Финальная работа по курсу “Аналитик данных с нуля 2.0”

**Студент:** Любавский П.Г.

**Цель работы:** Выяснить, стоит ли проводить акцию в дальнейшем. Если игроки, участвовавшие в

акции, принесли больше денег, чем игроки, у которых акции не было, то стоит

повторять акцию и при этом уже на всех игроках.

**План выполнения:**

1. Первоначальная обработка полученных данных, очистка выявленных читеров.

2. Поиск и очистка не выявленных читеров.

3. Проведение расчетов ARPU, ARPPU и средних трат внутриигровой валюты для каждой платформы и по каждой группе пользователей.

4. Проведение расчета 95% доверительного интервала для каждого выявленного показателя. Подготовка и разработка метрик для ПО Power BI и MS Excel.

5. Выводы.

**1. Первоначальная обработка полученных данных, очистка выявленных читеров.**

Для начала рассмотрим данные полученные после проведения А/В тестирования и более детально опишем их.

*import pandas as pd*

*import numpy as np*

*import matplotlib.pyplot as plt*

*df1 = pd.read\_csv("ABgroup.csv")*

*df2 = pd.read\_csv("Cash.csv")*

*df3 = pd.read\_csv("Money.csv")*

*df4 = pd.read\_csv("Cheaters.csv")*

*df5 = pd.read\_csv("Platforms.csv")*

df1 – метрика из 2 столбцов и 8640000 строк, в которой содержатся айди номера пользователей и булево значение группы игроков: тест и контрольная.

df2 – метрика из 3 столбцов и 8640000 строк, в которой содержатся айди номера, даты и сумма внутриигровой валюты.

df3 – метрика из 3 столбцов и 8640000 строк, в которой содержатся айди номера, даты и сумма внесенного платежа за внутриигровую валюту

df4 – метрика из 2 столбцов и 8640000 строк, в которой содержатся айди номера и булево значение по использованию чит-кода: да или нет (0 / 1)

df5 – метрика из 2 столбцов и 8640000 строк, в которой содержатся айди номера и использование платформы игры: PC = 2881992, Xbox = 2884256, PS4 = 2873752.

Предварительно посмотрев на датафреймы видно, что у всех таблиц есть общий столбец-ключ user\_id. Также видно, что количество игровых сессий на каждой платформе приблизительно одинаковое. Основные датафреймы у нас df2, df3 – остальные мы сможем использовать для фильтрации по платформам, выявленным читерам и кластерам пользователей. Данные достаточно объёмные (8640000 строк) поэтому для работы с данными будем использовать python.

Приступим к выполнению первой задачи.

1. Стоит определить в фрейме данных df1 общее кол-во пользователей по уникальному айди номеру:

*df1.user\_id.nunique() #1080000*

1. Теперь мы знаем, что общее кол-во игроков на всех платформах: 1080000. Далее, попробуем объединить 2 фрейма данных с внесенными платежами и полученной игровой валюты. Объединение будем использовать методом merge(), так как количество строк весьма большое, а в качестве ключей программа будет использовать столбец user\_id и date.

*df\_pays = df2.merge(df3, how = 'left')*

1. Отлично, теперь у нас есть общий датафрейм с айди номерами, датой, внесенным платежом и полученной внутриигровой валютой. Далее попробуем сгруппировать датафрейм по user\_id так, чтобы суммировать значения в столбцах cash и money по каждому пользователю.

*df\_pays = df\_pays.groupby(['user\_id']).sum()*

1. С помощью группировки теперь у нас есть датафрейм в котором в качестве индекса программа сразу определила столбец с нашими айди номерами пользователей, также по каждому отдельному пользователю есть столбец с его общими платежами за весь период и полученной внутриигровой валюты. Можем начать готовить фильтры под этот датафрейм. Рассмотрим df4. Как мы ранее уже видели, в качестве столбца cheaters у нас используются булевы значения 0 / 1. Логично, что значения 0 – не был применен чит, 1 – был применен чит. Кажется, можем воспользоваться предыдущим кодом с группировкой всех сессий одного пользователя по столбцу user\_id и найти каждого пользователя, кто не применял читы.

*df\_search = df4.loc[df4['cheaters'] > 0]*

*df\_search = df\_search.groupby(['user\_id']).sum()*

1. 353 пользователя по данным выявленной метрики df4 у нас использовали читы и 1079647 не применяли читы. При чем, можем применить следующий код и удостовериться, что каждый из пользователей применял читы 8 раз! Также вернем в нашу переменную тех, кто не использовал читы.

*df\_search.loc[df\_search['cheaters'] == 8] #353 пользователя, всё верно*

*df4 = df4.loc[df4['cheaters'] == 0]*

*df4 = df4.groupby(['user\_id']).sum()*

1. Отлично, теперь можем сделать наш первый фильтр по очистке выявленных читеров и сохраним его в переменную filter\_cheaters.

*filter\_cheaters = df4.index*

1. Теперь попробуем сделать фильтр по платформам из датафрейма df5. Предварительно ознакомившись с метрикой мы знаем, что здесь в столбце platform используются обозначения 3 видов игровых платформ: xbox, pc, ps4. У нас не получиться выполнить группировку с суммированием, так как в качестве обозначений платформы используется тип object в столбце и тем более, не факт, что каждый пользователь во время своей игровой сессии использовал только одну и ту же платформу. Можно это проверить. Мы можем поступить простым способом, настроив фильтрацию по столбцу platform и выбрать отдельно каждую платформу, но я хочу попробовать другим способом. Попробую в качестве эксперимента сделать список с числовым обозначением платформы, Xbox = 1, PC = 2, PS4 = 3. Мы знаем наше общее кол-во пользователей и мы можем посчитать, что при суммировании каждой сессии одного пользователя если он использовал только одну платформу – мы получим на одного пользователя числовые значения 8 или 16 или 24, так как мы знаем, что один пользователь по метрикам играл 8 раз за весь период. Тем более, возможно, этот список нам в дальнейшем пригодится для других целей, а если и нет, то он нам не будет мешать и будет находится в одной переменной. Попробуем.

*plats\_num = []*

*for i in df5['platform']:*

*if i == 'XBox':*

*plats\_num.append(1)*

*elif i == 'PC':*

*plats\_num.append(2)*

*else:*

*plats\_num.append(3)*

1. Теперь у нас есть список, по каждой сессии в соответствии с выбранной платформой. Попробуем добавить новый столбец к нашему df5 и сгруппируем его по каждому пользователю и заодно проверим, использовалась каждым пользователем одна платформа или разные.

*df5['platform\_num'] = plats\_num*

*df5 = df5.groupby(['user\_id']).sum()*

*df5.loc[(df5['platform\_num'] != 8) & (df5['platform\_num'] != 16) & (df5['platform\_num'] != 24)]*

1. Отлично, теперь мы знаем, что каждый айди номер (пользователь) использовал во всех своих игровых сессиях только одну и ту же платформу. Прежде чем сделать фильтры по платформам – очистим наш датафрейм от уже известных нам читеров и заодно посмотрим, сколько читеров у нас было на одной из платформ. Нам известно по предварительной подготовке данных, что всего у нас 1080000 пользователей из которых: Xbox – 360532 пользователя, РС – 360249 пользователя и PS4 – 359219 пользователей. Кстати, можно сделать промежуточный вывод, что у нас приблизительно одинаковое кол-во пользователейна платформах Xbox и PC. Посмотрим на какой из платформ у нас было зафиксировано большее кол-во читеров.

*df5 = df5.loc[df5.index.isin(filter\_cheaters)]*

*filter\_xbox = df5.loc[df5['platform\_num'] == 8] # читеров нет*

*filter\_xbox = filter\_xbox.index*

*filter\_pc = df5.loc[df5['platform\_num'] == 16] # 360249 – 359896 = 353 читера*

*filter\_pc = filter\_pc.index*

*filter\_ps4 = df5.loc[df5['platform\_num'] == 24] # читеров нет*

*filter\_ps4 = filter\_ps4.index*

1. Все наши расчеты - совпали и мы теперь знаем, что все обнаруженные читеры у нас были на одной платформе: PC (353 читера). Также у нас есть фильтры по платформам с уже очищенными данными от известных читеров. Теперь переходим к метрике по кластерам пользователей. По предварительной оценки датафрейма df1 мы знаем, что у нас есть 2 кластера пользователей: control = 4322776 пользователей, test = 4317224 пользователя. Группа control скорее всего относиться к пользователям которые не попали под внутриигровую акцию, а группа test – пользователи на которых тестировали проведенную акцию. Что мы можем сделать: вариантов как использовать наши уже работающие фильтры – много и мы вообще можем воспользоваться нашим алгоритмом действий из предыдущей подготовки фильтров по платформам. Но в этот раз попробуем сделать немного иначе. Мы знаем, что по условиям задания в дальнейшем – мы будем работать с нашим уже подготовленным, но ещё не отфильтрованным датафреймом df\_pays. Также нам нужно подготовить под этот датафрейм 6 фильтров, то есть фильтры по каждой платформе с фильтром по каждому кластеру, платформ у нас 3, а кластеров 2. Кажется самым логичным решением будет взять наш датафрейм df1 и с помощью наших фильтров filter\_xbox, filter\_pc и filter\_ps4 – подготовим 6 новых фильтров и посмотрим, что у нас из этого получиться. Напишем функцию, которая будет принимать значения df1 колонку: test или control, и фильтр: filter\_xbox, filter\_pc, filter\_ps4.

*def filters(column = 'test', filtering = filter\_xbox):*

*df1\_t = df1.loc[df1['group'] == column]*

*df1\_test = df1\_t.groupby(['user\_id'])*

*df1\_test = df1\_test.first()*

*filter\_ = df1\_test.loc[df1\_test.index.isin(filtering)]*

*filter\_ = filter\_.index*

*return filter\_*

1. Чтож попробуем с помощью нашей функции собрать все необходимые нам фильтры.

*fil\_t\_xbox = filters(column = 'test', filtering = filter\_xbox) # 180077 пользователей*

*fil\_t\_pc = filters(column = 'test', filtering = filter\_pc) # 179982 пользователей*

*fil\_t\_ps4 = filters(column = 'test', filtering = filter\_ps4) # 179415 пользователей*

*fil\_c\_xbox = filters(column = 'control', filtering = filter\_xbox) # 180455 пользователей*

*fil\_c\_pc = filters(column = 'control', filtering = filter\_pc) # 179914 пользователей*

*fil\_c\_ps4 = filters(column = 'control', filtering = filter\_ps4) # 179804 пользователей*

1. Проверим себя. Мы знаем, что у нас после удаления известных читеров, общее кол-во пользователей должно быть – 1079647. Соответственно, если мы суммируем по каждому фильтры всех пользоватлей – мы должны получить 1079647 и убедиться, что всё сработало правильно. 180077+179982+179415+180455+179914+179804 = 1079647. Отлично, у нас всё готово для дальнейшей работы.

**2. Поиск и очистка не выявленных читеров.**

1. Прежде чем начать, установим необходимые библиотеки.

*import matplotlib.pyplot as plt*

*import seaborn as sns*

*from scipy.stats import kurtosis*

*from scipy.stats import skew*

*from scipy import stats*

1. Теперь создадим новые датафреймы применив для этого наши фильтры для очистки не нужных данных. Для начала напишем функцию.

*def new\_frames(filters = fil\_t\_xbox):*

*n\_frame = df\_pays.loc[df\_pays.index.isin(filters)]*

*return n\_frame*

1. Теперь вызовем нашу функцию для создания датафреймов.

*df\_t\_xbox = new\_frames(filters = fil\_t\_xbox)*

*df\_t\_pc = new\_frames(filters = fil\_t\_pc)*

*df\_t\_ps4 = new\_frames(filters = fil\_t\_ps4)*

*df\_c\_xbox = new\_frames(filters = fil\_c\_xbox)*

*df\_c\_pc = new\_frames(filters = fil\_c\_pc)*

*df\_c\_ps4 = new\_frames(filters = fil\_c\_ps4)*

1. Отлично, у нас всё готово. Начнем с анализирования каждого датафрейма по стандартным параметрам (СКО, среднее, квантили 25, 50, 75, эксцесс, ассиметрия). Для этого напишем исследовательскую функцию.

*def research\_func(df = df\_t\_xbox, x = 'test/xbox'):*

*print('Базовые метрики для ', x)*

*print()*

*print(round(df.describe(), 2))*

*print('------------------------------------')*

*print('Эксцесс ', kurtosis(df))*

*print('Ассиметрия ', skew(df))*

1. Рассмотрим df\_t\_xbox. Также построим гистограмму распределения по каждому столбцу воспользовавшись библиотекой seaborn.

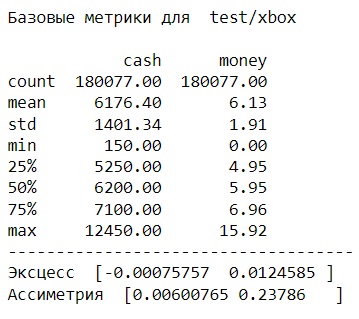
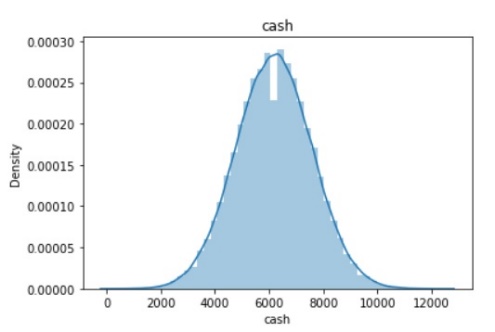
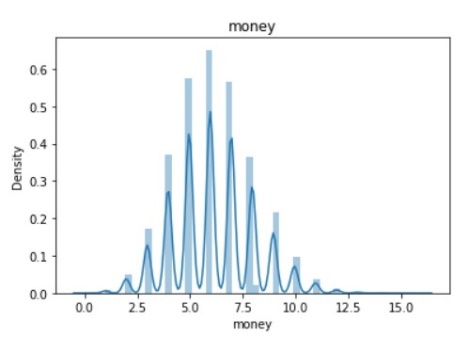
*research\_func()*

*sns.distplot(df\_t\_xbox['money'])*

*plt.title('money')*

*sns.distplot(df\_t\_xbox['cash'])*

*plt.title('cash')*

Кажется тут без аномалий, все параметры более - менее в норме, а показатели эксцесса и ассиметрии близки к 0.

Рассмотрим df\_t\_pc. Также построим гистограмму распределения по каждому столбцу.

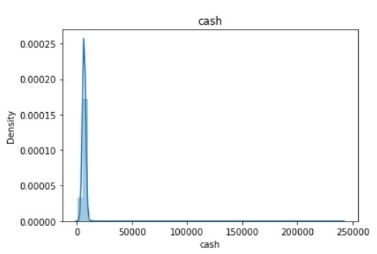
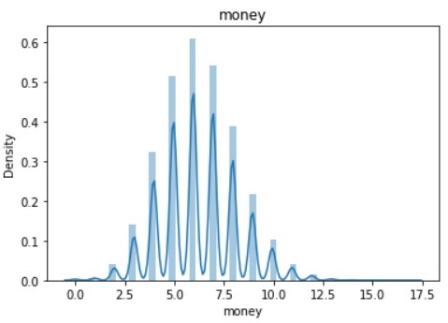
*research\_func(df = df\_t\_pc, x = ‘test/pc’)*

*sns.distplot(df\_t\_pc['money'])*

*plt.title('money')*

*sns.distplot(df\_t\_pc['cash'])*

*plt.title('cash')*



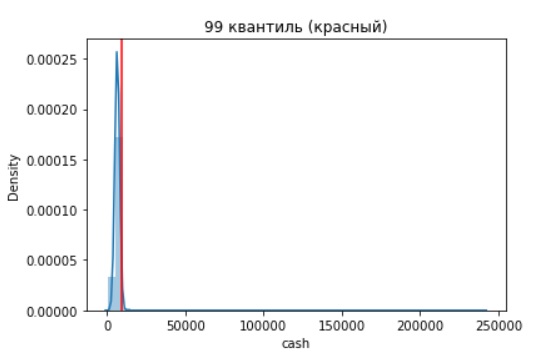
Кажется здесь мы сможем найти новых читеров. Предварительно проанализировав параметры мы видим слишком большую ассиметрию по столбцу cash, а также максимальное значение в 240950. Также логически мы можем взглянуть на гистограмму внесенных платежей money и увидеть, что тут никаких явно выраженных выбросов, то есть мы не имеем дело с нормальным или бимодельным распределением, а просто кто то из пользователей тратил намного больше виртуальной валюты, чем внес платеж или вообще его не вносил. Попробуем найти этих пользователей. Так как у нас не нормальное распределение по столбцу cash – мы не сможем построить доверительный интервал, но мы воспользуемся 99 квантилем. Для начала графически отобразим 99 квантиль на графике и убедимся, что всё отображается правильно.

*level\_99 = np.percentile(df\_t\_pc, 99)*

*sns.distplot(df\_t\_pc['cash'])*

*plt.axvline(x=level\_99, color="r", linestyle="-")*

*plt.title('99 квантиль (красный)')*



Готово. Теперь попробуем найти пользователей в датафрейме. Также воспользуемся 99 квантилем.

*new\_cheaters\_test =* *df\_t\_pc[df\_t\_pc['cash'] > np.percentile(df\_t\_pc['cash'], 99)]*

1756 пользователей использовали читы в тестовом кластере на платформе рс. На всякий случай, сохраним данные по этим пользователям new\_cheaters\_test и удалим их из нашего датафрейма.

*df\_t\_pc = df\_t\_pc.drop(new\_cheaters\_test.index, axis = 0)*

Рассмотрим df\_t\_ps4. Также построим гистограмму распределения по каждому столбцу.

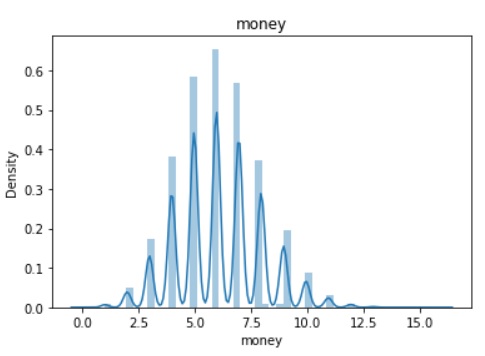
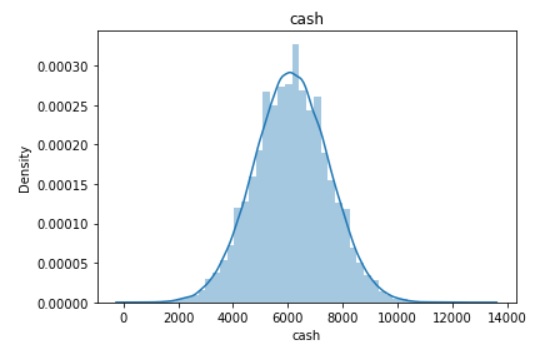
*research\_func(df = df\_t\_ps4, x = ‘test/ps4’)*

*sns.distplot(df\_t\_ps4['money'])*

*plt.title('money')*

*sns.distplot(df\_t\_ps4['cash'])*

*plt.title('cash')*



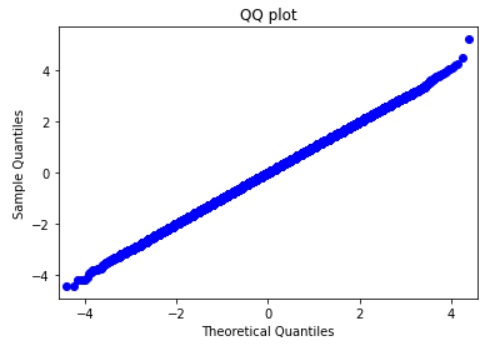
Кажется, здесь все в норме, несмотря на остроконечный пик по столбцу cash. Для уверенности проверим его с помощью графика QQ plot - <https://en.wikipedia.org/wiki/Q%E2%80%93Q_plot>. График сравнивает квантили выборки и выбранного распределения – в нашем случае по Гауссу. Для этого воспользуемся библиотекой statsmodels.

*import statsmodels.api as sm*

*sm.qqplot(df\_t\_ps4['cash'], fit = True)*

*plt.title("QQ plot")*

*plt.show()*



Всё отлично, идём дальше. Рассмотрим df\_с\_xbox. Также построим гистограмму распределения по каждому столбцу.

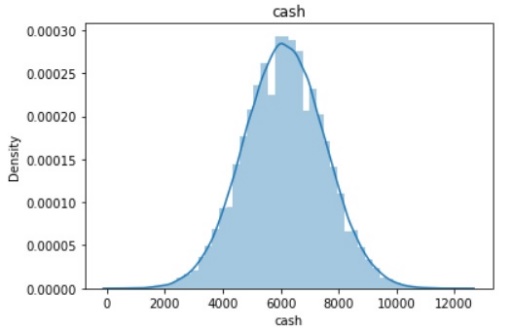
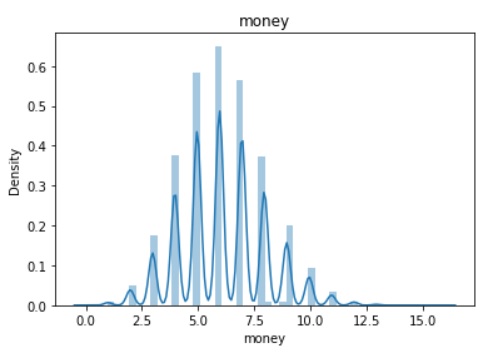
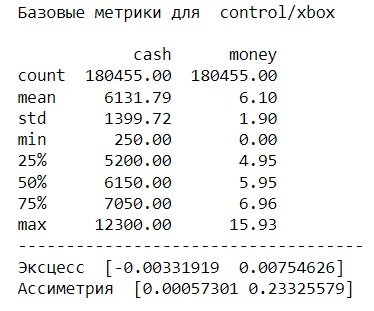
*research\_func(df = df\_c\_xbox, x = ‘control/xbox’)*

*sns.distplot(df\_c\_xbox['money'])*

*plt.title('money')*

*sns.distplot(df\_c\_xbox['cash'])*

*plt.title('cash')*



Кажется тут без аномалий, все параметры в норме.

Рассмотрим df\_с\_pc. Также построим гистограмму распределения по каждому столбцу.

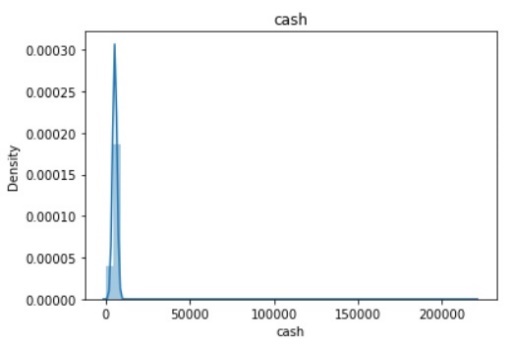
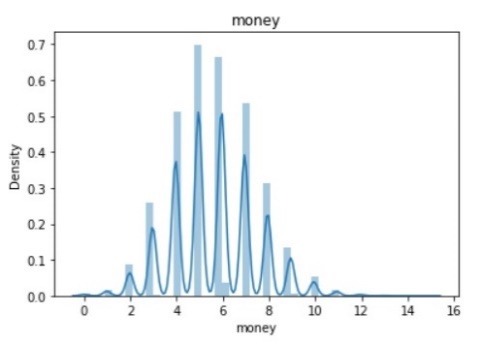
*research\_func(df = df\_c\_pc, x = ‘control/pc’)*

*sns.distplot(df\_c\_pc['money'])*

*plt.title('money')*

*sns.distplot(df\_c\_pc['cash'])*

*plt.title('cash')*



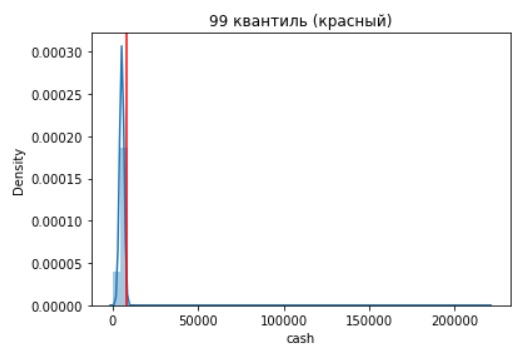
Кажется здесь мы снова сможем найти новых читеров. Воспользуемся тем же самым алгоритмом, которым нашли читеров в тестовом кластере на той же платформе.

*level\_99 = np.percentile(df\_c\_pc, 99)*

*sns.distplot(df\_c\_pc['cash'])*

*plt.axvline(x=level\_99, color="r", linestyle="-")*

*plt.title('99 квантиль (красный)')*



Обнаружили 1750 новых читеров. Теперь сохраним новых читеров в new\_cheaters\_control и удалим их из нашего датафрейма.

*new\_cheaters\_control = df\_c\_pc[df\_c\_pc['cash'] > np.percentile(df\_c\_pc['cash'], 99)]*

*df\_c\_pc = df\_c\_pc.drop(new\_cheaters\_control.index, axis = 0)*

Рассмотрим df\_с\_ps4. Также построим гистограмму распределения по каждому столбцу.

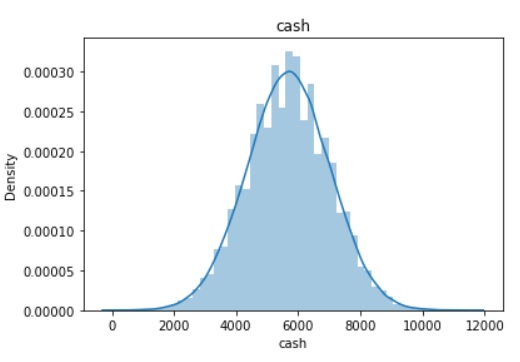
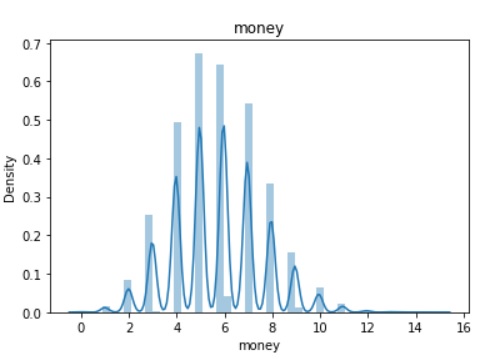
*research\_func(df = df\_c\_ps4, x = ‘control/ps4’)*

*sns.distplot(df\_c\_ps4['money'])*

*plt.title('money')*

*sns.distplot(df\_c\_ps4['cash'])*

*plt.title('cash')*



Кажется тут без аномалий, все параметры в норме. Можем подвести промежуточные итоги: нам удалось обнаружить новых читеров и очистить их из наших данных. Разобрали каждую метрику, проанализировали, можно отметить, что все найденные нами читеры использовали одну и ту же платформу: РС. Видимо, именно на этой платформе есть уязвимые места для системе безопасности, так как по метрике из df4 у нас также все читеры использовали платформу РС. Среди новых читеров в тестовом кластере мы обнаружили и очистили: 1756 пользователей, в контрольном кластере: 1750 пользователей.

**3. Проведение расчетов ARPU, ARPPU и средних трат внутриигровой валюты для каждой платформы и по каждой группе пользователей.**

1. Для начала проведем расчеты средней прибыли на одного игрока (ARPU) по каждой платформе и по каждой группе пользователей. Расчет проводится по формуле представленной ниже.

ARPU = A/B

где А – общий доход за весь интересующий период, В – общее кол-во всех пользователей

Напишем небольшую функцию и проведем всё расчеты.

*def arpu(df = df\_t\_xbox):*

*x = round(df['money'].sum() / df['money'].count(), 3)*

*return x*

*arpu\_t\_xbox = arpu() # 6.133*

*arpu\_t\_pc = arpu(df = df\_t\_pc) # 6.234*

*arpu\_t\_ps4 = arpu(df = df\_t\_ps4) # 6.085*

*arpu\_c\_xbox = arpu(df = df\_c\_xbox) # 6.104*

*arpu\_c\_pc = arpu(df = df\_c\_pc) # 5.617*

*arpu\_c\_ps4 = arpu(df = df\_c\_ps4) # 5.738*

1. Теперь проведем расчеты средней прибыли на одного платящего игрока (ARPPU) по каждой платформе и по каждой группе пользователей. Расчет проводится по формуле представленной ниже.

ARPPU = A/B

где А – общий доход за весь интересующий период, В – общее кол-во всех пользователей, которые вносили платежи

Напишем небольшую функцию и проведем всё расчеты.

*def arppu(df = df\_t\_xbox):*

*y = round(df['money'].sum() / df.query('money != 0')['money'].count(), 3)*

*return y*

*arppu\_t\_xbox = arppu() # 6.134*

*arppu\_t\_pc = arppu(df = df\_t\_pc) # 6.235*

*arppu\_t\_ps4 = arppu(df = df\_t\_ps4) # 6.086*

*arppu\_c\_xbox = arppu(df = df\_c\_xbox) # 6.104*

*arppu\_c\_pc = arppu(df = df\_c\_pc) # 5.62*

*arppu\_c\_ps4 = arppu(df = df\_c\_ps4) # 5.739*

1. Отлично, теперь проведем расчеты средней по каждому игроку внутриигровой валюты. Для этого просто воспользуемся встроенным методом mean.

*mean\_t\_xbox = round(df\_t\_xbox['cash'].mean(), 2) # 6176.4*

*mean\_t\_pc = round(df\_t\_pc['cash'].mean(), 2) # 6346.51*

*mean\_t\_ps4 = round(df\_t\_ps4['cash'].mean(), 2) # 6130.4*

*mean\_c\_xbox = round(df\_c\_xbox['cash'].mean(), 2) # 6131.79*

*mean\_c\_pc = round(df\_c\_pc['cash'].mean(), 2) # 5557.57*

*mean\_c\_ps4 = round(df\_c\_ps4['cash'].mean(), 2) # 5681.94*

**4. Проведение расчета 95% доверительного интервала для каждого выявленного показателя. Подготовка и разработка метрик для ПО Power BI и MS Excel.**

1. Теперь нам нужно провести расчеты 95% доверительного интервала для каждого датафрейма. Затем нам нужно будет подготовить данные для ПО Power BI и MS Excel. Для начала подготовим наши доверительные интервалы по каждой метрике. Чтобы ответить на главный вопрос нашей работы, нам нужно по парно сравнить каждую платформу из разных групп по каждой метрике (cash, money).
2. Рассмотрим метрики df\_t\_xbox, df\_c\_xbox. Мы уже знаем, что у нас по каждой метрике нормальное распределение, так как мы очистили наши данные от явно выраженных выбросов, поэтому будем использовать стандартные доверительные интервалы <https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.norm.html>

*def my\_norm\_confidence(df = df\_t\_xbox, column = 'cash'):*

*alpha = 0.95*

*interval = stats.norm.interval(alpha, loc = df[column].mean(), scale = df[column].std())*

*return interval*

*def basic\_reaserch(df = df\_t\_xbox, column = 'cash'):*

*print(round(df[column].loc[(df[column] > confidence[1]) |*

*(df[column] < confidence[0])].count() / df[column].count() \* 100, 2), '% аномалий по метрике')*

*df\_t\_xbox['cash'].plot(figsize = (14, 7))*

*plt.axhline(y = confidence[1], color="g", linestyle="-")*

*plt.axhline(y = confidence[0], color="r", linestyle="-")*

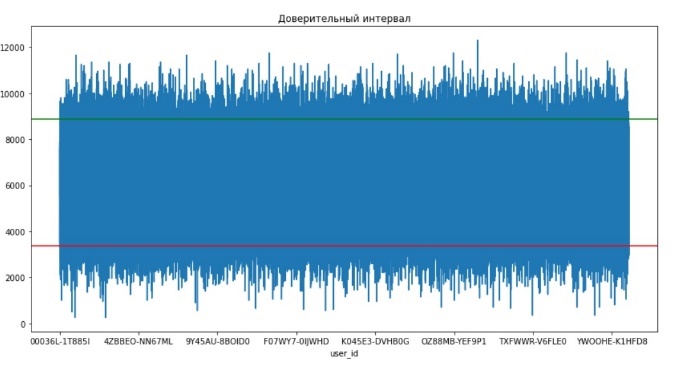
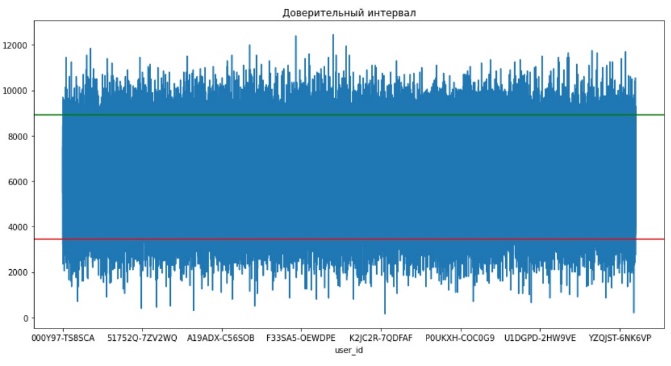
*plt.title('Доверительный интервал test,xbox,cash')*

*confidence = my\_norm\_confidence() # (3429.83, 8922.98)*

*basic\_reaserch() # 4.99 %*

*confidence = my\_norm\_confidence(df = df\_c\_xbox) # (3388.4, 8875.19)*

*basic\_reaserch(df = df\_c\_xbox) # 4.99 %*

*Control Test*

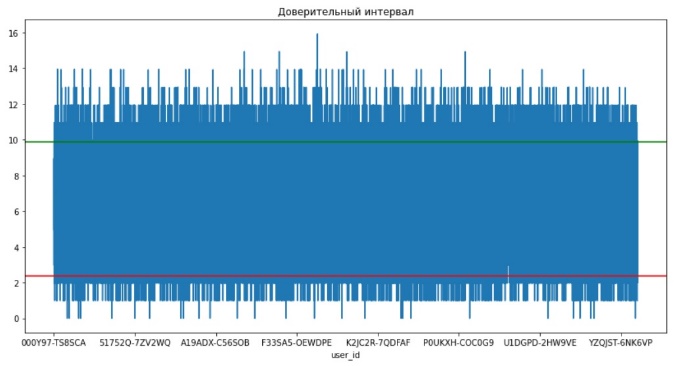
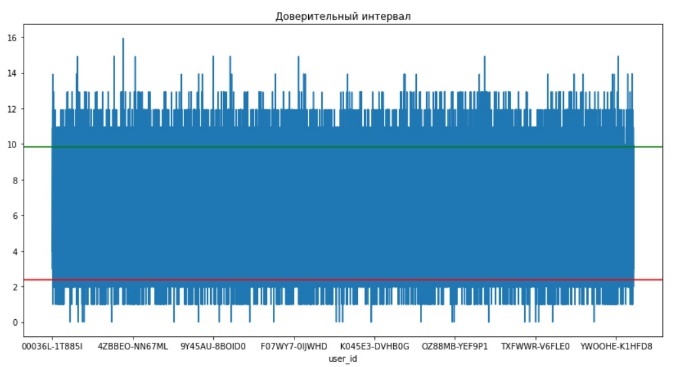
Промежуточный вывод: кажется отдачи от проведенной акции на платформе xbox, по метрике cash нет никакой. Интервалы находятся в приблизительно одном диапазоне, процент аномалий по обоим метрикам не превышает 5%.

*confidence = my\_norm\_confidence(df = df\_t\_xbox, column = ‘money’) # (2.38, 9.88)*

*basic\_reaserch(df = df\_t\_xbox, column = ‘money’) # 6.56 %*

*confidence = my\_norm\_confidence(df = df\_c\_xbox, column = ‘money’) # (2.37, 9.84)*

*basic\_reaserch(df = df\_c\_xbox, column = ‘money’) # 6.43 %*

 *Control Test*

Промежуточный вывод: кажется отдачи от проведенной акции на платформе xbox, по метрике money нет никакой. Интервалы находятся в приблизительно одном диапазоне, процент аномалий по тестовой группе немного выше, чем по контрольной группе.

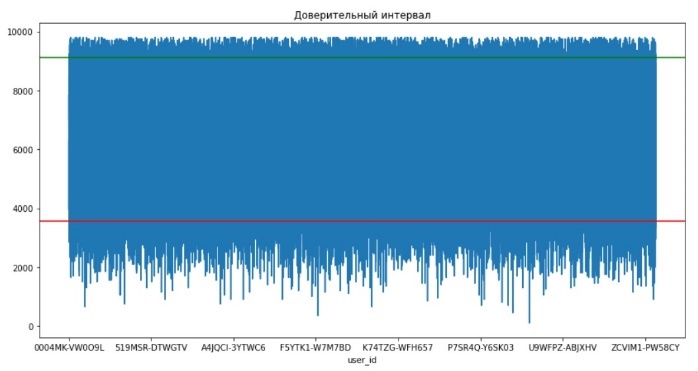
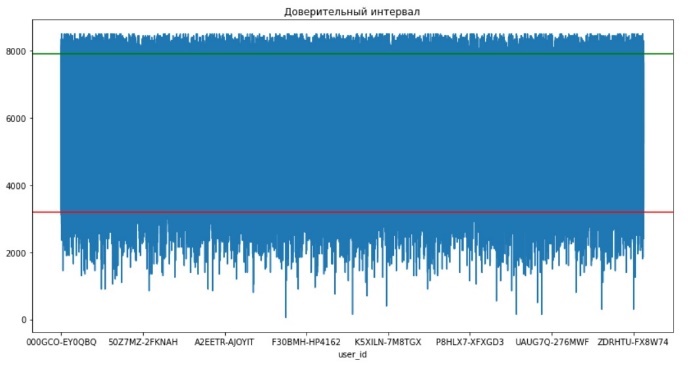
1. Рассмотрим метрики df\_t\_рс, df\_c\_рс.

*confidence = my\_norm\_confidence(df = df\_t\_pc, column = ‘cash’) # (3576.12, 9116.90)*

*basic\_reaserch(df = df\_t\_pc, column = ‘cash’) # 4.8 %*

*confidence = my\_norm\_confidence(df = df\_c\_pc, column = ‘cash’) # (3200.81, 7914.34)*

*basic\_reaserch(df = df\_c\_pc, column = ‘cash’) # 4.99 %*

 *Control Test*

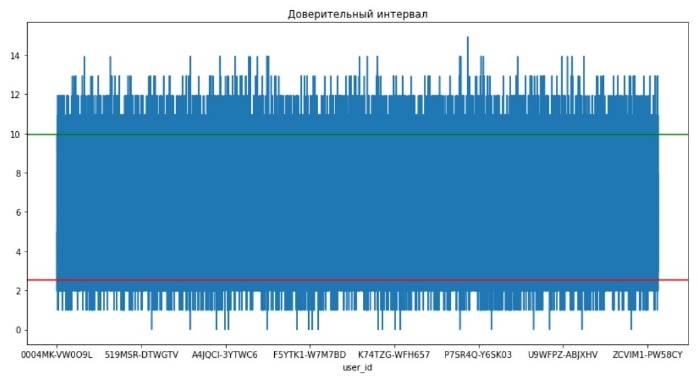
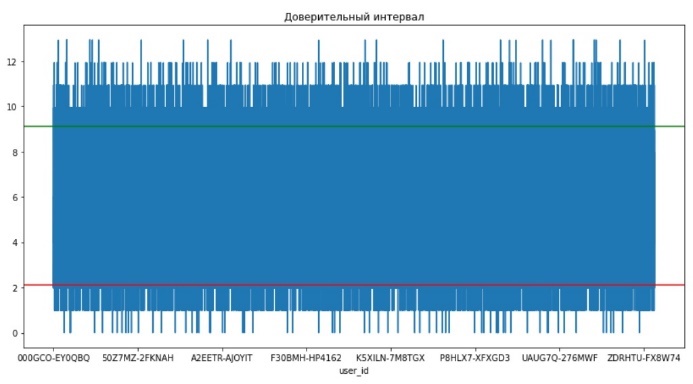
Промежуточный вывод: здесь мы уже можем наблюдать разницу в диапазонах. Мы видим, что верхний уровень интервала тестовой группы выше верхнего уровня интервала контрольной группы на 15%. Процент аномалий по контрольной группе немного выше, чем по тестовой группе.

*confidence = my\_norm\_confidence(df = df\_t\_рс, column = ‘money’) # (2.52, 9.95)*

*basic\_reaserch(df = df\_t\_рс, column = ‘money’) # 3.39 %*

*confidence = my\_norm\_confidence(df = df\_c\_рс, column = ‘money’) # (2.10, 9.13)*

*basic\_reaserch(df = df\_c\_рс, column = ‘money’) # 5.11 %*

 *Control Test*

Промежуточный вывод: мы также видим, что и уровень внесенных платежей у нас по тестовой группе выше, чем по контрольной и соответственно наша прибыль в тестовом режиме увеличилась почти на 9%. Также в контрольной группе почти на 2% было вычислено больше аномалий.

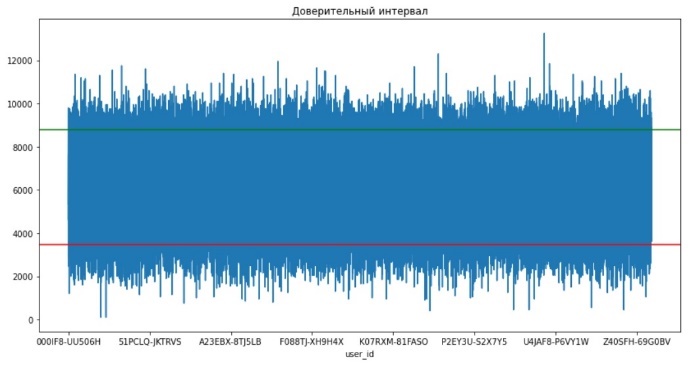
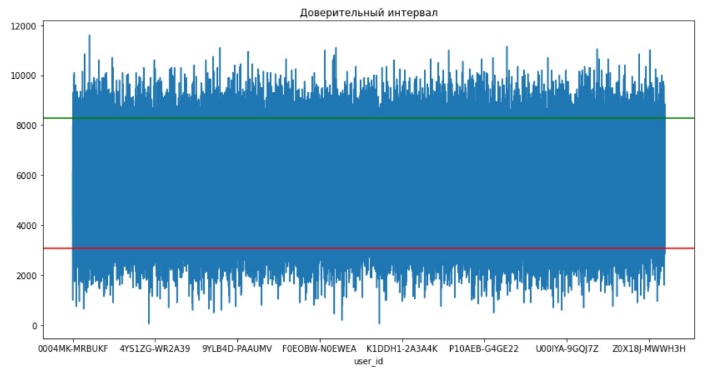
1. Теперь рассмотрим метрики df\_t\_ps4, df\_c\_ps4.

*confidence = my\_norm\_confidence(df = df\_t\_ps4, column = ‘cash’) # (3460.57, 8800.42)*

*basic\_reaserch(df = df\_t\_ps4, column = ‘cash’) # 4.93 %*

*confidence = my\_norm\_confidence(df = df\_c\_ps4, column = ‘cash’) # (3066.57, 8297.3)*

*basic\_reaserch(df = df\_c\_ps4, column = ‘cash’) # 5.13 %*

 *Control Test*

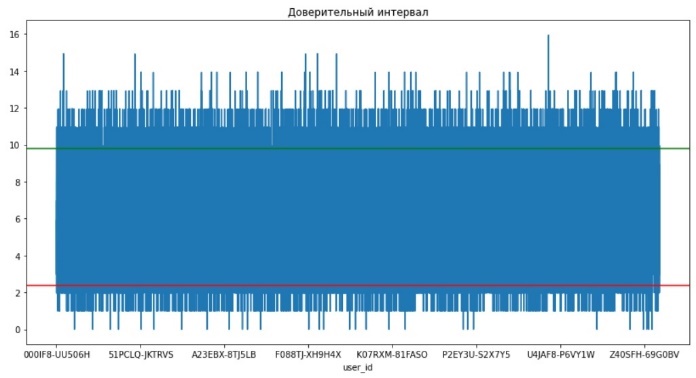
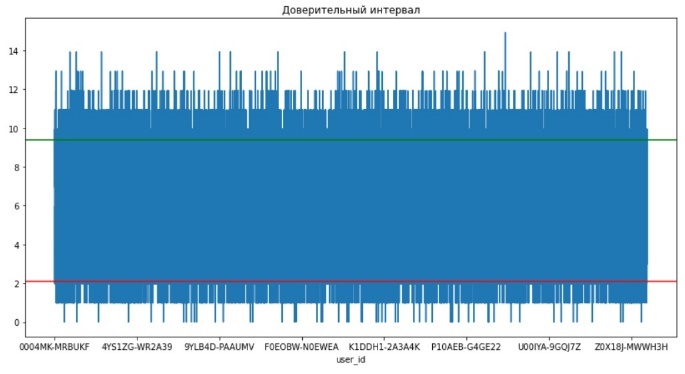
Промежуточный вывод: чтож, кажется, здесь наша акция тоже принесла свои плоды. Траты внутриигровой валюты на платформе рс4 тестовой группы пользователей на 6% выше, чем контрольной группы.

*confidence = my\_norm\_confidence(df = df\_t\_рs4, column = ‘money’) # (2.39, 9.78)*

*basic\_reaserch(df = df\_t\_рs4, column = ‘money’) # 6.1 %*

*confidence = my\_norm\_confidence(df = df\_c\_рs4, column = ‘money’) # (2.08, 9.4)*

*basic\_reaserch(df = df\_c\_рs4, column = ‘money’) # 5.82 %*

 *Control Test*

Промежуточный вывод: здесь кажется никакой большой разницы нет. Все данные приблизительно одинаковые.

1. Отлично, теперь нам нужно собрать данные для Power BI и Excel. По заданию нам нужно будет составить график сравнения метрик по дням в Power BI и сводную таблицу в эксель с ARPU по группам и платформам. Касаемо сводной таблицы в эксель по ARPU – мы её сможем сделать в самом экселе, тем более, что мы все наши значения сохранили в переменные. Поэтому нам остается только собрать данные для Power BI. Так как нам нужны по отдельности каждая игровая сессия всех пользователей с датами, мы поступим следующим образом: 1. Соберем из df1, df2, df3, df5 - один датафрейм, при чем так как у нас будут дубликаты в столбце ключ-значений, нам придется сгруппировать датафреймы df1, df5 и только потом их добавить в общий. 2. Очистим этот датафрейм от всех не нужных пользователей нашими уже готовыми фильтрами. Приступим.

*cash = pd.read\_csv("Cash.csv")*

*money = pd.read\_csv("Money.csv")*

*df\_cm = pd.merge(cash, money, on = ['user\_id', 'date'])*

Отлично, теперь сгруппируем и вольем df1.

*ABgroup = df1.groupby(['user\_id'])*

*df\_cm = df\_cm.merge(ABgroup.first(), on = 'user\_id')*

А теперь таким же образом вольем df5 и зададим столбец индекс с айди пользователями.

*Platform = pd.read\_csv("Platforms.csv")*

*Platform = Platform.groupby(['user\_id'])*

*df\_cm = df\_cm.merge(Platform.first(), on = 'user\_id')*

*df\_BI = df\_cm.set\_index('user\_id')*

1. Теперь осталось отфильтровать наш датафрейм для ПО Power BI. Создадим наш общий фильтр. Нам уже известно, что общее кол-во отфильтрованных пользователей = 1076141 пользователь и мы знаем, что у каждого пользователя было 8 игровых сессий, то есть в конечном результате мы должны получить 8609128 строк в датафрейме. Так как в общем фильтре нам нужны абсолютно все пользователи – будем использовать параметр outer.

*fil\_BI = df\_t\_xbox.merge(df\_t\_pc, how = 'outer', on = ['user\_id', 'cash', 'money'])*

*fil\_BI = fil\_BI.merge(df\_t\_ps4, how = 'outer', on = ['user\_id', 'cash', 'money'])*

*fil\_BI = fil\_BI.merge(df\_c\_xbox, how = 'outer', on = ['user\_id', 'cash', 'money'])*

*fil\_BI = fil\_BI.merge(df\_c\_pc, how = 'outer', on = ['user\_id', 'cash', 'money'])*

*fil\_BI = fil\_BI.merge(df\_c\_ps4, how = 'outer', on = ['user\_id', 'cash', 'money'])*

*fil\_BI = fil\_BI.index*

Всё готово, кол-во пользователей 1076141, теперь отфильтруем наш датафрейм и сконвертируем его в csv.

*df\_BI = df\_BI.loc[df\_BI.index.isin(fil\_BI)]*

*df\_BI.to\_csv('DS\_final\_powerbi.csv')*

**5. Выводы**

1. Кажется мы всё выполнили. Теперь подведем итоги. Будем идти по порядку нашей работы, чтобы не ничего не упустить.

Составим таблицу № 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Группа пользователей** | **Платформа** | **Количество игроков** | **Читеры** | **Обнаруженные читеры** |
| Test | XBox | 180 077 | 0 | 0 |
| PC | 180 161 | 179 | 1756 |
| PS4 | 179 415 | 0 | 0 |
| Control | XBox | 180 455 | 0 | 0 |
| PC | 180 088 | 174 | 1750 |
| PS4 | 179 804 | 0 | 0 |

Если рассматривать со стороны безопасности игры, то у нас явно есть проблемы с платформой РС. Из переданных нам метрик все обнаруженные читеры и в тестовом и контрольном кластере были на этой платформе и из тех, кого обнаружили мы позднее – все читеры использовали платформу РС. Однако, мы также можем заметить, что кол-во найденных читеров приблизительно одинаковое в двух кластерах, соответственно, проведенная нами акция не имеет никакого отношения к взломам игры, скорее это проблемы платформы.

Составим таблицу №2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Группа пользователей** | **Платформа** | **Средняя прибыль на каждого игрока (ARPU)** | **Средняя прибыль на каждого платящего игрока (ARPPU)** | **Средние траты внутриигровой валюты (СТВВ)** |
| Test | XBox | 6,133 | 6,134 | 6 176,4 |
| PC | 6,234 | 6,235 | 6 346,51 |
| PS4 | 6,085 | 6,086 | 6 130,4 |
| Control | XBox | 6,104 | 6,104 | 6 131,79 |
| PC | 5,617 | 5,62 | 5 557,57 |
| PS4 | 5,738 | 5,739 | 5 681,94 |

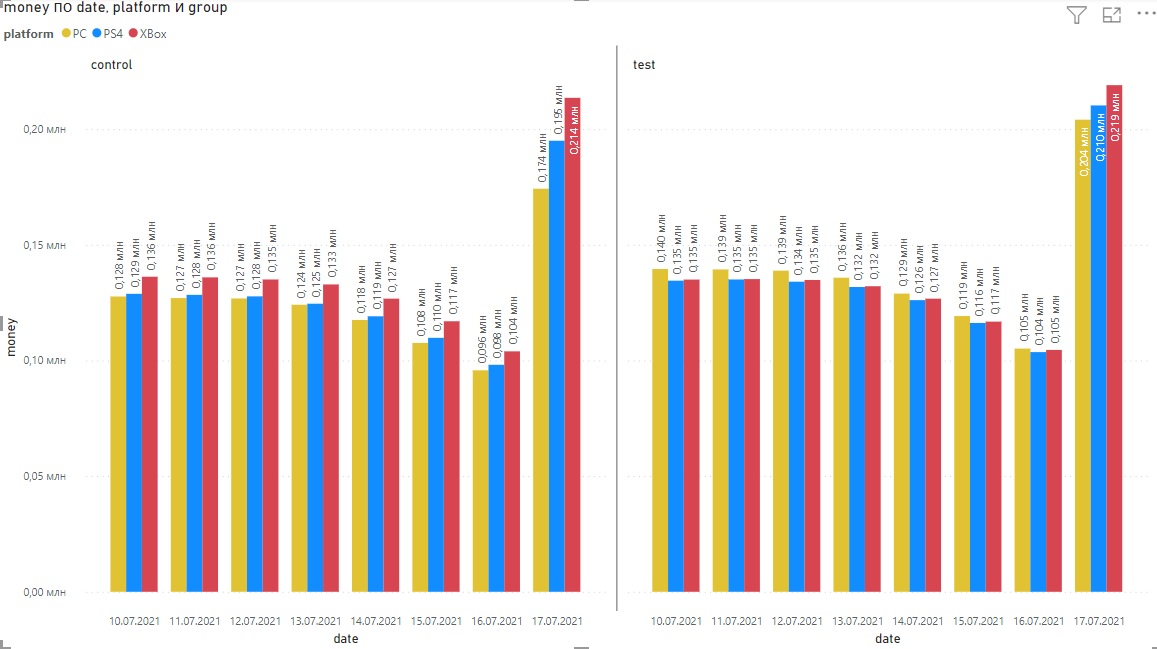
Здесь мы отчетливо видим, что по каждой метрике тестовой группы пользователей – все средние выше, чем в контрольной группе пользователей. Самый лучший показатель акции у нас на платформе РС и его разница между тестовой и контрольной группой почти 11%. Но мы также видим, что на платформе XBox наша акция практически не принесла результатов на рассматриваемых параметрах. Но всё равно показатели тестовой группы выше, чем в контрольной группе.

Составим таблицу №3.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Метрика** | **Платформа** | **Группа пользователей** | **Доверительный интервал** | **Доля выбросов** |
| Cash | XBox | Test | (3429.83, 8922.98) | 4,99% |
| Control | (3388.4, 8875.19) | 4,99% |
| PC | Test | (3576.12, 9116.90) | 4,8% |
| Control | (3200.81, 7914.34) | 4,99% |
| PS4 | Test | (3460.57, 8800.42) | 4,93% |
| Control | (3066.57, 8297.3) | 5,13% |
| Money | XBox | Test | (2.38, 9.88) | 6,56% |
| Control | (2.37, 9.84) | 6,43% |
| PC | Test | (2.52, 9.95) | 3,39% |
| Control | (2.10, 9.13) | 5,11% |
| PS4 | Test | (2.39, 9.78) | 6,11% |
| Control | (2.08, 9.4) | 5,82% |

Здесь мы видим, что практически по всем доверительным интервалам у нас результаты лучше в тестовой группе, чем в контрольной группе. Но опять же платформа XBox не чувствуется разница между контрольной и тестовой группой. Самая большая разница долей выбросов по платформе PC в метрике money, но и самая большая разница между кластерами у нас именно по этой платформе, а также самая большая доля выбросов у нас на платформе XBox. Все распределения по каждой метрике были в норме, проценты по выбросам у нас приблизительно везде одинаковые.

Также для более ясной картины используем наш график из ПО Power BI.



По графику мы также можем убедиться, что наша акция принесла больше прибыли от тестовой группы, чем от контрольной. Самый прибыльный день – 17.07.2021 в обоих группах, возможно это был выходной или праздничный день.

**Подведем итог.**

В дальнейшем нам нужно будет проводить акцию на всех пользователях, так как группа пользователей, которые не были в нашей акции – вносили меньше платежей и тратили меньше внутриигровой валюты, чем группа пользователей на которых мы тестировали нашу акцию. Наличие читеров никак не связано с проведением акции. Возможно, нам стоит улучшить условия нашей акции для платформы Xbox, так как по этой платформе наши результаты улучшились меньше, чем на других платформах. Также стоит обратить внимание на систему безопасности игры на платформе РС, так как эта платформа единственная поддается взломам.