(wargaming_all."Revenue") OVER (PARTITION BY wargaming_all."Type_of_group") AS count_transaction
FROM
    wargaming_all;

--количество уникальных id в группax (wargaming_uniq_id_pay), пользователей, которые платили
CREATE TEMPORARY TABLE wargaming_uniq_id_pay("Type_of_group" text NULL,
    "count_id_pay" int8 NULL
);

INSERT INTO wargaming_uniq_id_pay
SELECT "Type_of_group", count(DISTINCT "SPA_ID") as count_id_pay FROM wargaming_all group by "Type_of_group";

--количество уникальных id в группax (wargaming_uniq_id_group), пользователей, которые платили
CREATE TEMPORARY TABLE wargaming_uniq_id_group("Type_of_group" text NULL,
    "count_id_max" int8 NULL
);

INSERT INTO wargaming_uniq_id_group
SELECT "Type_of_group", count(DISTINCT "SPA_ID") as count_id_max FROM wargaming group by "Type_of_group";

-- сводная таблица агрегированных значений по группам
CREATE TEMPORARY TABLE wargaming_total("Type_of_group" text NULL,
    "summ" float8 NULL,
    "avg_revenue" float8 NULL,
    "count_transaction" int8 NULL,
    "count_id_pay" int8 NULL,
    "count_id_max" int8 NULL
);

INSERT INTO wargaming_total
SELECT * from wargaming_agg LEFT JOIN wargaming_uniq_id_pay USING ("Type_of_group")
LEFT JOIN wargaming_uniq_id_group USING ("Type_of_group");

Select*from wargaming_total;
""", conn)
total_pivot.head()
```

Out[8]:

	Type_of_group	summ	avg_revenue	count_transaction	count_id_pay	count_id_max
0	Control	72449.9	10.657532	6798	3819	20000
1	Target	84550.8	10.558292	8008	4092	20000

```
In [105]: # Сравнение количества плательщиков в контрольной и тестовой группе
profit_control=total_pivot.count_id_pay.min()*100/total_pivot.count_id_max.max()
profit_target=total_pivot.count_id_pay.max()*100/total_pivot.count_id_max.max()
print('%% плательщиков без скидки: %.3f' % profit_control)
print('%% плательщиков со скидкой: %.3f' % profit_target)

%% плательщиков без скидки: 19.095
%% плательщиков со скидкой: 20.460

In [107]: # Разница между количеством плательщиков в %
profit_target-profit_control
```

Out[107]: 1.3650000000000002

Скидка привлекла на 1% плательщиков больше

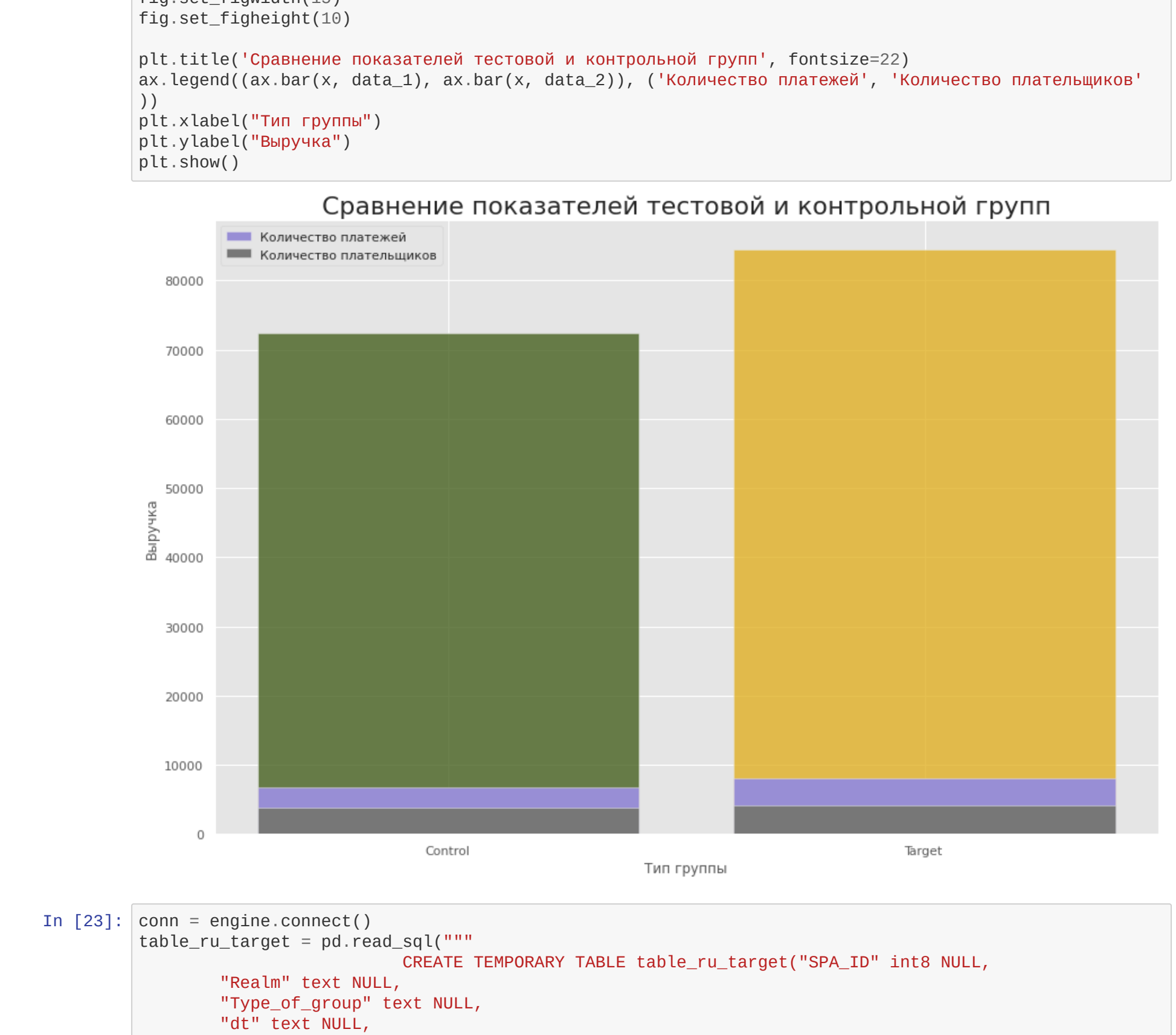
```
In [112]: # Разница выручки в контрольной и тестовой группе
profit=(total_pivot.summ.min()*100/total_pivot.summ.max())-100
profit

Out[112]: -14.311987586161436
```

```
In [114]: # Платежные операции в контрольной и тестовой группе
transaction=(total_pivot.count_transaction.min()*100/total_pivot.count_transaction.max())-100
transaction

Out[114]: -15.109890109890117
```

Доход в тестовой группе на 14% выше, как и процент платежных операций. Постоянные пользователи стали совершать операции чаще, но сумма операций существенно не изменилась



```
In [23]: conn = engine.connect()
table_ru_target = pd.read_sql("""
        CREATE TEMPORARY TABLE table_ru_target("SPA_ID" int8 NULL,
        "Realm" text NULL,
        "Type_of_group" text NULL,
        "dt" text NULL,
        "Revenue" float8 NULL
    );

    INSERT INTO table_ru_target
    SELECT
        *
    FROM
        wargaming
    left JOIN wargaming_second
    USING ("SPA_ID")
    WHERE
        wargaming."Realm" = 'RU'
    and
        (wargaming_second."dt" = '1/8/2018'
        OR wargaming_second."dt" = '1/9/2018')
    and
        wargaming."Type_of_group" = 'Target'
    ORDER BY
        wargaming."SPA_ID";

Select*from table_ru_target;
""", conn)
table_ru_target.head()

Out[23]:
```

	SPA_ID	Realm	Type_of_group	dt	Revenue
0	1	RU	Target	1/8/2018	2.8
1	9	RU	Target	1/9/2018	10.3
2	9	RU	Target	1/8/2018	21.6
3	17	RU	Target	1/9/2018	14.2
4	17	RU	Target	1/8/2018	8.4

```
In [53]: table_ru_target.Revenue.hist(bins=100)

Out[53]: <AxesSubplot:~>
```



```
In [80]: # Проверка распределения на нормальность
from scipy.stats import normaltest
data = table_ru_target.Revenue
stat, p = normaltest(data)
alpha = 0.05
if p > alpha:
    print('Нормальное распределение')
else:
    print('Распределение не является нормальным')
p

Распределение не является нормальным

Out[80]: 1.45841702908998e-20
```

```
In [25]: conn = engine.connect()
table_ru_control = pd.read_sql("""
        CREATE TEMPORARY TABLE table_ru_control("SPA_ID" int8 NULL,
        "Realm" text NULL,
        "Type_of_group" text NULL,
        "dt" text NULL,
        "Revenue" float8 NULL
    );

    INSERT INTO table_ru_control
    SELECT
        *
    FROM
        wargaming
    left JOIN wargaming_second
    USING ("SPA_ID")
    WHERE
        wargaming."Realm" = 'RU'
    and
        (wargaming_second."dt" = '1/8/2018'
        OR wargaming_second."dt" = '1/9/2018')
    and
        wargaming."Type_of_group" = 'Control'
    ORDER BY
        wargaming."SPA_ID";

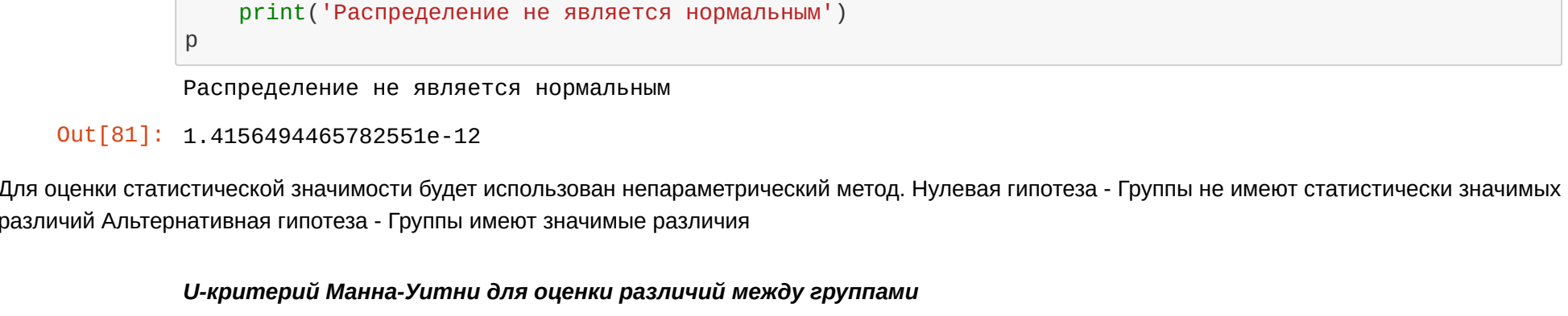
Select*from table_ru_control;
""", conn)
table_ru_control.head()

Out[25]:
```

	SPA_ID	Realm	Type_of_group	dt	Revenue
0	5	RU	Control	1/9/2018	3.3
1	5	RU	Control	1/9/2018	8.6
2	29	RU	Control	1/8/2018	21.5
3	29	RU	Control	1/8/2018	6.3
4	37	RU	Control	1/8/2018	20.2

```
In [57]: table_ru_control.Revenue.hist(bins=100)

Out[57]: <AxesSubplot:~>
```



```
In [81]: # Проверка распределения на нормальность
from scipy.stats import normaltest
data = table_ru_control.Revenue
stat, p = normaltest(data)
alpha = 0.05
if p > alpha:
    print('Нормальное распределение')
else:
    print('Распределение не является нормальным')
p

Распределение не является нормальным

Out[81]: 1.4156494465782551e-12
```

Для оценки статистической значимости будет использован непараметрический метод. Нулевая гипотеза - Группы не имеют статистически значимых различий Альтернативная гипотеза - Группы имеют значимые различия

U-критерий Манна-Уитни для оценки различий между группами

```
In [87]: from scipy.stats import mannwhitneyu

# generate two independent samples
data1 = table_ru_target.Revenue
data2 = table_ru_control.Revenue
# compare samples
stat, p = mannwhitneyu(data1, data2)
print('Statistics=%.3f, p=%.3f' % (stat, p))
# interpret
alpha = 0.05
if p > alpha:
    print('Не можем отклонить нулевую гипотезу ')
else:
    print('Отклоняем нулевую гипотезу')

Statistics=26929913.500, p=0.132
Не можем отклонить нулевую гипотезу

In [ ]: # Статистически достоверные различия в группах не обнаружены

In [ ]: Для пользователей RU региона статистически значимый эффект от скидки отсутствует.
```