Екатерина, привет! Мы рады тебя видеть на территории код-ревьюеров. Ты проделала большую работу над проектом, но давай познакомимся и сделаем его еще лучше! У нас тут своя атмосфера и несколько правил: 1. Меня зовут Александр Матвеевский. Я работаю код-ревьюером, моя основная цель — не указать на совершенные тобою ошибки, а поделиться своим опытом и помочь тебе стать аналитиком данных. 2. Общаемся на ты. 3. Если хочешь написать, спросить - не нужно стесняться. Только выбери свой цвет для комментария. 4. Это учебный проект, тут можно не бояться сделать ошибку. 5. У тебя неограниченное количество попыток для сдачи проекта. 6. Let's Go! --- Я буду красить комментарии цветом, пожалуйста, не удаляй их:
✍ __Комментарий от ревьюера №1__ Такой комментарий нужно исправить обязательно, он критически влияет на удачное выполнение проекта.

__Комментарий от ревьюера №1__ Такой комментарий является рекомендацией или советом. Можешь использовать их на своё усмотрение.

✔ __Комментарий от ревьюера №1__ Такой комментарий говорит о том, что было сделано что-то качественное и правильное =)

Предлагаю работать над проектом в диалоге: если ты что-то меняешь в проекте или отвечаешь на мои комментарии — пиши об этом. Мне будет легче отследить изменения, если ты выделишь свои комментарии:

Комментарии студента: Например, вот так.

Всё это поможет выполнить повторную проверку твоего проекта оперативнее. Если будут какие-нибудь вопросы по моим комментариям, пиши, будем разбираться вместе :)

Описание проекта

Вы — маркетинговый аналитик развлекательного приложения Procrastinate Pro+. Несмотря на огромные вложения в рекламу, последние несколько месяцев компания терпит убытки. Ваша задача — разобраться в причинах и помочь компании выйти в плюс. Есть данные о пользователях, привлечённых с 1 мая по 27 октября 2019 года: лог сервера с данными об их посещениях, выгрузка их покупок за этот период, рекламные расходы.

Вам предстоит изучить:

- откуда приходят пользователи и какими устройствами они пользуются,
- сколько стоит привлечение пользователей из различных рекламных каналов;
- сколько денег приносит каждый клиент,
- когда расходы на привлечение клиента окупаются,
- какие факторы мешают привлечению клиентов.

__Комментарий от ревьюера №1__ Отличная практика - расписывать цель и основные этапы своими словами (этот навык очень поможет на фильнальном проекте). Хорошо было бы добавить ход и цель исследования. Вот мой личный пример из 3 проекта: ![image.png](attachment:image.png)

Описание данных

visits_info_short.csv - лог сервера с информацией о посещениях сайта, orders_info_short.csv — информация о заказах, costs_info_short.csv — информация о расходах на рекламу.

visits_info_short.csv:

User Id — уникальный идентификатор пользователя, Region — страна пользователя, Device — тип устройства пользователя, Channel — идентификатор источника перехода, Session Start — дата и время начала сессии, Session End — дата и время окончания сессии.

orders_info_short.csv:

User Id — уникальный идентификатор пользователя, Event Dt — дата и время покупки, Revenue — сумма заказа.

costs_info_short.csv:

dt — дата проведения рекламной кампании, Channel — идентификатор рекламного источника, costs — расходы на эту кампанию.

visits orders costs

✔

Комментарий от ревьюера №2

Для подгрузки данных можно использовать конструкцию try-except , она поможет избежать потенциальных ошибок при загрузке данных, связанных, например, с некорректным указанием путей.

Подробнее о конструкции по ссылке:

https://pythonworld.ru/tipy-dannyx-v-python/isklyucheniya-v-python-konstrukciya-try-except-dlya-obrabotki-isklyuchenij.html

Либо же можно использовать стандартную библиотеку os:

https://pythonworld.ru/moduli/modul-os.html

Несколько интересных статей кейсы использования конструкции:

https://www.programiz.com/python-programming/exception-handling

https://towardsdatascience.com/do-not-abuse-try-except-in-python-d9b8ee59e23b

https://www.techbeamers.com/use-try-except-python/

Как вариант в try можно указать корректные пути (в нашем случае глобальные) в except - некорректные (локальные). Можно также специфицровать тип ошибки, FileNotFoundError или задать кастомный тип ошибки (FilePathError, например)

Она полезна, если ты работаешь локально, а потом подгружаешь проект на платформу. Конструкция позволит не падать коду и локально, и на сервере ЯП, так как если не сработает один блок с путями, сработает другой.

Ну и вообще, в целом полезно про эту констуркцию знать, она универсальна и может быть использована в разных задачах.

✍ __Комментарий от ревьюера №1__ Кажется, работа над проектом велась локально и пути к файлам не были изменены на общедоступные. Стоит переписать код таким образом, что бы он запускался и в jupiter hub, и локально без внесения дополнительных правок. Этого можно добиться многими способами. Например, использовать конструкцию try-except для путей файлов: try - пути на локальном компьютере, except - пути на сервере.

✍ __Комментарий от ревьюера №1__ В описании проекта дана структура, она поможет тебе построить последовательное исследование. Пожалуйста, расположи результаты своей работы в соответствии с этими шагами. До тех пор, пока это не будет сделано - я не смогу проверить твою работу.

✍ __Комментарий от ревьюера №2__ Екатерина, твоя работа выполнена не по структуре задания. Повторюсь, расположи, свою работу относительно последовательно. Также твоя работа имеет ошибку в коде. Некоторые блоки кода не работают. Посмотри, пожалуйста, что пошло не так. Перед отправкой проекта стоит проверять работоспособность кода - это можно сделать, нажав на панели Jupiter Hub Kernel и Restart & Run All

✔ __Комментарий от ревьюера №3__ Спасибо за правки

Шаг 1. Загрузите данные и подготовьте их к анализу

Загрузите данные о визитах, заказах и рекламных расходах из CSV-файлов в переменные. Пути к файлам • /datasets/visits_info_short.csv. Скачать датасет; • /datasets/orders_info_short.csv. Скачать датасет; • /datasets/costs_info_short.csv. Скачать датасет.

- Изучите данные и выполните предобработку. Есть ли в данных пропуски и дубликаты? Убедитесь, что типы данных во всех колонках соответствуют сохранённым в них значениям. Обратите внимание на столбцы с датой и временем.

In [1]:

```
import pandas as pd
from datetime import datetime, timedelta
import seaborn as sns
from matplotlib import pyplot as plt
import numpy as np
visits=pd.read_csv('/datasets/visits_info_short.csv')
orders=pd.read_csv('/datasets/orders_info_short.csv')
costs=pd.read_csv('/datasets/costs_info_short.csv')

print(visits.head())
print(orders.head())
print(costs.head())
```

```
       User Id         Region   Device  Channel         Session Start  \
0  981449118918  United States   iPhone  organic  2019-05-01 02:36:01
1  278965908054  United States   iPhone  organic  2019-05-01 04:46:31
2  590706206550  United States      Mac  organic  2019-05-01 14:09:25
3  326433527971  United States  Android   TipTop  2019-05-01 00:29:59
4  349773784594  United States      Mac  organic  2019-05-01 03:33:35

           Session End
0  2019-05-01 02:45:01
1  2019-05-01 04:47:35
2  2019-05-01 15:32:08
3  2019-05-01 00:54:25
4  2019-05-01 03:57:40
       User Id            Event Dt  Revenue
0  188246423999  2019-05-01 23:09:52     4.99
1  174361394180  2019-05-01 12:24:04     4.99
2  529610067795  2019-05-01 11:34:04     4.99
3  319939546352  2019-05-01 15:34:40     4.99
4  366000285810  2019-05-01 13:59:51     4.99
          dt   Channel  costs
0  2019-05-01  FaceBoom  113.3
1  2019-05-02  FaceBoom   78.1
2  2019-05-03  FaceBoom   85.8
3  2019-05-04  FaceBoom  136.4
4  2019-05-05  FaceBoom  122.1
```

✔

Комментарий от ревьюера №3

Для подгрузки данных можно использовать конструкцию try-except , она поможет избежать потенциальных ошибок при загрузке данных, связанных, например, с некорректным указанием путей.

Подробнее о конструкции по ссылке:

https://pythonworld.ru/tipy-dannyx-v-python/isklyucheniya-v-python-konstrukciya-try-except-dlya-obrabotki-isklyuchenij.html

Либо же можно использовать стандартную библиотеку os:

https://pythonworld.ru/moduli/modul-os.html

Несколько интересных статей кейсы использования конструкции:

https://www.programiz.com/python-programming/exception-handling

https://towardsdatascience.com/do-not-abuse-try-except-in-python-d9b8ee59e23b

https://www.techbeamers.com/use-try-except-python/

Как вариант в try можно указать корректные пути (в нашем случае глобальные) в except - некорректные (локальные). Можно также специфицровать тип ошибки, FileNotFoundError или задать кастомный тип ошибки (FilePathError, например)

Она полезна, если ты работаешь локально, а потом подгружаешь проект на платформу. Конструкция позволит не падать коду и локально, и на сервере ЯП, так как если не сработает один блок с путями, сработает другой.

Ну и вообще, в целом полезно про эту констуркцию знать, она универсальна и может быть использована в разных задачах.

In [2]:

```
# Проверяем на пропуски: чудно-пропусков нет
visits.isna().sum()
orders.isna().sum()
costs.isna().sum()
```

Out[2]:

```
dt         0
Channel    0
costs      0
dtype: int64
```

In [3]:

```
# Проверяем на дубликаты: волшебно-дубликатов нет
print(visits.duplicated().sum())
print(orders.duplicated().sum())
print(costs.duplicated().sum())
```

```
0
0
0
```

In [4]:

```
visits.info()
orders.info()
costs.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 309901 entries, 0 to 309900
Data columns (total 6 columns):
 #   Column         Non-Null Count   Dtype 
---  ------         --------------   ----- 
 0   User Id        309901 non-null  int64 
 1   Region         309901 non-null  object
 2   Device         309901 non-null  object
 3   Channel        309901 non-null  object
 4   Session Start  309901 non-null  object
 5   Session End    309901 non-null  object
dtypes: int64(1), object(5)
memory usage: 14.2+ MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40212 entries, 0 to 40211
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   User Id   40212 non-null  int64  
 1   Event Dt  40212 non-null  object 
 2   Revenue   40212 non-null  float64
dtypes: float64(1), int64(1), object(1)
memory usage: 942.6+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1800 entries, 0 to 1799
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   dt       1800 non-null   object 
 1   Channel  1800 non-null   object 
 2   costs    1800 non-null   float64
dtypes: float64(1), object(2)
memory usage: 42.3+ KB
```

In [5]:

```
visits['Session Start']=pd.to_datetime(visits['Session Start'])
visits['Session End']=pd.to_datetime(visits['Session End'])
orders['Event Dt']=pd.to_datetime(orders['Event Dt'])
costs['dt']=pd.to_datetime(costs['dt'])
```

In [6]:

```
visits.info()
orders.info()
costs.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 309901 entries, 0 to 309900
Data columns (total 6 columns):
 #   Column         Non-Null Count   Dtype         
---  ------         --------------   -----         
 0   User Id        309901 non-null  int64         
 1   Region         309901 non-null  object        
 2   Device         309901 non-null  object        
 3   Channel        309901 non-null  object        
 4   Session Start  309901 non-null  datetime64[ns]
 5   Session End    309901 non-null  datetime64[ns]
dtypes: datetime64[ns](2), int64(1), object(3)
memory usage: 14.2+ MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40212 entries, 0 to 40211
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype         
---  ------    --------------  -----         
 0   User Id   40212 non-null  int64         
 1   Event Dt  40212 non-null  datetime64[ns]
 2   Revenue   40212 non-null  float64       
dtypes: datetime64[ns](1), float64(1), int64(1)
memory usage: 942.6 KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1800 entries, 0 to 1799
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype         
---  ------   --------------  -----         
 0   dt       1800 non-null   datetime64[ns]
 1   Channel  1800 non-null   object        
 2   costs    1800 non-null   float64       
dtypes: datetime64[ns](1), float64(1), object(1)
memory usage: 42.3+ KB
```

In [7]:

```
# перевели в тип данных object чтобы можно было выполнять операции
```

```
visits['Session Start']=pd.to_datetime(visits['Session Start']).dt.date
costs['dt']=pd.to_datetime(costs['dt']).dt.date
orders['Event Dt']=pd.to_datetime(orders['Event Dt']).dt.date
```

✍ __Комментарий от ревьюера №3__ Отсутствует проверка на дубликаты и пропуски. Так же нету вывода о качестве исходных данных. Поправишь, пожалуйста?

Всё сделала!

ВЫВОД: дубликатов нет, пропусков нет. Преобразовали столбцы со временем в нужный тип данных.

✔ __Комментарий от ревьюера №4__ Хорошо, основные действия по предобработке сделаны.

Комментарий от ревьюера №3

- чтобы переименовать колонки, можно также использовать метод rename()
- если мы только привели преобразование в строчные символы и заменили пробел на знак _ , то мы можем использовать такой подход:

```
visits.columns = visits.columns.str.lower().str.replace(' ', '_')
```

также интересным является подход с использованием list comprehensions:

```
visits.columns = [x.lower().replace(' ', '_') for x in visits.columns.values]
```

- на стадии загрузки и подготовки данных к исследовательскому анализу, советую посмотреть на данные более детально, чтобы избежать невынужденных ошибок в дальнейшем. Чем большем мы знаем о данных - тем более корректны и обоснованы выглядят наши выводы. Такие проверки много времени не занимают, но зато мы можем лучше контролировать данные и их анализ. Например, мы можем:
  - проверить временной интервал на соответствие условию проекта, а также на возможные ошибки (например, проверить случаи, когда окончание сессии было раньше, и так далее);
  - для численных данных посмотреть на их статистические показатели и проверить их на наличие каких-то ошибок или аномалий (например, нули или отрицательные значения там, где они не должны быть).

Шаг 2. Задайте функции для расчёта и анализа LTV, ROI, удержания и конверсии.

Это функции для вычисления значений метрик: • get_profiles() — для создания профилей пользователей, • get_retention() — для подсчёта Retention Rate, • get_conversion() — для подсчёта конверсии, • get_ltv() — для подсчёта LTV. А также функции для построения графиков: • filter_data() — для сглаживания данных, • plot_retention() — для построения графика Retention Rate, • plot_conversion() — для построения графика конверсии, • plot_ltv_roi — для визуализации LTV и ROI.

In [8]:

```
# функция для создания пользовательских профилей

def get_profiles(visits, orders, costs):

    # находим параметры первых посещений
    profiles = (
        visits.sort_values(by=['User Id', 'Session Start'])
        .groupby('User Id')
        .agg(
            {
                'Session Start': 'first',
                'Channel': 'first',
                'Device': 'first',
                'Region': 'first',
            }
        )
        # время первого посещения назовём first_ts
        .rename(columns={'Session Start': 'first_ts'})
        .reset_index()  # возвращаем user_id из индекса
    )


    # для когортного анализа определяем дату первого посещения и первый день месяца,
    # в который это посещение произошло

    profiles['dt'] = profiles['first_ts']  # !важно добавить столбец с тем же названием что и в costs
    profiles['month'] = profiles['first_ts'].astype('datetime64[M]')
    # добавляем признак платящих пользователей
    profiles['payer'] = profiles['User Id'].isin(orders['User Id'].unique())
```

```
    # считаем количество уникальных пользователей с одинаковыми источником и датой привлечения
    new_users = (
        profiles.groupby(['dt', 'Channel'])
        .agg({'User Id': 'nunique'})
         # столбец с числом пользователей назовём unique_users
        .rename(columns={'User Id': 'unique_users'})
        .reset_index()                          # возвращаем dt и Channel из индексов
    )
     # от profiles оставили 3 колонки:'dt', 'Channel' и 'unique_users'



    # объединяем траты на рекламу и число привлечённых пользователей по дате и каналу привлечения
    costs = costs.merge(new_users, on=['dt', 'Channel'], how='left')


    # делим рекламные расходы на число привлечённых пользователей
    # результаты сохраним в столбец acquisition_cost (CAC)
    costs['acquisition_cost'] = costs['costs'] / costs['unique_users']

    # добавим стоимость привлечения в профили
    profiles = profiles.merge(

        costs[['dt', 'Channel', 'acquisition_cost','costs']],
        on=['dt', 'Channel'],
        how='left',

    )

     # органические пользователи не связаны с данными о рекламе,
     # поэтому в столбце acquisition_cost у них значения NaN
     # заменим их на ноль, ведь стоимость привлечения равна нулю
    profiles['acquisition_cost'] = profiles['acquisition_cost'].fillna(0)
    profiles['costs'] = profiles['costs'].fillna(0)
    return profiles
profiles = get_profiles(visits,orders,costs)
```

In [9]:

```
# функция для расчёта удержания

def get_retention(
    profiles,
    visits,
    observation_date,
    horizon_days,
    dimensions=[],
    ignore_horizon=False,
):

    # добавляем столбец payer в передаваемый dimensions список
    dimensions = ['payer'] + dimensions

    # исключаем пользователей, не «доживших» до горизонта анализа
    last_suitable_acquisition_date = observation_date
    if not ignore_horizon:
        last_suitable_acquisition_date = observation_date - timedelta(
            days=horizon_days - 1
        )
    result_raw = profiles.query('first_ts <= @last_suitable_acquisition_date')

    # собираем «сырые» данные для расчёта удержания
    result_raw = result_raw.merge(
        visits[['User Id', 'Session Start']], on='User Id', how='left'
    )
    result_raw['lifetime'] = (
        result_raw['Session Start'] - result_raw['first_ts']
    ).dt.days

    # функция для группировки таблицы по желаемым признакам
    def group_by_dimensions(df, dims, horizon_days):
        result = df.pivot_table(
            index=dims, columns='lifetime', values='User Id', aggfunc='nunique'
        )
        cohort_sizes = (
            df.groupby(dims)
            .agg({'User Id': 'nunique'})
            .rename(columns={'User Id': 'cohort_size'})
        )
        result = cohort_sizes.merge(result, on=dims, how='left').fillna(0)
        result = result.div(result['cohort_size'], axis=0)
```

```
    result = result[['cohort_size'] + list(range(horizon_days))]
    result['cohort_size'] = cohort_sizes
    return result

# получаем таблицу удержания
result_grouped = group_by_dimensions(result_raw, dimensions, horizon_days)

# получаем таблицу динамики удержания
result_in_time = group_by_dimensions(
    result_raw, dimensions + ['first_ts'], horizon_days
)

# возвращаем обе таблицы и сырые данные
return result_raw, result_grouped, result_in_time
```

In [10]:

```
# функция для расчёта конверсии

def get_conversion(
    profiles,
    orders,  # заменили visits
    observation_date,
    horizon_days,
    dimensions=[],
    ignore_horizon=False,
):

    # Шаг 1. Получить пользовательские профили и данные о покупках
    # передаём их в качестве аргументов profiles и orders

    # исключаем пользователей, не «доживших» до горизонта анализа
    last_suitable_acquisition_date = observation_date
    if not ignore_horizon:
        last_suitable_acquisition_date = observation_date - timedelta(
            days=horizon_days - 1
        )
    result_raw = profiles.query('first_ts <= @last_suitable_acquisition_date')

    # Шаг 2. Найти дату и время первой покупки для каждого пользователя

    first_purchases = (
        orders.sort_values(by=['User Id', 'Event Dt'])
        .groupby('User Id')
        .agg({'Event Dt': 'first'})
        .reset_index()
    )

    # Шаг 3. Добавить данные о покупках в профили
    result_raw = result_raw.merge(
        first_purchases[['User Id', 'Event Dt']], on='User Id', how='left'
    )

    # Шаг 4. Рассчитать лайф тайм для каждой покупки
    result_raw['lifetime'] = (
        result_raw['Event Dt'] - result_raw['first_ts']
    ).dt.days

    # функция для группировки таблицы по желаемым признакам
    def group_by_dimensions(df, dims, horizon_days):

        # Шаг 5. Построить таблицу конверсии
        result = df.pivot_table(
            index=dims, columns='lifetime', values='User Id', aggfunc='nunique'
        )

        # Шаг 6. Посчитать сумму с накоплением для каждой строки
        result = result.fillna(0).cumsum(axis = 1)

        # Шаг 7. Вычислить размеры когорт
        cohort_sizes = (
            df.groupby(dims)
            .agg({'User Id': 'nunique'})
            .rename(columns={'User Id': 'cohort_size'})
        )

        # Шаг 8. Объединить таблицы размеров когорт и конверсии
        result = cohort_sizes.merge(result, on=dims, how='left').fillna(0)

        # Шаг 9. Разделить каждую «ячейку» в строке на размер когорты
        result = result.div(result['cohort_size'], axis=0)
```

```
    # исключаем все лайф таймы, превышающие горизонт анализа
    result = result[['cohort_size'] + list(range(horizon_days))]
    # восстанавливаем размеры когорт
    result['cohort_size'] = cohort_sizes
    return result

# получаем таблицу конверсии
result_grouped = group_by_dimensions(result_raw, dimensions, horizon_days)

# получаем таблицу динамики конверсии
result_in_time = group_by_dimensions(
    result_raw, dimensions + ['first_ts'], horizon_days
)

# возвращаем обе таблицы и сырые данные
return result_raw, result_grouped, result_in_time
```

In [11]:

```
# функция для расчёта LTV и ROI

def get_ltv(
    profiles,                    # Шаг 1. Получить профили и данные о покупках
    orders,
    observation_date,
    horizon_days,
    dimensions=[],
    ignore_horizon=False,
):
    # исключаем пользователей, не «доживших» до горизонта анализа
    last_suitable_acquisition_date = observation_date
    if not ignore_horizon:
        last_suitable_acquisition_date = observation_date - timedelta(
            days=horizon_days - 1
        )
        result_raw = profiles.query('dt <= @last_suitable_acquisition_date')


    # Шаг 2. Добавляем данные о покупках в профили

    result_raw = result_raw.merge(
        orders[['User Id', 'Event Dt', 'Revenue']], on='User Id', how='left'
    )


    # Шаг 3. Рассчитать лайф тайм пользователя для каждой покупки

    result_raw['lifetime'] = (
        result_raw['Event Dt'] - result_raw['first_ts']
    ).dt.days

    # группируем по cohort, если в dimensions ничего нет

    if len(dimensions) == 0:
        result_raw['cohort'] = 'All users'
        dimensions = dimensions + ['cohort']


    # функция для группировки таблицы по желаемым признакам

    def group_by_dimensions(df, dims, horizon_days):

        # Шаг 4. Построить таблицу выручки
        # строим «треугольную» таблицу
        result = df.pivot_table(
            index=dims,
            columns='lifetime',
            values='Revenue', # в ячейках — выручка за каждый лайф тайм
            aggfunc='sum'
        )
        result.head()

        # Шаг 5. Посчитать сумму выручки с накоплением
        result = result.fillna(0).cumsum(axis=1)

        # Шаг 6. Вычислить размеры когорт
        cohort_sizes = (
            df.groupby(dims)
            .agg({'User Id': 'nunique'})
            .rename(columns={'User Id': 'cohort_size'})
        )
```

```
    # Шаг 7. Объединить размеры когорт и таблицу выручки
    result = cohort_sizes.merge(result, on=dims, how='left').fillna(0)



    # Шаг 8. Посчитать LTV - делим каждую «ячейку» в строке на размер когорты
    # LTV = общая выручка(с накоплением) на тек.день / размер когорты

    result = result.div(result['cohort_size'], axis=0)

    # исключаем все лайф таймы, превышающие горизонт анализа
    result = result[['cohort_size'] + list(range(horizon_days))]

     # восстанавливаем размеры когорт
    result['cohort_size'] = cohort_sizes

    # сохраняем в датафрейм данные пользователей и значения CAC,
    # добавив параметры из dimensions
    cac = df[['User Id', 'acquisition_cost'] + dims].drop_duplicates()

    # считаем средний CAC по параметрам из dimensions
    cac = (
        cac.groupby(dims)
        .agg({'acquisition_cost': 'mean'})
        .rename(columns={'acquisition_cost': 'cac'})
    )

    # считаем ROI: делим LTV на CAC
    roi = result.div(cac['cac'], axis=0)

    # удаляем строки с бесконечным ROI
    roi = roi[~roi['cohort_size'].isin([np.inf])]

    # восстанавливаем размеры когорт в таблице ROI
    roi['cohort_size'] = cohort_sizes

    # добавляем CAC в таблицу ROI
    roi['cac'] = cac['cac']

    # в финальной таблице оставляем размеры когорт, CAC
    # и ROI в лайф таймы, не превышающие горизонт анализа
    roi = roi[['cohort_size', 'cac'] + list(range(horizon_days))]

    # возвращаем таблицы LTV и ROI
    return result, roi


# получаем таблицы LTV и ROI
result_grouped, roi_grouped = group_by_dimensions(    # result_grouped-таблица LTV
    result_raw, dimensions, horizon_days

)

# для таблиц динамики убираем 'cohort' из dimensions
if 'cohort' in dimensions:
    dimensions = []

 # получаем таблицы динамики LTV и ROI
result_in_time, roi_in_time = group_by_dimensions(
    result_raw, dimensions + ['dt'], horizon_days
)

return (
    result_raw,  # сырые данные
    result_grouped,  # таблица LTV
    result_in_time,  # таблица динамики LTV
    roi_grouped,  # таблица ROI
    roi_in_time,  # таблица динамики ROI
)
```

А также функции для визуализации этих метрик — filter_data(), plot_retention(), plot_conversion() и plot_ltv_roi().

In [12]:

```
# функция для сглаживания фрейма
def filter_data(df, window):
    # для каждого столбца применяем скользящее среднее
    for column in df.columns.values:
        df[column] = df[column].rolling(window).mean()
    return df
```

In [13]:

```
# функция для визуализации удержания
```

```
def plot_retention(retention, retention_history, horizon, window=7):

    # задаём размер сетки для графиков
    plt.figure(figsize=(15, 10))

    # исключаем размеры когорт и удержание первого дня
    retention = retention.drop(columns=['cohort_size', 0])
    # в таблице динамики оставляем только нужный лайфтайм
    retention_history = retention_history.drop(columns=['cohort_size'])[
        [horizon - 1]
    ]

    # если в индексах таблицы удержания только payer,
    # добавляем второй признак — cohort
    if retention.index.nlevels == 1:
        retention['cohort'] = 'All users'
        retention = retention.reset_index().set_index(['cohort', 'payer'])

    # в таблице графиков — два столбца и две строки, четыре ячейки
    # в первой строим кривые удержания платящих пользователей
    ax1 = plt.subplot(2, 2, 1)
    retention.query('payer == True').droplevel('payer').T.plot(
        grid=True, ax=ax1
    )
    plt.legend()
    plt.xlabel('Лайфтайм')
    plt.title('Удержание платящих пользователей')

    # во второй ячейке строим кривые удержания неплатящих
    # вертикальная ось — от графика из первой ячейки
    ax2 = plt.subplot(2, 2, 2, sharey=ax1)
    retention.query('payer == False').droplevel('payer').T.plot(
        grid=True, ax=ax2
    )
    plt.legend()
    plt.xlabel('Лайфтайм')
    plt.title('Удержание неплатящих пользователей')

    # в третьей ячейке — динамика удержания платящих
    ax3 = plt.subplot(2, 2, 3)
    # получаем названия столбцов для сводной таблицы
    columns = [
        name
        for name in retention_history.index.names
        if name not in ['first_ts', 'payer']
    ]
    # фильтруем данные и строим график
    filtered_data = retention_history.query('payer == True').pivot_table(
        index='first_ts', columns=columns, values=horizon - 1, aggfunc='mean'
    )
    filter_data(filtered_data, window).plot(grid=True, ax=ax3)
    plt.xlabel('Дата привлечения')
    plt.title(
        'Динамика удержания платящих пользователей на {}-й день'.format(
            horizon
        )
    )

    # в чётвертой ячейке — динамика удержания неплатящих
    ax4 = plt.subplot(2, 2, 4, sharey=ax3)
    # фильтруем данные и строим график
    filtered_data = retention_history.query('payer == False').pivot_table(
        index='first_ts', columns=columns, values=horizon - 1, aggfunc='mean'
    )
    filter_data(filtered_data, window).plot(grid=True, ax=ax4)
    plt.xlabel('Дата привлечения')
    plt.title(
        'Динамика удержания неплатящих пользователей на {}-й день'.format(
            horizon
        )
    )

    plt.tight_layout()
    plt.show()
```

In [14]:

```
# функция для визуализации конверсии

def plot_conversion(conversion, conversion_history, horizon, window=7):

    # задаём размер сетки для графиков
```

```
    plt.figure(figsize=(15, 5))

    # исключаем размеры когорт
    conversion = conversion.drop(columns=['cohort_size'])
    # в таблице динамики оставляем только нужный лайфтайм
    conversion_history = conversion_history.drop(columns=['cohort_size'])[
        [horizon - 1]
    ]

    # первый график — кривые конверсии
    ax1 = plt.subplot(1, 2, 1)
    conversion.T.plot(grid=True, ax=ax1)
    plt.legend()
    plt.xlabel('Лайфтайм')
    plt.title('Конверсия пользователей')

    # второй график — динамика конверсии
    ax2 = plt.subplot(1, 2, 2, sharey=ax1)
    columns = [
        # столбцами сводной таблицы станут все столбцы индекса, кроме даты
        name for name in conversion_history.index.names if name not in ['dt']
    ]
    filtered_data = conversion_history.pivot_table(
        index='first_ts', columns=columns, values=horizon - 1, aggfunc='mean'
    )
    filter_data(filtered_data, window).plot(grid=True, ax=ax2)
    plt.xlabel('Дата привлечения')
    plt.title('Динамика конверсии пользователей на {}-й день'.format(horizon))

    plt.tight_layout()
    plt.show()
```

In [15]:

```
# функция для визуализации LTV и ROI

def plot_ltv_roi(ltv, ltv_history, roi, roi_history, horizon, window=7):

    # задаём сетку отрисовки графиков
    plt.figure(figsize=(20, 10))

    # из таблицы ltv исключаем размеры когорт
    ltv = ltv.drop(columns=['cohort_size'])
    # в таблице динамики ltv оставляем только нужный лайфтайм
    ltv_history = ltv_history.drop(columns=['cohort_size'])[[horizon - 1]]

    # стоимость привлечения запишем в отдельный фрейм
    cac_history = roi_history[['cac']]

    # из таблицы roi исключаем размеры когорт и cac
    roi = roi.drop(columns=['cohort_size', 'cac'])
    # в таблице динамики roi оставляем только нужный лайфтайм
    roi_history = roi_history.drop(columns=['cohort_size', 'cac'])[
        [horizon - 1]
    ]

    # первый график — кривые ltv
    ax1 = plt.subplot(2, 3, 1)
    ltv.T.plot(grid=True, ax=ax1)
    plt.legend()
    plt.xlabel('Лайфтайм')
    plt.title('LTV')

    # второй график — динамика ltv
    ax2 = plt.subplot(2, 3, 2, sharey=ax1)
    # столбцами сводной таблицы станут все столбцы индекса, кроме даты
    columns = [name for name in ltv_history.index.names if name not in ['dt']]
    filtered_data = ltv_history.pivot_table(
        index='dt', columns=columns, values=horizon - 1, aggfunc='mean'
    )
    filter_data(filtered_data, window).plot(grid=True, ax=ax2)
    plt.xlabel('Дата привлечения')
    plt.title('Динамика LTV пользователей на {}-й день'.format(horizon))

    # третий график — динамика cac
    ax3 = plt.subplot(2, 3, 3, sharey=ax1)
    # столбцами сводной таблицы станут все столбцы индекса, кроме даты
    columns = [name for name in cac_history.index.names if name not in ['dt']]
    filtered_data = cac_history.pivot_table(
        index='dt', columns=columns, values='cac', aggfunc='mean'
    )
    filter_data(filtered_data, window).plot(grid=True, ax=ax3)
```

```
    plt.xlabel('Дата привлечения')
    plt.title('Динамика стоимости привлечения пользователей')

    # четвёртый график — кривые roi
    ax4 = plt.subplot(2, 3, 4)
    roi.T.plot(grid=True, ax=ax4)
    plt.axhline(y=1, color='red', linestyle='--', label='Уровень окупаемости')
    plt.legend()
    plt.xlabel('Лайфтайм')
    plt.title('ROI')

    # пятый график — динамика roi
    ax5 = plt.subplot(2, 3, 5, sharey=ax4)
    # столбцами сводной таблицы станут все столбцы индекса, кроме даты
    columns = [name for name in roi_history.index.names if name not in ['dt']]
    filtered_data = roi_history.pivot_table(
       index='dt', columns=columns, values=horizon - 1, aggfunc='mean'
    )
    filter_data(filtered_data, window).plot(grid=True, ax=ax5)
    plt.axhline(y=1, color='red', linestyle='--', label='Уровень окупаемости')
    plt.xlabel('Дата привлечения')
    plt.title('Динамика ROI пользователей на {}-й день'.format(horizon))

    plt.tight_layout()
    plt.show()
```

✔

Комментарий от ревьюера №3

Хорошо, все необходимые функции были заданы, можно приступать к расчета и анализу

3. Исследовательский анализ данных

visits_info_short.csv:

User Id — уникальный идентификатор пользователя, Region — страна пользователя, Device — тип устройства пользователя, Channel — идентификатор источника перехода, Session Start — дата и время начала сессии, Session End — дата и время окончания сессии.

In [16]:

```
print(visits.head(2))
print(orders.head(2))
print(costs.head(2))
```

```
       User Id         Region  Device  Channel Session Start  \
0  981449118918  United States  iPhone  organic    2019-05-01
1  278965908054  United States  iPhone  organic    2019-05-01

         Session End
0 2019-05-01 02:45:01
1 2019-05-01 04:47:35
       User Id    Event Dt  Revenue
0  188246423999  2019-05-01     4.99
1  174361394180  2019-05-01     4.99
         dt   Channel  costs
0  2019-05-01  FaceBoom  113.3
1  2019-05-02  FaceBoom   78.1
```

3.1) Составьте профили пользователей. Определите минимальную и максимальную даты привлечения пользователей.

In [17]:

```
# Определяем доступный интервал привлечения пользователей
min_date=profiles['first_ts'].min()
observation_date = profiles['first_ts'].max() # это момент анализа
print(min_date,observation_date)
```

2019-05-01 2019-10-27

Это говорит что в таблице profiles есть данные с 01 мая по 27 октября 2019 года.

Если мы берём дату отсчёта - 1 ноября 2019 года,если гризонт анализа 14 дней, то включать в анализ можно только пользователей, пришедших не позднее 18 октября 2019 года.

✍ __Комментарий от ревьюера №3__ Немного не так. В данном шаге мы сравниваем исходные даты из таблицы профилей с датами ТЗ заказчика. Горизон и момент анализа мы используем только в 5 разделе, ведь именно там мы смотрим окупаемость за 2 недели

✍ __Комментарий от ревьюера №3__ Горизонт анализа и момент указаны в ТЗ

✍ __Комментарий от ревьюера №4__ В целом верный подход, но как описал выше - нам нужно рассмотреть из таблицы `профилей`, а ты смотришь из `visits`

Исправила)

__Комментарий от ревьюера №5__ Всё верно. Но отсутствует вывод, иначе не совсем понятно, зачем мы смотрели на эти даты (например, соответствую ли даты ТЗ или нет).

3.2) Выясните, из каких стран пользователи приходят в приложение и на какую страну приходится больше всего платящих пользователей. - Постройте таблицу, отражающую количество пользователей и долю платящих из каждой страны.

In [18]:

```
profiles.head(1)
```

Out[18]:

	User Id	first_ts	Channel	Device	Region	dt	month	payer	acquisition_cost	costs
0	599326	2019-05-07	FaceBoom	Mac	United States	2019-05-07	2019-05-01	True	1.088172	101.2

In [21]:

```
prof_country=100 * profiles.groupby('Region').agg({'payer':'mean'}).sort_values(by='payer', ascending = False)
prof_country
```

Out[21]:

	payer
Region	
United States	6.901862
Germany	4.111875
UK	3.982930
France	3.799427

__Комментарий от ревьюера №1__ Давай, полученные в этом разделе значения переведем в %? (умножим на 100), чтобы коллегам и заказчику было удобнее понимать суть данных

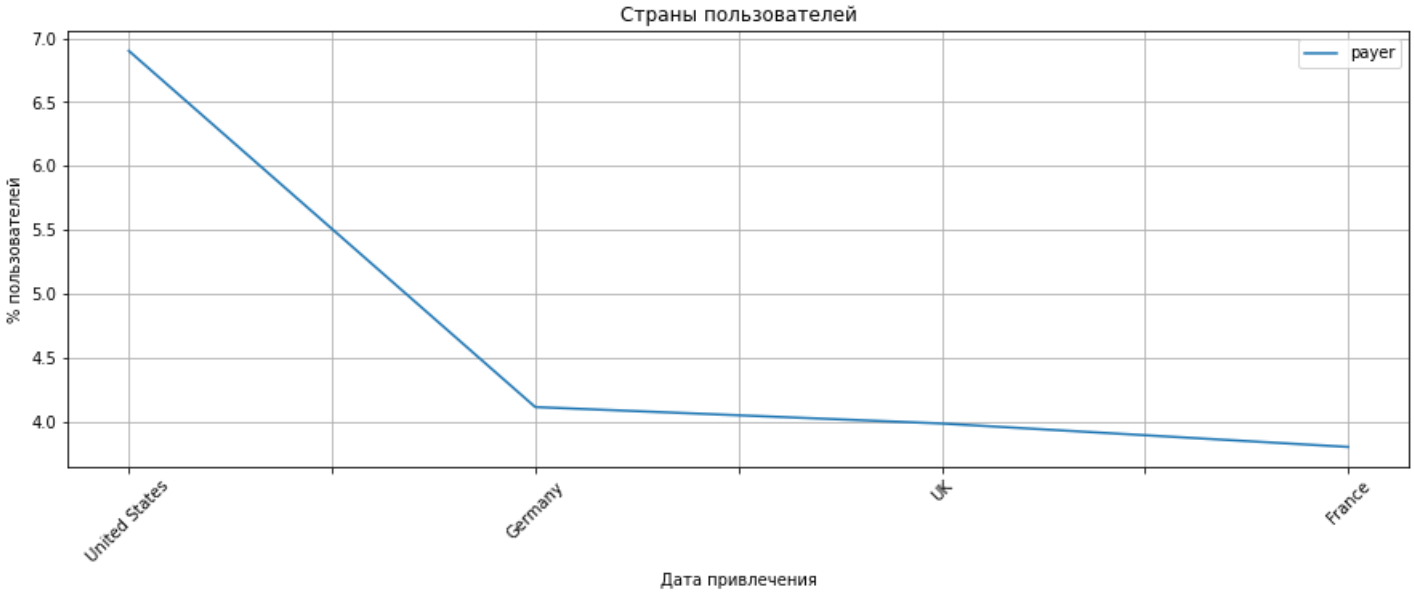
In [19]:

```
import matplotlib.pyplot as plt
%matplotlib inline

profiles.pivot_table(
    index='first_ts',
    columns='Region',
    values='payer',
    aggfunc='mean'
).plot(figsize = (13,5), grid=True)

plt.xlabel('Дата привлечения')
plt.title('Страны пользователей')
plt.show()
```

In [22]:

```
prof_country.plot(grid=True, figsize=(15,5))
plt.title('Страны пользователей')
plt.xlabel('Дата привлечения')
plt.xticks(rotation = 45)
plt.ylabel('% пользователей')
```

```
plt.show()
```



ВЫВОД:Больше всего платящих пользователей в США, потом Германия.

✍ __Комментарий от ревьюера №1__ График хорошо бы сгруппировать перед построением. Также отсутствует его оформление (заголовки + подписи осей). Поправь, пожалуйста, этот момент во всем проекте. Чтобы дальше на этом не акцентировать внимание

- 3.3 Узнайте, какими устройствами пользуются клиенты и какие устройства предпочитают платящие пользователи. Постройте таблицу, отражающую количество пользователей и долю платящих для каждого устройства.

Устройства- Device - из датафрейм visits Платящие пользователи - User Id - из датафрейма orders. С помощью функции по созданию профилей - добавляем 2 элемента. И по примеру выше-делаем тоже самое, только меняем на Device.

orders:

User Id — уникальный идентификатор пользователя, Event Dt — дата и время покупки, Revenue — сумма заказа.

visits:

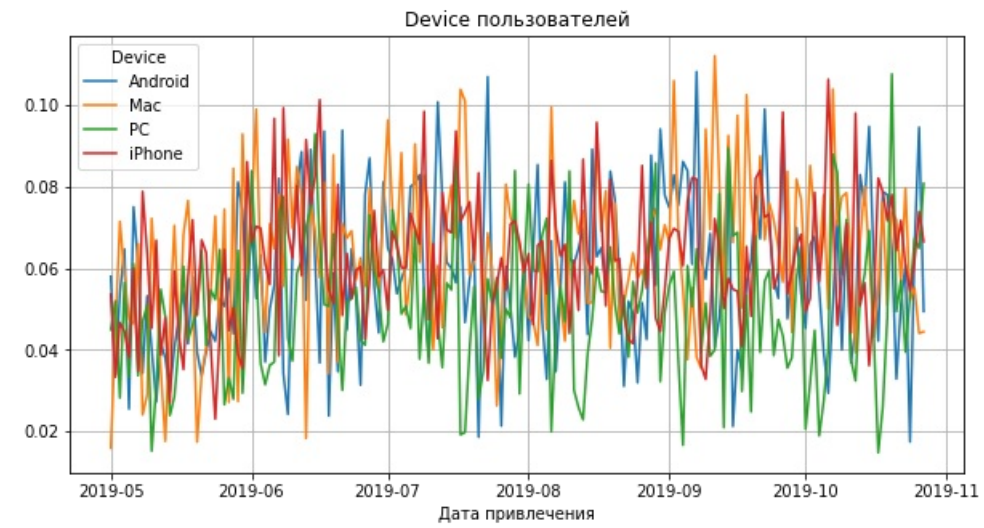
User Id — уникальный идентификатор пользователя, Region — страна пользователя, Device — тип устройства пользователя, Channel — идентификатор источника перехода, Session Start — дата и время начала сессии, Session End — дата и время окончания сессии.

In [23]:

```
prof_device=100 * profiles.groupby('Device').agg({'payer':'mean'}).sort_values(by='payer', ascending = False)
```

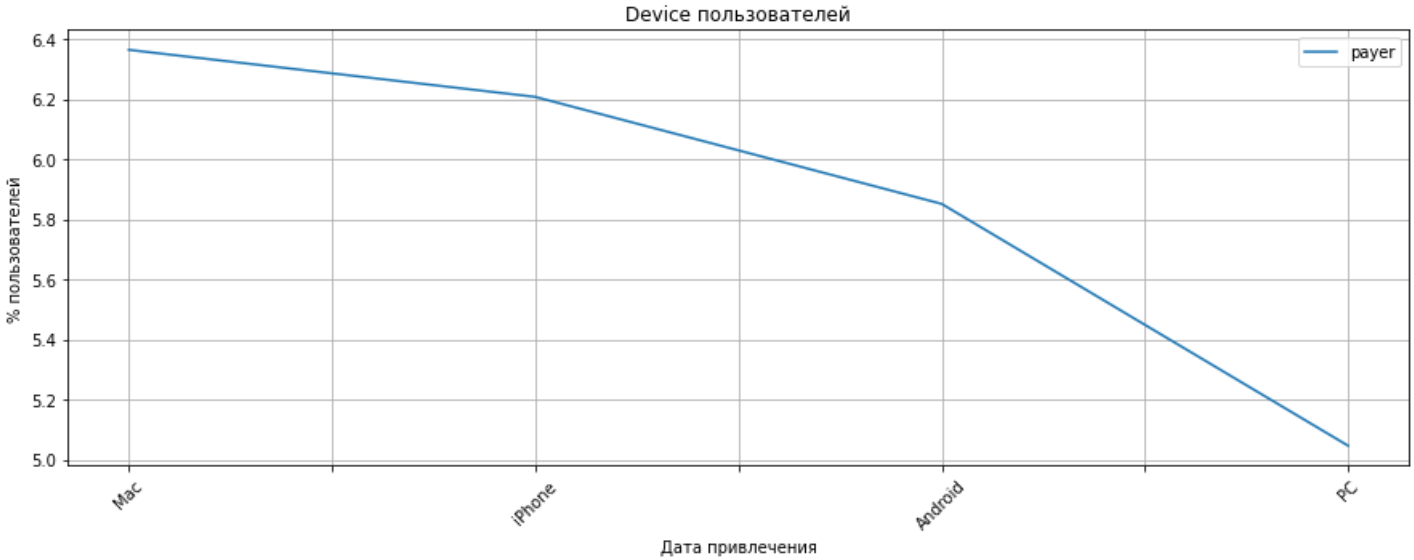
In [24]:

```
import matplotlib.pyplot as plt
%matplotlib inline

profiles.pivot_table(
    index='first_ts',
    columns='Device',
    values='payer',
    aggfunc='mean'
).plot(figsize = (10,5), grid=True)

plt.xlabel('Дата привлечения')
plt.title('Device пользователей')

plt.show()
```



In [25]:

```
prof_device.plot(grid=True, figsize=(15,5))

plt.title('Device пользователей')
plt.xlabel('Дата привлечения')
plt.xticks(rotation = 45)
plt.ylabel('% пользователей')

plt.show()
```



Получились устройства -Mac и iPhone- больше всего платящих

- 3.4 Изучите рекламные источники привлечения и определите каналы, из которых пришло больше всего платящих пользователей. Постройте таблицу, отражающую количество пользователей и долю платящих для каждого канала привлечения.

Идентификатор рекламного источника - Channel из таблицы costs. Вторая таблица должна быть orders и 3 таблица

visits:

User Id — уникальный идентификатор пользователя, Region — страна пользователя, Device — тип устройства пользователя, Channel — идентификатор источника перехода, Session Start — дата и время начала сессии, Session End — дата и время окончания сессии.

costs:

dt — дата проведения рекламной кампании, Channel — идентификатор рекламного источника, costs — расходы на эту кампанию.

orders:

User Id — уникальный идентификатор пользователя, Event Dt — дата и время покупки, Revenue — сумма заказа.

- ▪ profiles['payer'] = profiles['User Id'].isin(orders['User Id'].unique())

orders['User Id'] - сначала берем всех пользователей совершивших заказы, далее оставляем только уникальные: orders['User Id'].unique() и далее проверяем, входит ли очередной пользователь из датафрейма profiles в этот наш уникальный список. В результатирующем столбце получим True, если входит False -если нет.

In [26]:

```
prof_chanel=100 * profiles.groupby('Channel').agg({'payer':'mean'}).sort_values(by='payer', ascending = False)
```

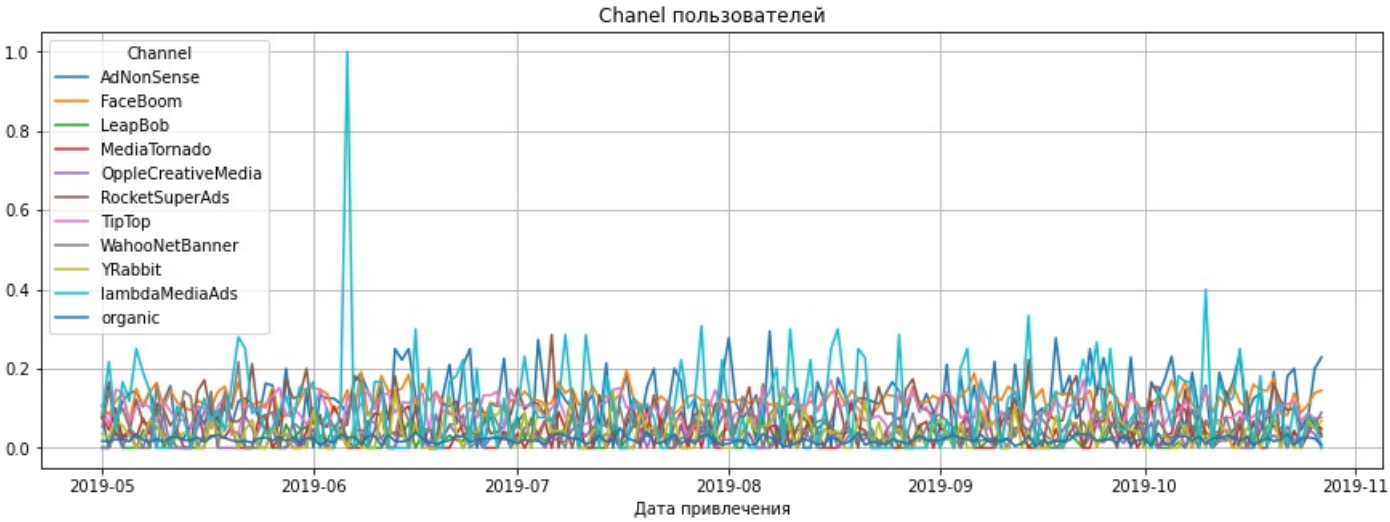
prof_chanel

Out[26]:

Channel	payer
FaceBoom	12.204914
AdNonSense	11.340206
lambdaMediaAds	10.469986
TipTop	9.600736
RocketSuperAds	7.913669
WahooNetBanner	5.296387
YRabbit	3.826531
MediaTornado	3.574702
LeapBob	3.063253
OppleCreativeMedia	2.707728
organic	2.055316

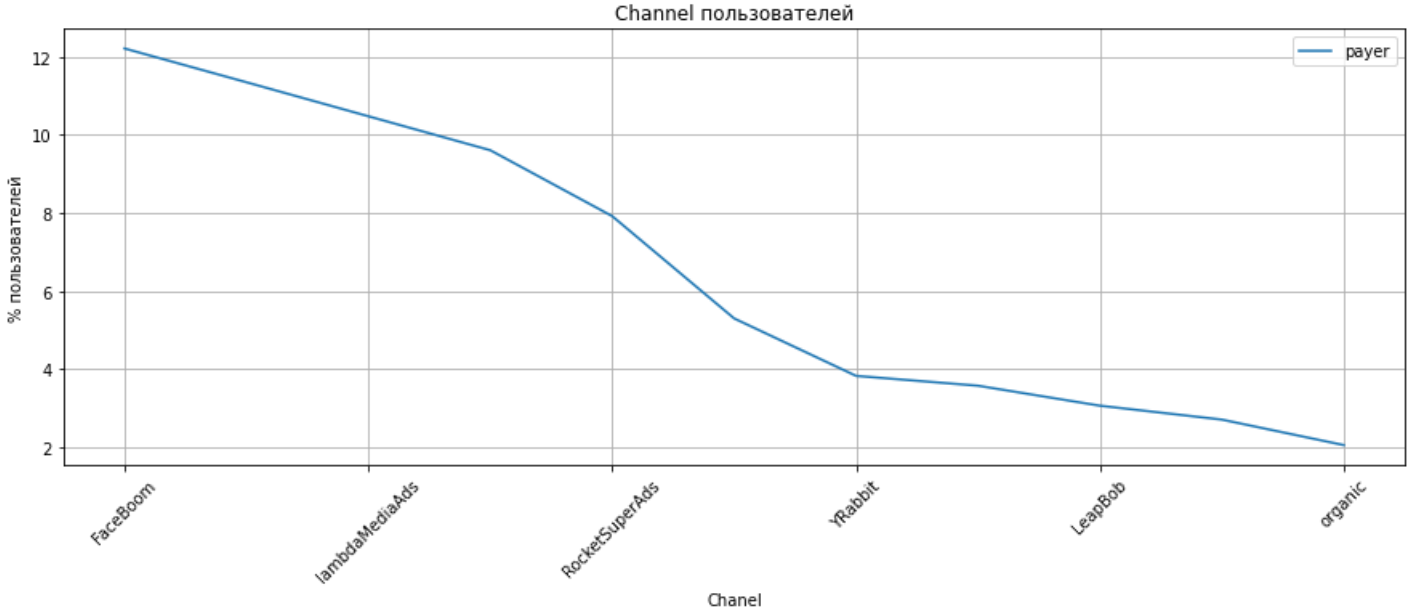
In [27]:

```
import matplotlib.pyplot as plt
%matplotlib inline

profiles.pivot_table(
    index='first_ts',
    columns='Channel',
    values='payer',
    aggfunc='mean'
).plot(figsize = (15,5), grid=True)

plt.xlabel('Дата привлечения')
plt.title('Chanel пользователей')

plt.show()

# Больше всего платящих пользователей было из FaceBoom и lambdaMediaAds.
```



In [27]:

```
prof_chanel.plot(grid=True, figsize=(15,5))

plt.title('Channel пользователей')
plt.xlabel('Chanel')
plt.xticks(rotation = 45)
plt.ylabel('% пользователей')

plt.show()
```



Вывод: Больше всего платящих пользователей было из FaceBoom и lambdaMediaAds. Получились устройства -Mac и iPhone- больше всего платящих. Больше всего платящих пользователей в США, потом Германия.

✍ __Комментарий от ревьюера №3__ Расчёт верный. Но нету общего вывода по разделу. Поправишь, пожалуйста, этот момент во всем проекте, чтобы дальше не акцентировать на этом внимаение?

✔ __Комментарий от ревьюера №4__ Отличный и наглядный график Здорово, когда он подписан. Так быстрее понять о чем идёт речь на нём.

ШАГ 4

Шаг 4. Маркетинг

- Посчитайте общую сумму расходов на маркетинг.
- Выясните, как траты распределены по рекламным источникам, то есть сколько денег потратили на каждый источник.
- Постройте визуализацию динамики изменения расходов во времени (по неделям и месяцам) по каждому источнику. Постарайтесь отразить это на одном графике.
- Узнайте, сколько в среднем стоило привлечение одного пользователя (CAC) из каждого источника. Используйте профили пользователей.
- Напишите промежуточные выводы.

In [28]:

```
profiles.head(1)
```

Out[28]:

	User Id	first_ts	Channel	Device	Region	dt	month	payer	acquisition_cost	costs
0	599326	2019-05-07	FaceBoom	Mac	United States	2019-05-07	2019-05-01	True	1.088172	101.2

In [59]:

```
profiles['acquisition_cost'].sum()
```

Out[59]:

105497.3

✔ __Комментарий от ревьюера №5__ Да, теперь корректно) Спасибо за правку

✍ __Комментарий от ревьюера №4__ А откуда ты взяла столбец `costs` в таблице профилей? Нам в исходных функциях ничего не нужно менять и добавлять --- P.S. расчёты ниже связанные с расходом некорректны

странный вопрос-что значит откуда. таблица profiles состоит из 3 таблиц-и costs конечно же туда входит. Тогда как надо?

✍ __Комментарий от ревьюера №5__ В исходной функции такой строки ячейки нету. Вероятно ты сама её добавила? Нам нужны рассчёты из столбца `acquisition_cost` ![image.png](attachment:image.png) --- Вон например, в тренажере - урок "Разбор кейчас" ![image-2.png](attachment:image-2.png)

In [28]:

```
profiles.head(1)
```

Out[28]:

	User Id	first_ts	Channel	Device	Region	dt	month	payer	acquisition_cost	costs
0	599326	2019-05-07	FaceBoom	Mac	United States	2019-05-07	2019-05-01	True	1.088172	101.2

In [69]:

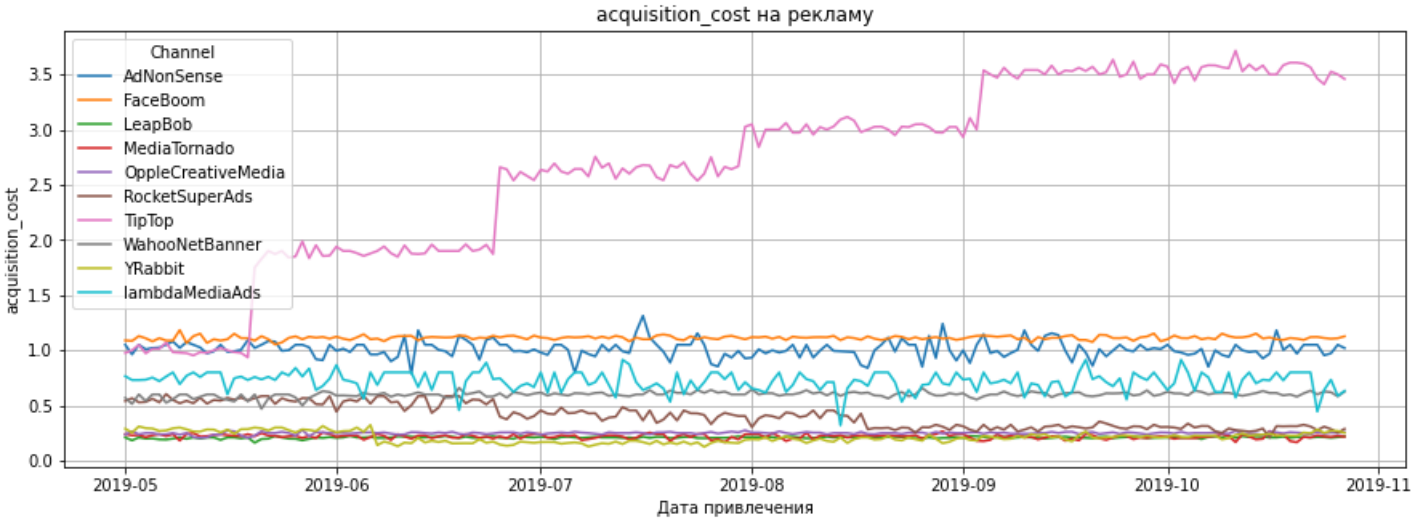
```
prof_costs=profiles.groupby('Channel').agg({'acquisition_cost':'mean'}).sort_values(by='acquisition_cost', ascending = False)
prof_costs
```

Out[69]:

Channel	acquisition_cost
TipTop	2.799003
FaceBoom	1.113286
AdNonSense	1.008054
lambdaMediaAds	0.724802
WahooNetBanner	0.602245
RocketSuperAds	0.412095
OppleCreativeMedia	0.250000
YRabbit	0.218975
MediaTornado	0.218717
LeapBob	0.210172

In [71]:

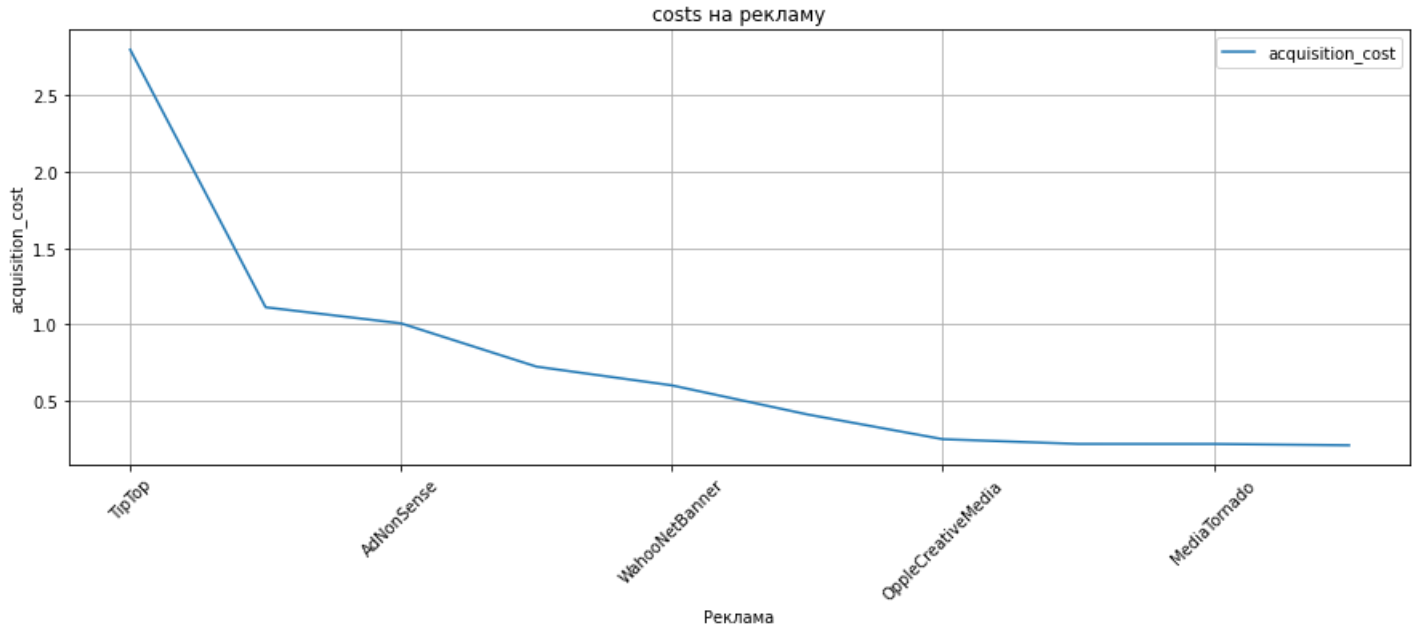
```
import matplotlib.pyplot as plt
%matplotlib inline

profiles.pivot_table(
    index='first_ts',
    columns='Channel',
    values='acquisition_cost',
    aggfunc='mean'
).plot(figsize = (15,5), grid=True)

plt.xlabel('Дата привлечения')
plt.title('acquisition_cost на рекламу')
plt.ylabel('acquisition_cost')
plt.show()
# Больше всего расходов на TipTop и FaceBoom.
# Но с начала октября эти расходы падают до середины октября
```



У всех каналов - рекламы- расходы одинаковые, кроме TipTop-тут расходы растут с середины мая 2019 года

In [79]:

```
prof_costs.plot(grid=True, figsize=(15,5))

plt.title('costs на рекламу')
plt.xlabel('Реклама')
plt.xticks(rotation = 45)
plt.ylabel('acquisition_cost')

plt.show()
```



Вывод: Всего на рекламу было потрачено- 13018376.490000004. Из них больше всего на рекламу от рекламного источника TipTop. На органические пользователи(кот.сами пришли)-мы ничего не тратили.

In [80]:

```
# Динамика по месяцам, неделям
profiles['month']= profiles['first_ts'].astype('datetime64[M]')
profiles['week']= profiles['first_ts'].astype('datetime64[W]')
profiles['week']
```

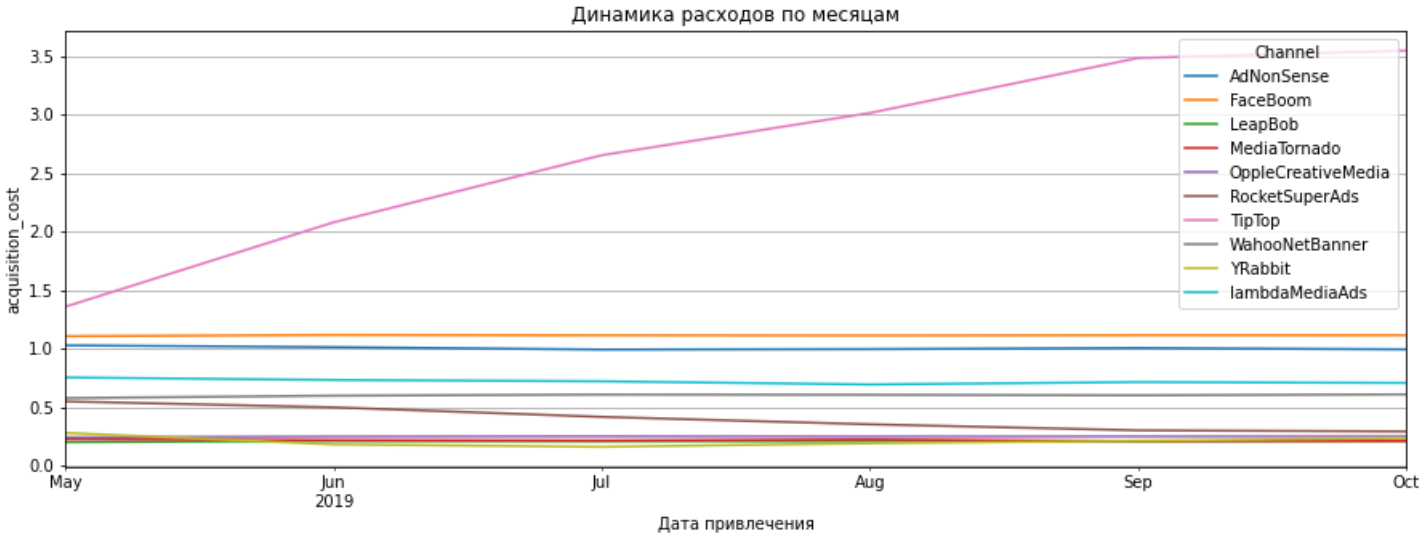
```
/tmp/ipykernel_31/3951410324.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  profiles['month']= profiles['first_ts'].astype('datetime64[M]')
/tmp/ipykernel_31/3951410324.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  profiles['week']= profiles['first_ts'].astype('datetime64[W]')
```

Out[80]:

```
0       2019-05-02
1       2019-07-04
3       2019-08-22
4       2019-09-26
7       2019-06-27
           ...
149999  2019-05-23
150001  2019-08-08
150003  2019-09-26
150005  2019-07-18
150006  2019-09-26
Name: week, Length: 93569, dtype: datetime64[ns]
```

In [81]:

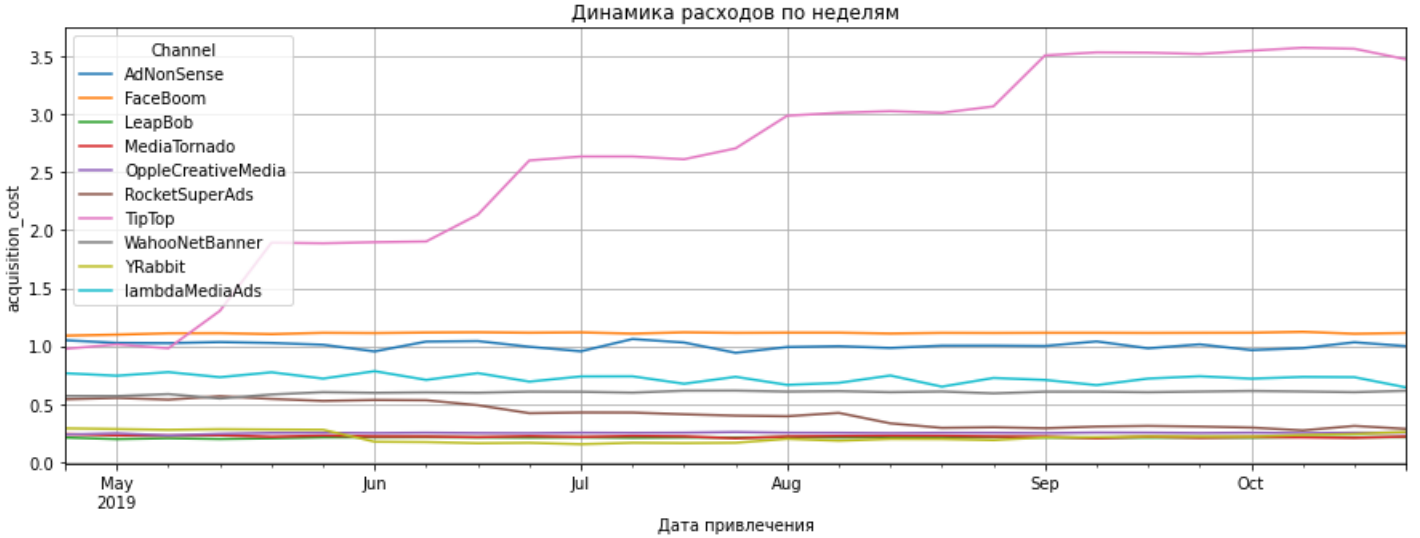
```
import matplotlib.pyplot as plt
%matplotlib inline

profiles.pivot_table(
    index='month',
    columns='Channel',
    values='acquisition_cost',
    aggfunc='mean'
).plot(figsize = (15,5), grid=True)

plt.xlabel('Дата привлечения')
plt.title('Динамика расходов по месяцам')
plt.ylabel('acquisition_cost')

plt.show()
```



✍ __Комментарий от ревьюера №4__ Оформление графиков (подпись оси ординат тут и ниже)

In [82]:

```
import matplotlib.pyplot as plt
%matplotlib inline

profiles.pivot_table(
    index='week',
    columns='Channel',
    values='acquisition_cost',
    aggfunc='mean'
).plot(figsize = (15,5), grid=True)

plt.xlabel('Дата привлечения')
plt.title('Динамика расходов по неделям')
plt.ylabel('acquisition_cost')
plt.show()
```



Вывод: по динамике расходов по месяцам и неделям-растут расходы на 1 канал привлечения-Tip Top.

✔ __Комментарий от ревьюера №5__ Отличные и наглядные графики Здорово, когда они подписаны. Так быстрее понять о чем идёт речь на нём.

✍ __Комментарий от ревьюера №3__ Нет решения по шагу 4.1 - 4.2. Добавь, пожалуйста

✍ __Комментарий от ревьюера №1__ Нет решения по одному из пунктов задания (из брифа) - визуализировать динамику расходов по каналам. По месяцам (с помощью метода `df[''].astype('datetime64[M]')` привести к месячной дате.). Добавишь, пожалуйста? --- А с помощью `[W]` - недельной

Чтобы рассчитать и проанализировать CAC, добавим данные о тратах на рекламу в пользовательские профили. Наше приложение хранит данные о рекламных расходах в costs.

Как рассчитать CAC в Python

- Передать функции для создания профилей данные о тратах на рекламу.
- Объединить данные о тратах на рекламу и новых пользователях.
- Вычислить CAC: разделить рекламные расходы на количество новых пользователей.
- Добавить CAC для каждой даты привлечения и источника в профили.

costs:

dt — дата проведения рекламной кампании, Channel — идентификатор рекламного источника, costs — расходы на эту кампанию.

orders:

User Id — уникальный идентификатор пользователя, Event Dt — дата и время покупки, Revenue — сумма заказа.

In [40]:

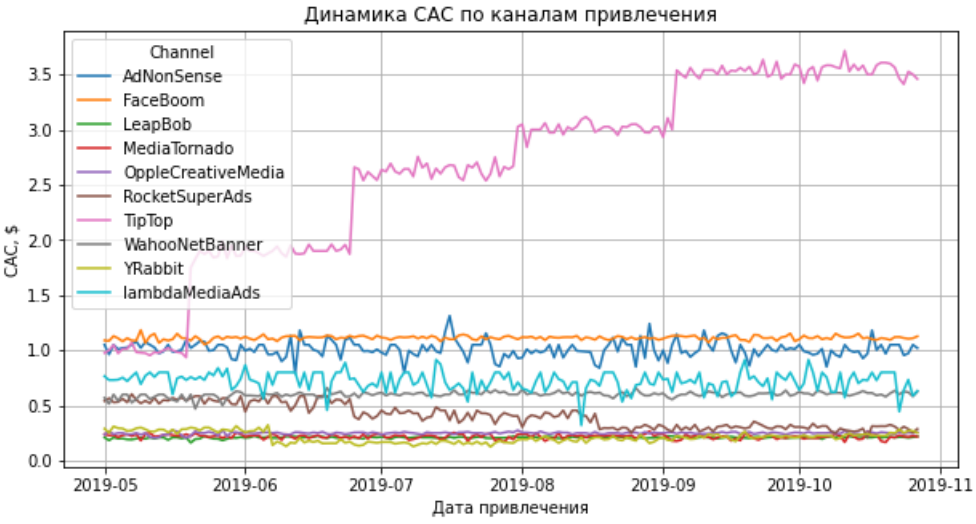
```
profiles.head(1)
```

Out[40]:

	User Id	first_ts	Channel	Device	Region	dt	month	payer	acquisition_cost	costs	week
0	599326	2019-05-07	FaceBoom	Mac	United States	2019-05-07	2019-05-01	True	1.088172	101.2	2019-05-02

In [76]:

```
# строим график истории изменений CAC (расходы на рекламу) по каналам привлечения

profiles.pivot_table(
    index='dt', columns='Channel', values='acquisition_cost', aggfunc='mean'
).plot(grid=True, figsize=(10, 5))
plt.ylabel('CAC, $')
plt.xlabel('Дата привлечения')
plt.title('Динамика CAC по каналам привлечения')
plt.show()
```



✔ __Комментарий от ревьюера №4__ Здорово, что визуализировла динамику CAC по каналам

Стоимость привлечения «органических» пользователей во всех когортах равна нулю, потому что они перешли на сайт приложения самостоятельно, а не благодаря рекламе. Реклама всех каналов адекватна кроме-Tip Top-огромные расходы на рекламу.

In [77]:

```
# выведем сводную таблицу, в которой расчитаем средний CAC для каждого канала

cac_chanel=profiles.groupby('Channel').agg({'acquisition_cost':'mean'}).sort_values(by='acquisition_cost', ascending = False)
cac_chanel
```

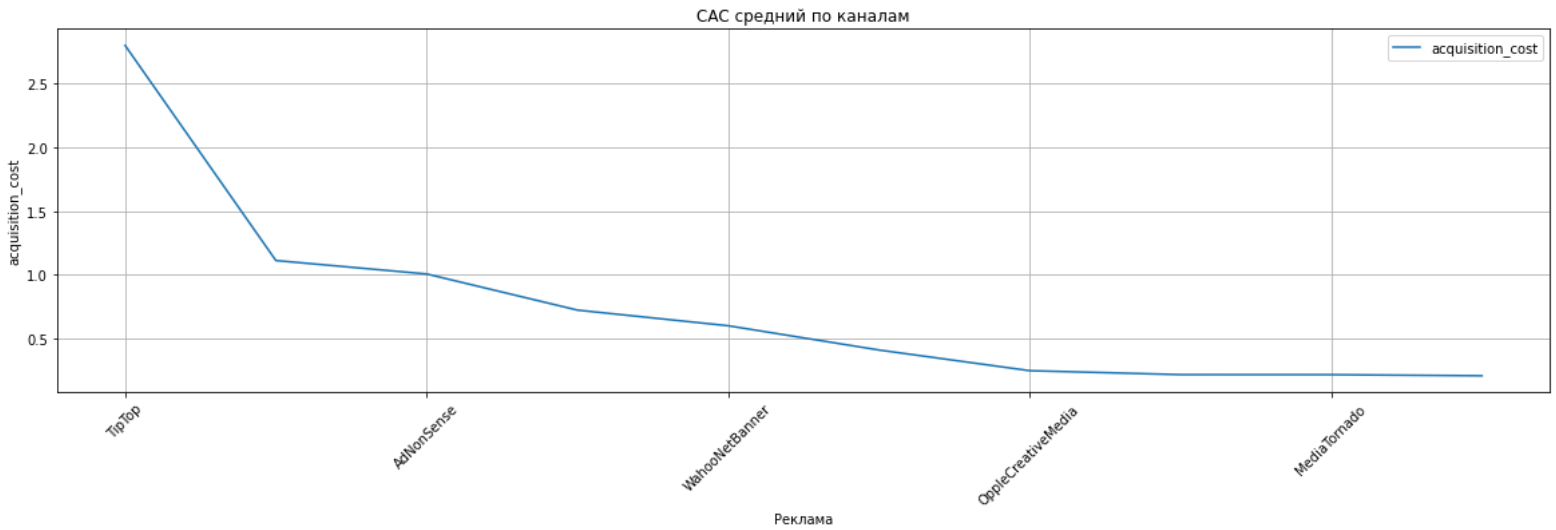
Out[77]:

Channel	acquisition_cost
TipTop	2.799003
FaceBoom	1.113286
AdNonSense	1.008054
lambdaMediaAds	0.724802
WahooNetBanner	0.602245
RocketSuperAds	0.412095
OppleCreativeMedia	0.250000
YRabbit	0.218975
MediaTornado	0.218717
LeapBob	0.210172

__Комментарий от ревьюера №4__ Неплохо бы посчитать средний CAC по проекту

In [78]:

```
cac_chanel.plot(grid=True, figsize=(20,5))

plt.title('CAC средний по каналам')
plt.xlabel('Реклама')
plt.xticks(rotation = 45)
plt.ylabel('acquisition_cost')

plt.show()
```



CAC как всегда самый высокий у рекламного источника TipTop

✍ __Комментарий от ревьюера №3__ Давай также выведем сводную таблицу, в которой расчитаем средний CAC для каждого канала?

а давай) и сделала

✍ __Комментарий от ревьюера №4__ Многие расчёты верны, но отсутствует общий вывод по каналу

ВЫВОД:

У всех каналов - рекламы- расходы одинаковые, кроме TipTop-тут расходы растут с середины мая 2019 года. Это можно увидеть на графике недельном и месячном.

Стоимость привлечения «органических» пользователей во всех когортах равна нулю, потому что они перешли на приложение самостоятельно, а не благодаря рекламе. Привлечение одного пользователя из рекламных сетей обошлось компании в среднем в 1 доллар, кроме TipTop — цена здесь достигла примерно $3. Очень высокие расходы на канал TipTop.

Шаг 5. Оцените окупаемость рекламы

Используя графики LTV, ROI и CAC, проанализируйте окупаемость рекламы. Считайте, что на календаре 1 ноября 2019 года, а в бизнес-плане заложено, что пользователи должны окупаться не позднее чем через две недели после привлечения. Необходимость включения в анализ органических пользователей определите самостоятельно. • Проанализируйте окупаемость рекламы с помощью графиков LTV и ROI, а также графики динамики LTV, CAC и ROI. • Проверьте конверсию пользователей и динамику её изменения. То же самое сделайте с удержанием пользователей. Постройте и изучите графики конверсии и удержания. • Проанализируйте окупаемость рекламы с разбивкой по устройствам. Постройте графики LTV и ROI, а также графики динамики LTV, CAC и ROI. • Проанализируйте окупаемость рекламы с разбивкой по странам. Постройте графики LTV и ROI, а также графики динамики LTV, CAC и ROI. • Проанализируйте окупаемость рекламы с разбивкой по рекламным каналам. Постройте графики LTV и ROI, а также графики динамики LTV, CAC и ROI. • Ответьте на такие вопросы: o Окупается ли реклама, направленная на привлечение пользователей в целом? o Какие устройства, страны и рекламные каналы могут оказывать негативное влияние на окупаемость рекламы? o Чем могут быть вызваны проблемы окупаемости?

Установим момент и горизонт анализа данных.На календаре 1 ноября 2019 года, и зададим 14 дневный горизонт анализа.

In [44]:

```
observation_date = datetime(2019, 11, 1).date()  # момент анализа
horizon_days = 14  # горизонт анализа
```

In [45]:

```
# Зададим горизонт анализа в 14 дней и посчитаем максимальную дату привлечения
analysis_horizon = 14
max_date = observation_date - timedelta(days=analysis_horizon - 1)
print(max_date)
```

2019-10-19
Считаем бизнес-показатели Для начала оценим общую ситуацию — посмотрим на окупаемость рекламы. Рассчитаем и визуализируем LTV и ROI, вызвав функции get_ltv() и plot_ltv_roi().

✍

Комментарий от ревьюера №3

Есть небольшая неточность, которая немного искажает общие данные: из расчетов нам следует исключить пользователей с органическим трафиком: profiles = profiles.query('channel != "organic"') - поскольку мы за них ничего не платим, а нам нужно изучить именно окупаемость рекламы.

In [46]:

```
# нам следует исключить пользователей с органическим трафиком

profiles = profiles.query('Channel != "organic"')
profiles['Channel'].unique()
```

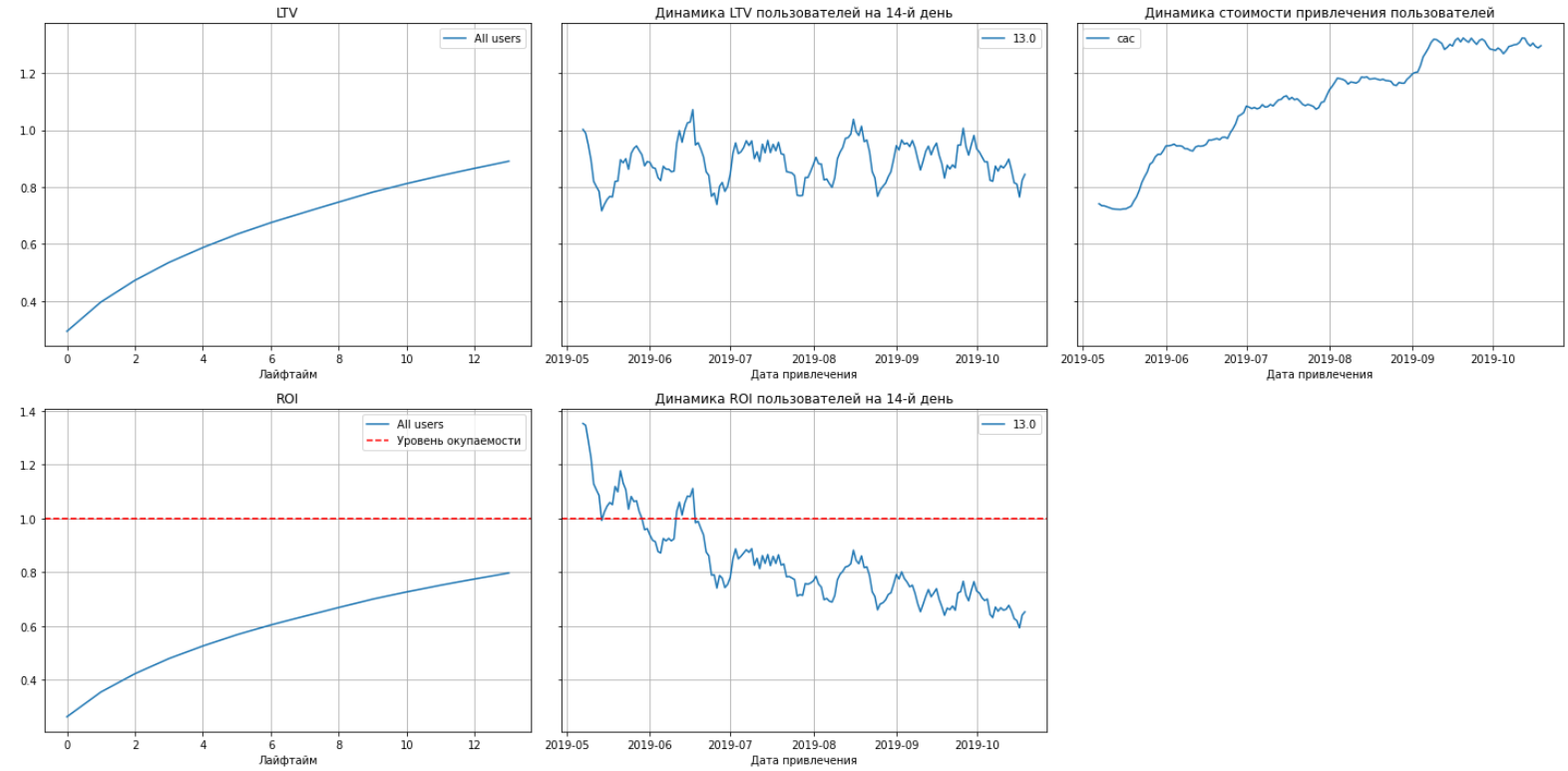
Out[46]:

```
array(['FaceBoom', 'AdNonSense', 'YRabbit', 'MediaTornado',
       'RocketSuperAds', 'LeapBob', 'TipTop', 'WahooNetBanner',
       'OppleCreativeMedia', 'lambdaMediaAds'], dtype=object)
```

✔ __Комментарий от ревьюера №4__ Совершенно верно, поскольку мы за них ничего не платим, а нам нужно изучить именно окупаемость рекламы.

In [47]:

```
# считаем LTV и ROI
ltv_raw, ltv_grouped, ltv_history, roi_grouped, roi_history = get_ltv(
    profiles, orders, observation_date, horizon_days
)

# строим графики
plot_ltv_roi(ltv_grouped, ltv_history, roi_grouped, roi_history, horizon_days)
```



In [48]:

```
profiles.head(1)
```

Out[48]:

	User Id	first_ts	Channel	Device	Region	dt	month	payer	acquisition_cost	costs	week
0	599326	2019-05-07	FaceBoom	Mac	United States	2019-05-07	2019-05-01	True	1.088172	101.2	2019-05-02

По графикам можно сделать такие выводы: Реклама не окупается. ROI в конце 2-ой недели 80% практически. Кривая динамики ROI всё время падает. CAC не стабилен. Значит, дело в увеличении рекламного бюджета. Кривая постояно растёт. LTV достаточно стабилен. Значит, дело не в ухудшении качества пользователей. Чтобы разобраться в причинах, пройдём по всем доступным характеристикам пользователей — стране, источнику и устройству первого посещения.

✍

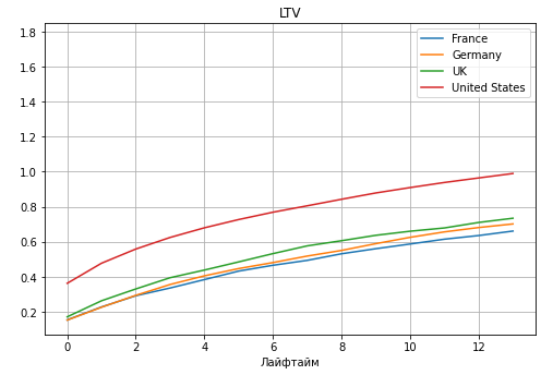
Комментарий от ревьюера №3

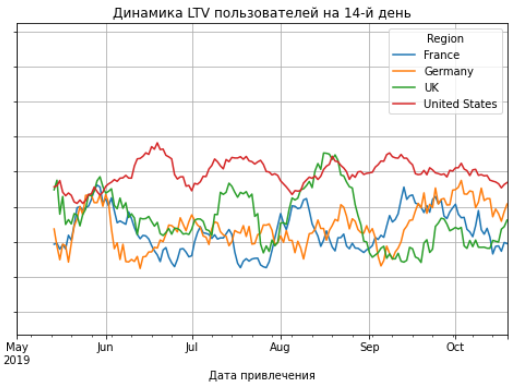
Поправь, пожалуйста, описание динамики изменения стоимости привлечения пользователей по каналам, устройствам и странам, когда будет проведена фильтрация органических пользователей.

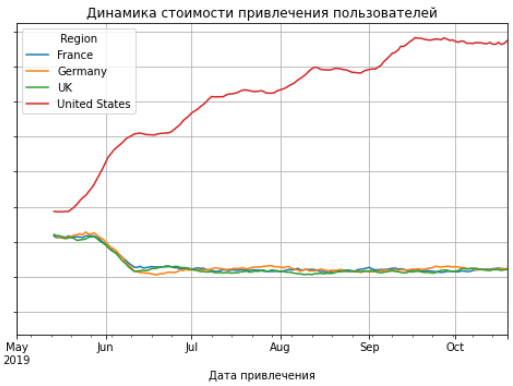
✔ __Комментарий от ревьюера №4__ Логика анализа верная, согласен с выводом. Наблюдаем, что динамика ROI за лайфтайм падает. При относительно стабильной динамике LTV, динамика CAC растёт с мая по конец октября. Эту закономерность мы наблюдаем в динамике ROI, что при сильном увеличении CAC, в равной степени падает динамика ROI пользователей.

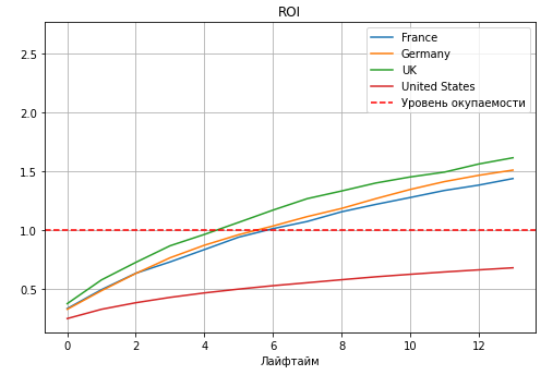
In [49]:

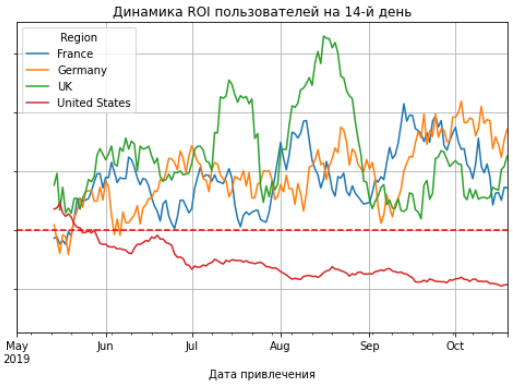
```
# смотрим окупаемость с разбивкой по странам

dimensions = ['Region']

ltv_raw, ltv_grouped, ltv_history, roi_grouped, roi_history = get_ltv(
    profiles, orders, observation_date, horizon_days, dimensions=dimensions
)

plot_ltv_roi(
    ltv_grouped, ltv_history, roi_grouped, roi_history, horizon_days, window=14
)
```











Начнём с разбивки по странам: передадим параметру dimensions столбец Region.

С разбивкой по странам всё тоже стабильно плохо.Вот что говорят графики:

Реклама окупается только в 3 странах: Фран.,Герм. и Великоб. И то с середины недели. США вообще не окупается.

Динамика ROI по странам говорит чтоу всех стран всё в порядке-кроме США - она падает. Растут расходы на привлечение пользователей из США.

США огромная стоимость привлечений. LTV всё так же стабилен. Лучше всего окупается Великобритания, а вот явных аутсайдеров нет.

Значит, дело в стране — эту версию работающая. И страна эта-США

✔

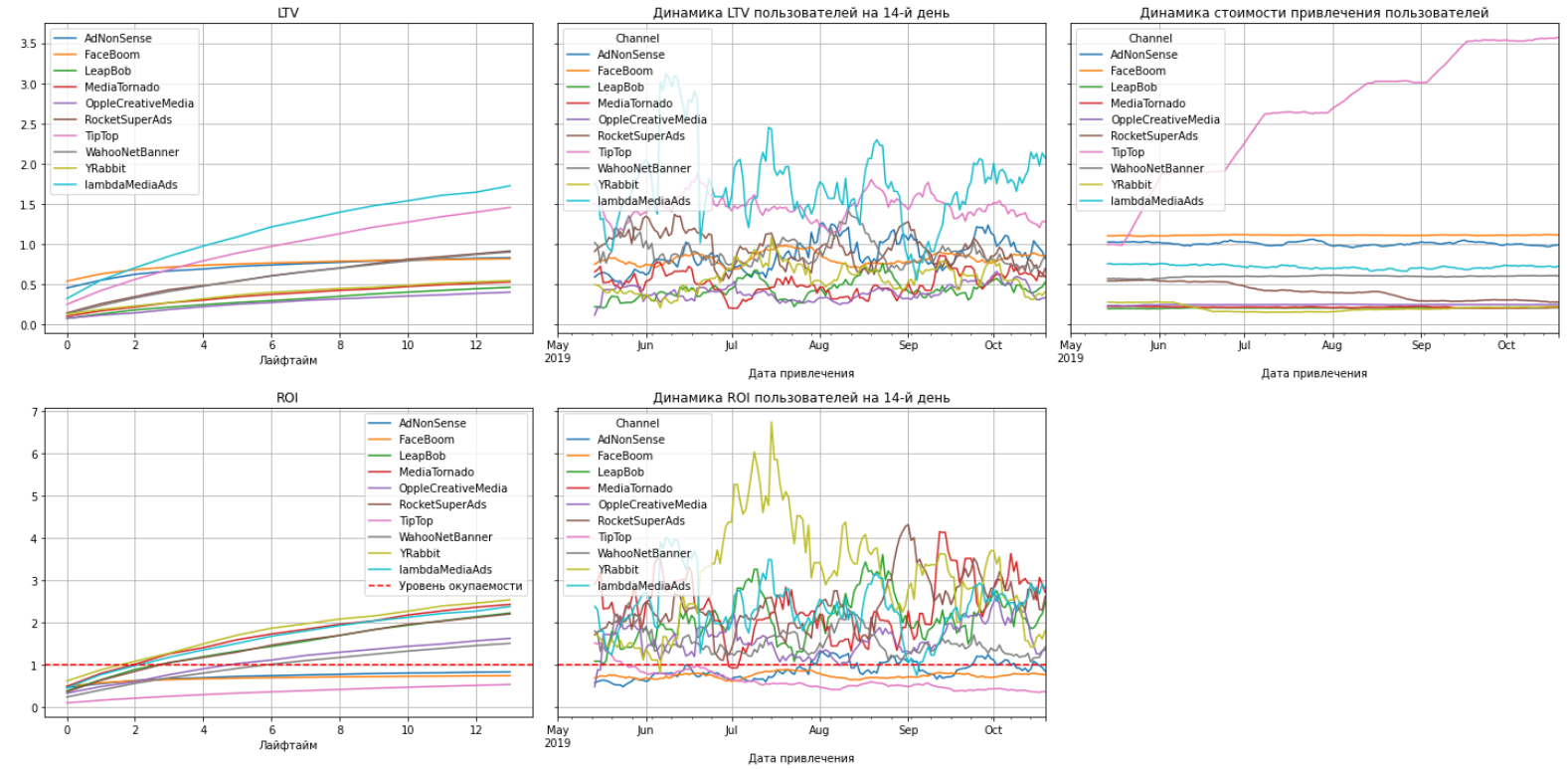
Комментарий от ревьюера №4

Все верно. Нужно разбираться детальнее с рекламой в США, тем более, что это наш основной рынок.

Теперь проверим источники привлечения (Channel). Возможно, на окупаемость влияет неудачная реклама в каком-нибудь одном канале.

In [50]:

```
#смотрим окупаемость с разбивкой по источникам привлечения

dimensions = ['Channel']

ltv_raw, ltv_grouped, ltv_history, roi_grouped, roi_history = get_ltv(
    profiles, orders, observation_date, horizon_days, dimensions=dimensions
)

plot_ltv_roi(
    ltv_grouped, ltv_history, roi_grouped, roi_history, horizon_days, window=14
)
```



✔ __Комментарий от ревьюера №4__ Да, действительно, есть проблемы с каналом TipTop, видим значительный рост затрат на привлечение.

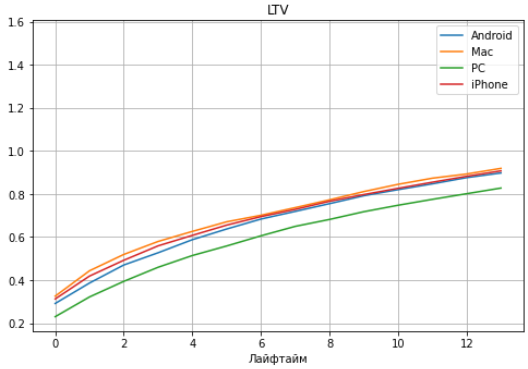
Перейдём к рекламе.Окупаемость половины рекламных средств и то только с середины недели. Особенно канал привлечения TipTop- он вообще убыточен на протяжении всех 2 недель. Однако на этот канал-самые большие растущие расходы. Дело в канале привлечения TipTop. Совет маркетолгам немедленно его убрать-именно он тянет компанию вниз.

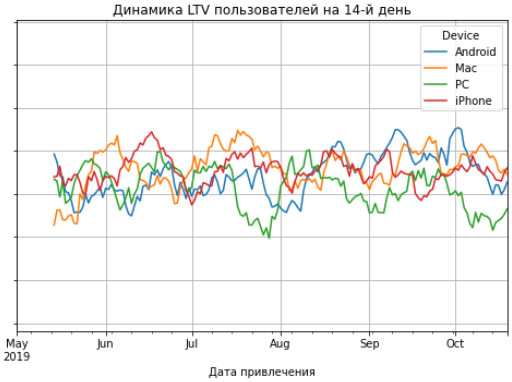
давай предложим альтернативные каналы, по которому показатели конверсии, удержания и ROI на приемлемом уровне. Как думаешь, а канал Yrabbit и RocketSuperAds подойдут?

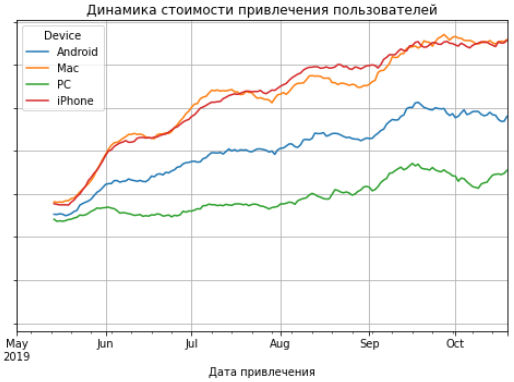
- Я думаю эти каналы подойдут, т.к. у них хорошая конвертируемость и удержание клиентов.

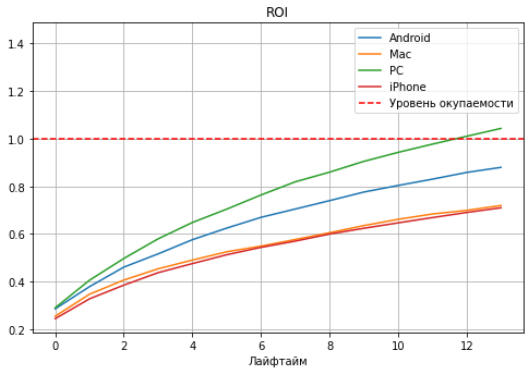
In [51]:

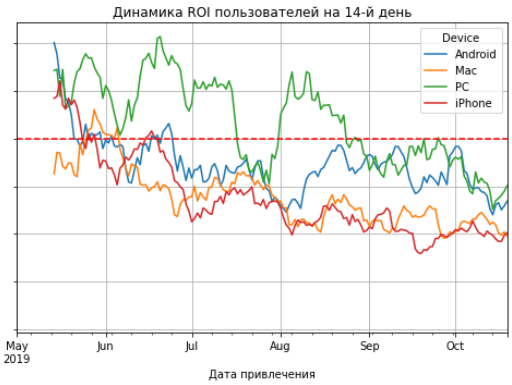
```
# смотрим окупаемость с разбивкой по устройствам

dimensions = ['Device']

ltv_raw, ltv_grouped, ltv_history, roi_grouped, roi_history = get_ltv(
    profiles, orders, observation_date, horizon_days, dimensions=dimensions
)

plot_ltv_roi(
    ltv_grouped, ltv_history, roi_grouped, roi_history, horizon_days, window=14
)
```











CAC- расходы на привлечение - растут синхронно. Но вот окупается только PC- и на него самые маленькие расходы. Интересно по динамике ROI видно что он не окупается последние 2 месяца-прям как в описанных проблемах. Фирма последние 2 месяца убыточна. Это зацепка. Узнаем в чём причина: в низкой конверсии или низком удержании.

Посчитаем и визуализируем конверсию, вызвав функции get_conversion()

✔ __Комментарий от ревьюера №4__ С окупаемостью проблемы по всем устройствам кроме PC. Это значит, что у нас, по крайней мере, нет технических проблем, влияющих на монетизацию;

Рассчитаем конверсию с разбивкой по странам, передав get_conversion() фреймы profiles и orders, а также столбец region в качестве параметра dimensions, и построим тепловую карту по таблице конверсии. Момент и горизонт анализа данных остаются прежними — 1 ноября 2019 года и 14 дней соответственно.

In [52]:

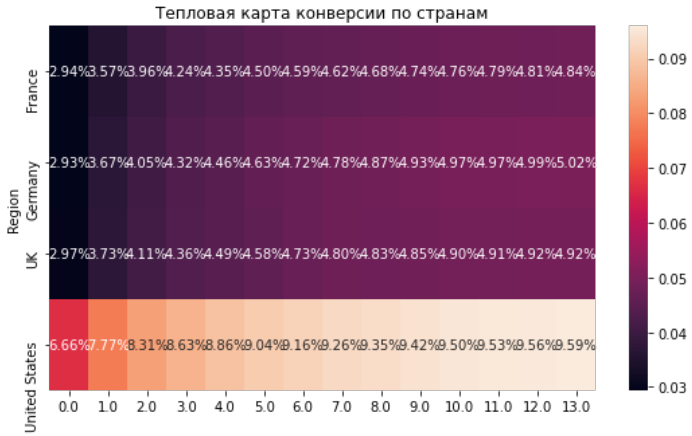
```
# получаем сырые данные, таблицу конверсии и таблицу динамики конверсии
conversion_raw, conversion, conversion_history = get_conversion(
    profiles, orders, datetime(2019, 11, 1).date(), 14, dimensions=['Region']
)
```

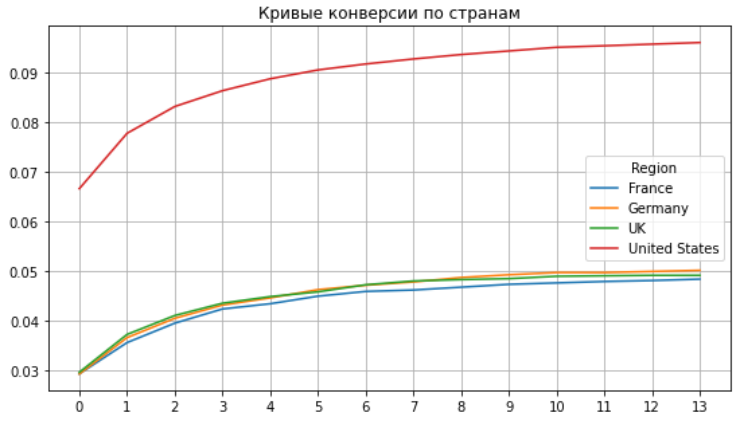
In [53]:

```
plt.figure(figsize = (20, 5)) # задаём размер «подложки»

# исключаем размеры когорт
# конверсии первого дня различаются, их удалять не нужно
report = conversion.drop(columns = ['cohort_size'])

sns.heatmap(
    report, annot=True, fmt='.2%', ax=plt.subplot(1, 2, 1)
) # в первой ячейке таблицы графиков строим тепловую карту
plt.title('Тепловая карта конверсии по странам')

report.T.plot(
    grid=True, xticks=list(report.columns.values), ax=plt.subplot(1, 2, 2)
) # во второй — кривые конверсии
plt.title('Кривые конверсии по странам')
```

Out[53]:

Text(0.5, 1.0, 'Кривые конверсии по странам')





По графику можно сделать такие выводы: Выше всего конверсия у посетителей из США: 6.6% новых пользователей совершают покупки в день первого посещения, а к 14 дню «жизни» доля покупателей почти достигает 10%. Хуже всего конвертируются пользователи из Франции: конверсия первого дня ниже у Франции — около 2.9%. В целом конверсия пользователей из всех стран постепенно растёт на протяжении всех 14 дней.

In [54]:

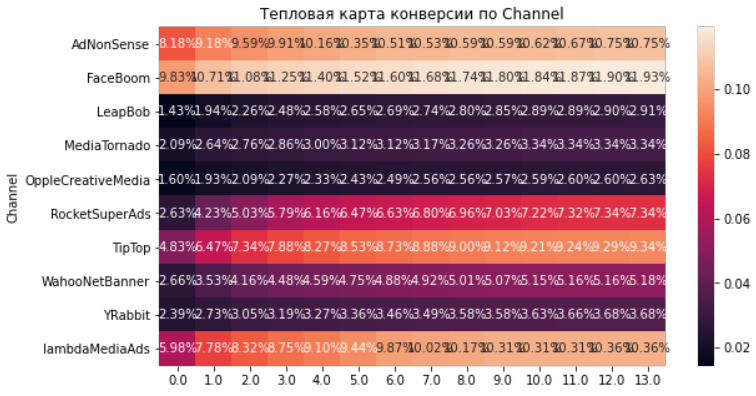
```
# получаем сырые данные, таблицу конверсии и таблицу динамики конверсии Channel
conversion_raw, conversion, conversion_history = get_conversion(
    profiles, orders, datetime(2019, 11, 1).date(), 14, dimensions=['Channel']
)
```

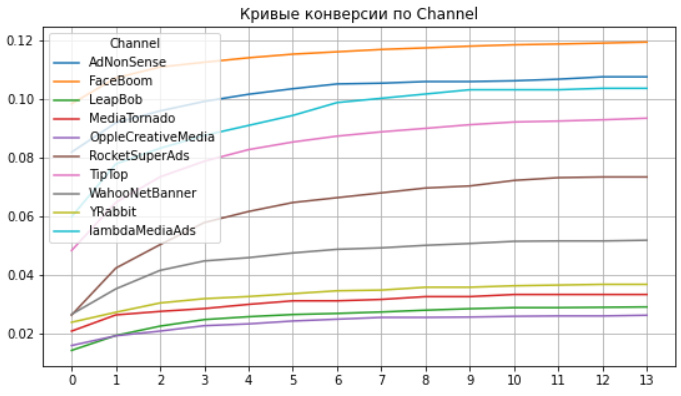
In [55]:

```
plt.figure(figsize = (20, 5)) # задаём размер «подложки»

# исключаем размеры когорт
# конверсии первого дня различаются, их удалять не нужно
report = conversion.drop(columns = ['cohort_size'])

sns.heatmap(
    report, annot=True, fmt='.2%', ax=plt.subplot(1, 2, 1)
) # в первой ячейке таблицы графиков строим тепловую карту
plt.title('Тепловая карта конверсии по Channel')

report.T.plot(
    grid=True, xticks=list(report.columns.values), ax=plt.subplot(1, 2, 2)
) # во второй — кривые конверсии
plt.title('Кривые конверсии по Channel')
```

Out[55]:

Text(0.5, 1.0, 'Кривые конверсии по Channel')



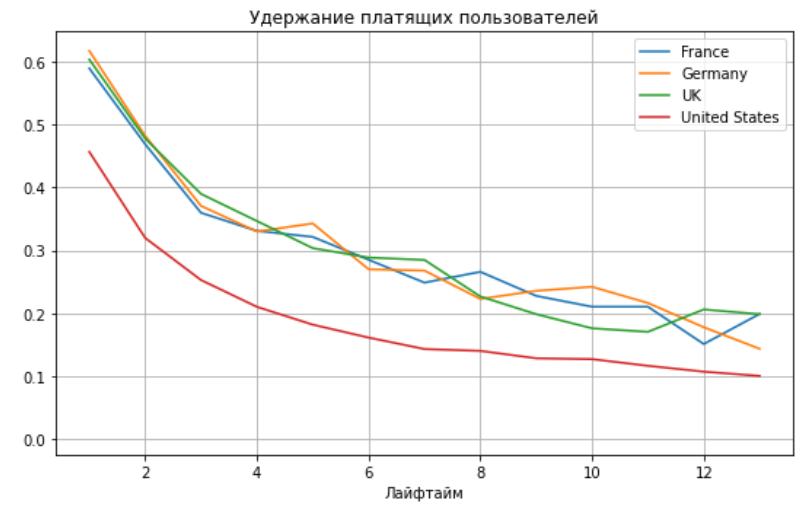


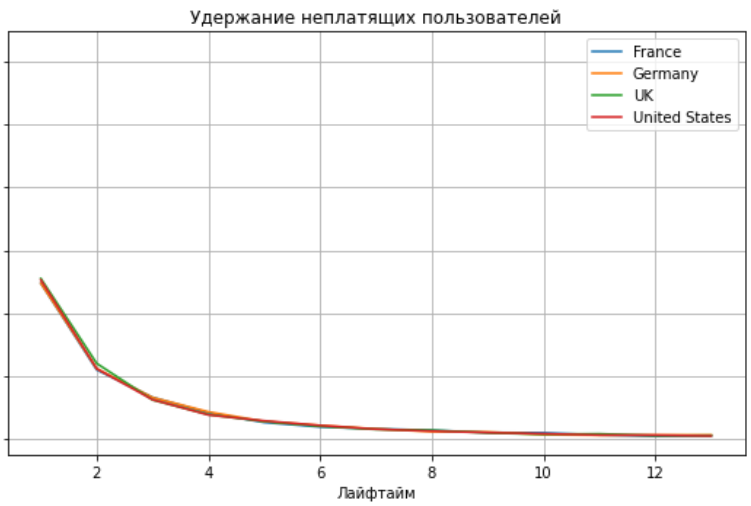
✍ __Комментарий от ревьюера №3__ А что мы тут видим? Возьми, пожалуйста, расчёт конверсии и удержания из тренажера. Например расчёт по странам - отличный (только нету конверсии)

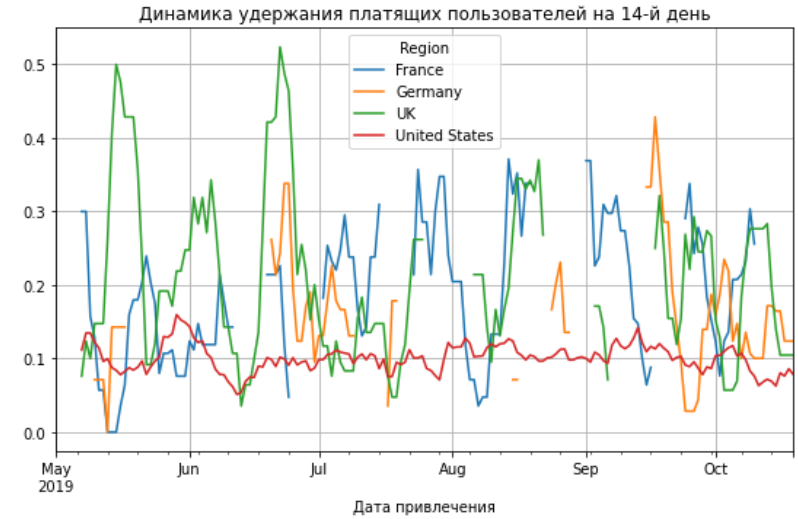
Пользователи всех устройств стабильно плохо удерживаются. Дело не в удержании и конверсии, а в рекламе.

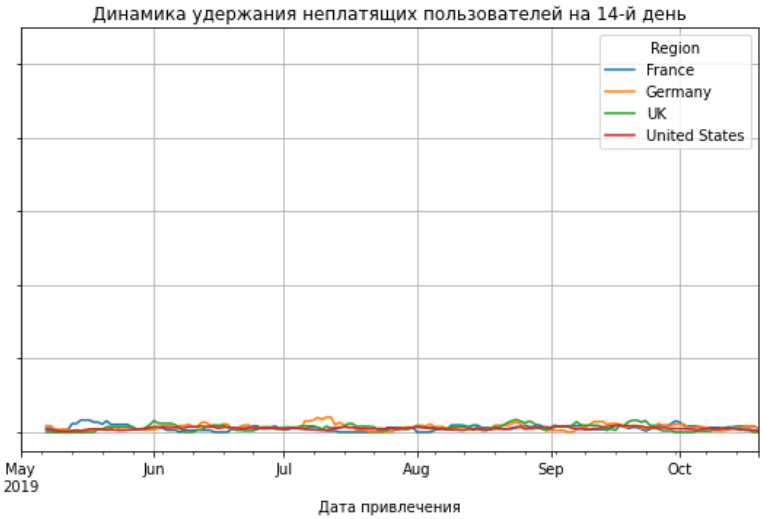
In [56]:

```
# смотрим удержание с разбивкой по странам

retention_raw, retention_grouped, retention_history = get_retention(
    profiles, visits, observation_date, horizon_days, dimensions=['Region']
)

plot_retention(retention_grouped, retention_history, horizon_days)
```





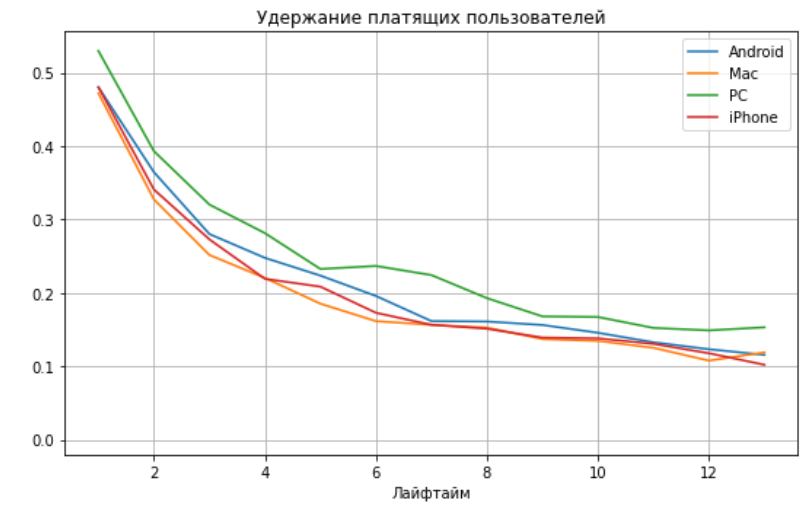


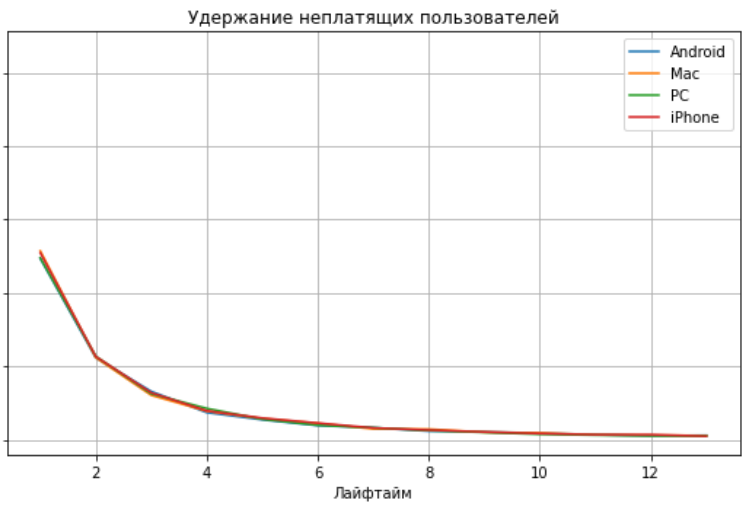


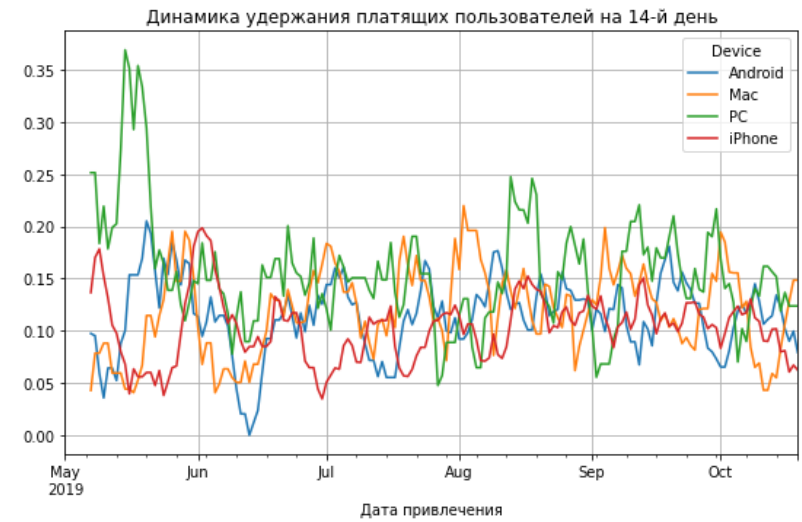
Действительно, пользователи США стабильно плохо удерживаются. Для платящих пользователей в США - удержание самое низкое из всех стран.

In [57]:

```
# смотрим удержание с разбивкой по устройствам

retention_raw, retention_grouped, retention_history = get_retention(
    profiles, visits, observation_date, horizon_days, dimensions=['Device']
)

plot_retention(retention_grouped, retention_history, horizon_days)
```



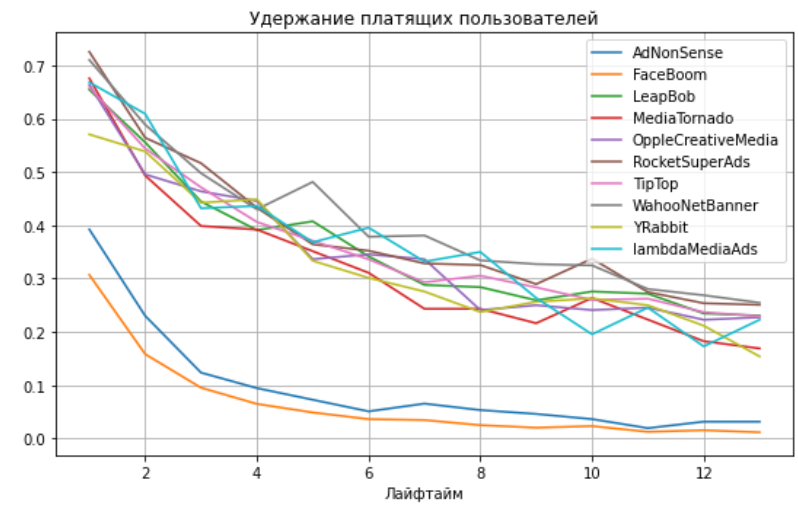


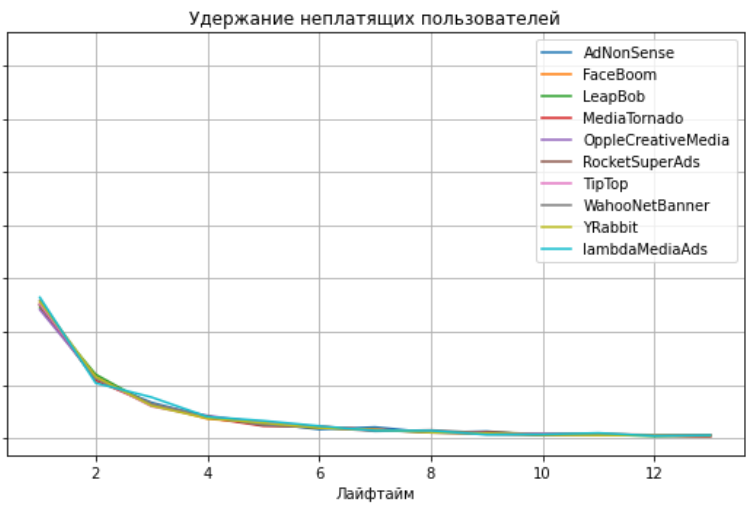


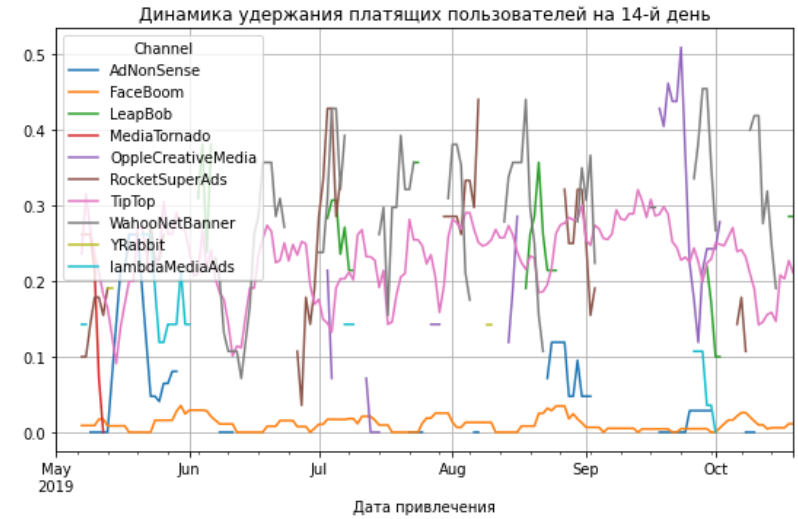
Действительно, пользователи iPhone стабильно плохо удерживаются. Для платящих пользователей на iPhone удержание 7-го дня ниже, чем на остальных устройствах, примерно на 40%. Это очень низкий показатель.

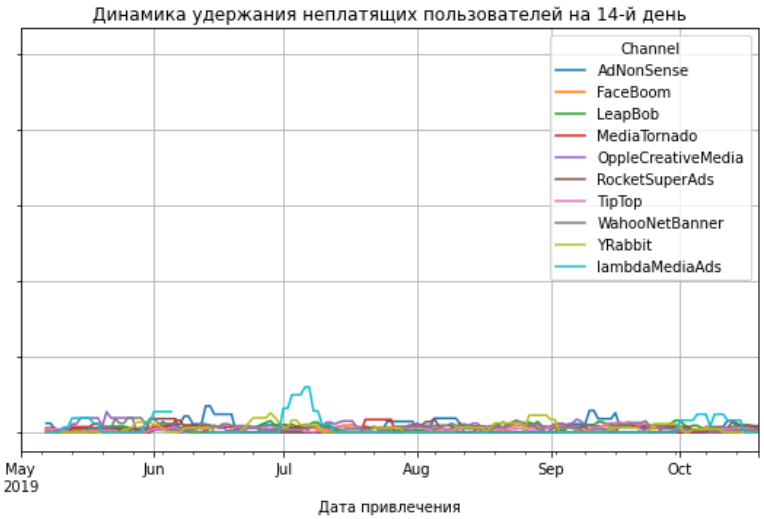
In [58]:

```
# смотрим удержание с разбивкой по рекламе

retention_raw, retention_grouped, retention_history = get_retention(
    profiles, visits, observation_date, horizon_days, dimensions=['Channel']
)

plot_retention(retention_grouped, retention_history, horizon_days)
```









Самое низкое удержание клиентов от рекламы AdNon и FaceBoom

✍ __Комментарий от ревьюера №4__ Точно lambda?

Да-исправила. перепутала цвет)

✔

Комментарий от ревьюера №5

Хорошо, анализ окупаемости корректен, согласен с интерпретацией результатов. Проблемные источники определены верно.

✍ __Комментарий от ревьюера №3__ Анализ не полный, т.к. проблемы могут быть в удержании каналов/стран/устройств/, а без анализа конверсии и удержания, мы не сможем дать рекомендации маркетологам.

Шаг 6. Напишите выводы

Вывод: Больше всего платящих пользователей было из FaceBoom и lambdaMediaAds.

Получились устройства -Mac и iPhone- больше всего платящих. Больше всего платящих пользователей в США, потом Германия.

Вывод: Всего на рекламу было потрачено- 105497.3-на каждого привлеченного пользователя. Из них больше всего на рекламу от рекламного источника TipTop. На органические пользователи(кот.сами пришли)-мы ничего не тратили.

У всех каналов - рекламы- расходы одинаковые, кроме TipTop-тут расходы растут с середины мая 2019 года. Это можно увидеть на графике недельном и месячном.

Стоимость привлечения «органических» пользователей во всех когортах равна нулю, потому что они перешли на сайт приложения самостоятельно, а не благодаря рекламе. Реклама всех каналов адекватна кроме-Tip Top-огромные расходы на рекламу. CAC как всегда самый высокий у рекламного источника TipTop.

По графикам можно сделать такие выводы: Реклама не окупается. ROI в конце 2-ой недели 80% практически. Кривая динамики ROI всё время падает. CAC не стабилен. Значит, дело в увеличении рекламного бюджета. Кривая постояно растёт. LTV достаточно стабилен. Значит, дело не в ухудшении качества пользователей. Чтобы разобраться в причинах, пройдём по всем доступным характеристикам пользователей — стране, источнику и устройству первого посещения.

Начнём с разбивки по странам: передадим параметру dimensions столбец Region.

С разбивкой по странам всё тоже стабильно плохо.Вот что говорят графики:

Реклама окупается только в 3 странах: Фран.,Герм. и Великоб. И то с середины недели. США вообще не окупается.

Динамика ROI по странам говорит чтоу всех стран всё в порядке-кроме США - она падает. Растут расходы на привлечение пользователей из США.

США огромная стоимость привлечений. LTV всё так же стабилен. Лучше всего окупается Великобритания, а вот явных аутсайдеров нет.

Значит, дело в стране — эту версию работающая. И страна эта-США

Перейдём к рекламе.Окупаемость половины рекламных средств и то только с середины недели. Особенно канал привлечения TipTop- он вообще убыточен на протяжении всех 2 недель. Однако на этот канал-самые большие растущие расходы. Дело в канале привлечения TipTop. Совет маркетолгам немедленно его убрать-именно он тянет компанию вниз.

давай предложим альтернативные каналы, по которому показатели конверсии, удержания и ROI на приемлемом уровне. Как думаешь, а канал Yrabbit и RocketSuperAds подойдут?

- Я думаю эти каналы подойдут, т.к. у них хорошая конвертируемость и удержание клиентов.

CAC- расходы на привлечение - растут синхронно. Но вот окупается только PC- и на него самые маленькие расходы. Интересно по динамике ROI видно что он не окупается последние 2 месяца-прям как в описанных проблемах. Фирма последние 2 месяца убыточна. Это зацепка. Узнаем в чём причина: в низкой конверсии или низком удержании.

Посчитаем и визуализируем конверсию, вызвав функции get_conversion() По графику можно сделать такие выводы: Выше всего конверсия у посетителей из США: 6.6% новых пользователей совершают покупки в день первого посещения, а к 14 дню «жизни» доля покупателей почти достигает 10%. Хуже всего конвертируются пользователи из Франции: конверсия первого дня ниже у Франции — около 2.9%. В целом конверсия пользователей из всех стран постепенно растёт на протяжении всех 14 дней. Действительно, пользователи США стабильно плохо удерживаются. Для платящих пользователей в США - удержание самое низкое из всех стран.

Действительно, пользователи iPhone стабильно плохо удерживаются. Для платящих пользователей на iPhone удержание 7-го дня ниже, чем на остальных устройствах, примерно на 40%. Это очень низкий показатель. Самое низкое удержание клиентов от рекламы Lambda и AdNon

Причина убытков фирмы-канал привлечения TipTop. Именно он тянет вниз. Совет маркетолгам немедленно его убрать-именно он тянет компанию вниз.Как альтернатива новых источников - канал Yrabbit и RocketSuperAds

Комментарий от ревьюера №3

Шикарный вывод, молодец, что в итоговом выводе сформулировала рекомендации для отдела маркетинга: про каналы, от которых стоит отказаться ты упомянула, давай предложим альтернативные каналы, по которому показатели конверсии, удержания и ROI на приемлемом уровне. Как думаешь, а канал Yrabbit и RocketSuperAds подойдут?

Комментарий от ревьюера №4

Я бы не рекомендовал вкладываться в канал Yrabbit . Почему я так думаю: да, канал сейчас окупается, потому что там низкий CAC. Фактически мы сейчас сильно много не платим за новых клиентов (CAC = 0,21) . И уже на этих клиентах (которые обошлись нам недорого) мы сможем спрогнозировать будущую картину. Они имеют низкую конвертируемость и сильно падающее удержание на последних днях горизонта. Т.е. можно сделать вывод, что им не интересно наше приложение. Возможно, это просто не наша аудитория. Они не заинтересованы в нашем приложении

Комментарий от ревьюера №3 У тебя получилась очень сильная и хорошая работа. Здорово, что расчеты ты сопровождаешь иллюстрациями, а так же не забываешь про комментарии, твой проект интересно проверять. --- Нужно поправить: 1) Отсутствует проверка на пропуски и дубликаты 2) Шаг 3, даты 3) Горизонт анализа и момент анализа 4) Оформление графиков 5) Шаг 4.1 - 4.2 (отсутствует) 6) Визуализировать динамику расходов по каналам, по месяцам и неделям (2 графика) 7) Средний CAC для каналов 8) После каждого раздела / графика (или серии тестов) писать вывод по полученным данным с учетом поставленной бизнес задачи 9) Из расчетов нам следует исключить пользователей с органическим трафиком 10) Конверсия и удержание по метрикам (страны / каналы / девайсы) 11) Подправить выводы, после изменений ---- Если у тебя будут какие-то вопросы по моим комментариям - обязательно пиши! Буду ждать работу на повторное ревью :)

Комментарий от ревьюера №4 Отличная работа, осталось поправить несколько моментов: --- Нужно поправить: 1) Шаг 3, даты 2) Оформление графиков 3) Раздел 4 (расчёт расходов) 4) После каждого раздела / графика (или серии тестов) писать вывод по полученным данным с учетом поставленной бизнес задачи 5) Подправить выводы, после изменений ---- Если у тебя будут какие-то вопросы по моим комментариям - обязательно пиши! Буду ждать работу на повторное ревью :)

Вроде всё исправила-я написала вопрос про затраты. они же входят в таблицу профайлс.

Комментарий от ревьюера №5 Отличная работа, осталось поправить несколько моментов: --- Нужно поправить: 1) Раздел (расчёт расходов) 2) Подправить выводы, после изменений ---- Если у тебя будут какие-то вопросы по моим комментариям - обязательно пиши! Буду ждать работу на повторное ревью :)

Я исправила выводы и сделала анализ по другому столбцу. И написала по дат

✔

Коментарий от ревьюера №6

В остальном всё чудно😊. Твой проект так и просится на github =)

Поздравляю с успешным завершением проекта 😊 И желаю успехов в новых работах 😊

От себя хочу порекомендовать тебе отличный метериал про продуктовую аналитику Дмитрия Животворева.

https://www.youtube.com/watch?v=Vy_rq-x9QEo

In [ ]: