

Анализ причин убытков компании-разработчика развлекательного приложения

Содержание

- 1 Импорт библиотек и определение функций проекта
 - 1.1 Импорт библиотек
 - 1.2 Функции загрузки и обзора данных
 - 1.3 Функции определения и работы с профилями пользователей
 - 1.4 Функции расчёта бизнес-метрик
 - 1.4.1 Функция расчёта конверсии (Conversion Rate)
 - 1.4.2 Функция расчёта удержания (Retention Rate)
 - 1.4.3 Функция расчёта пожизненной ценности (Lifetime Value), стоимости привлечения (Customer Acquisition Cost) и возврата на инвестиции (Return On Investment)
 - 1.5 Функции визуализации бизнес-метрик
 - 1.5.1 Функция визуализации удержания (Retention Rate) и его динамики
 - 1.5.2 Функция визуализации конверсии (Conversion Rate) и её динамики
 - 1.5.3 Функция визуализации пожизненной ценности (Lifetime Value), стоимости привлечения (Customer Acquisition Cost) и возврата на инвестиции (Return On Investment) и их динамики
 - 2 Обзор данных
 - 2.1 Обзор данных о посещениях пользователей
 - 2.2 Обзор данных о покупках пользователей
 - 2.3 Обзор данных о рекламных расходах
 - 2.4 Выводы
 - 3 Предварительная обработка данных
 - 3.1 Переименование столбцов
 - 3.2 Приведение типов данных
 - 3.3 Проверка категориальных столбцов на неявные дубликаты
 - 3.4 Выводы
 - 4 Исследовательский анализ данных
 - 4.1 Создание профилей пользователей
 - 4.2 Изучение профилей в разрезе регионов привлечения
 - 4.3 Изучение профилей в разрезе используемых устройств
 - 4.4 Изучение профилей в разрезе каналов привлечения
 - 4.5 Изучение регионов деятельности каналов привлечения
 - 4.6 Определение наиболее часто используемых устройств по регионам
 - 4.7 Выводы
 - 5 Исследование маркетинговых затрат
 - 5.1 Общая сумма и распределение по источникам затрат на маркетинг
 - 5.2 Расчёт средней стоимости привлечения (CAC)
 - 5.3 Выводы
 - 6 Оценка окупаемости рекламы
 - 6.1 Анализ общей окупаемости рекламы
 - 6.2 Анализ окупаемости рекламы по каналам привлечения
 - 6.3 Анализ окупаемости рекламы по странам привлечения
 - 6.4 Анализ окупаемости рекламы по устройствам
 - 6.5 Изучение конверсии и удержания пользователей
 - 6.5.1 Конверсия и удержание пользователей в разрезе рекламных каналов в США
 - 6.5.2 Конверсия и удержание пользователей в разрезе рекламных каналов в Европе
 - 6.5.3 Конверсия и удержание пользователей в разрезе регионов
 - 6.5.4 Конверсия и удержание пользователей в разрезе устройств
 - 6.6 Выводы
 - 7 Общий вывод исследования
-

Несмотря на огромные вложения в рекламу, компания-разработчик развлекательного приложения Procrastinate Pro+ последние несколько месяцев терпит убытки. Требуется разобраться в причинах такого положения дел и помочь компании выйти в плюс.

В наличии имеются следующие данные о пользователях, привлечённых с 1 мая по 27 октября 2019 года:

- лог сервера с данными об их посещениях (`visits_info_short.csv`),
- выгрузка их покупок за этот период (`orders_info_short.csv`),
- рекламные расходы (`costs_info_short.csv`).

В ходе исследования предполагается 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.

Момент анализа: 1 ноября 2019 года.

Целевой срок окупаемости пользователей: не позднее чем через две недели после привлечения.

Импорт библиотек и определение функций проекта

Импорт библиотек

```
In [1]: import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime, timedelta
```

Функции загрузки и обзора данных

```
In [2]: # определение функции обзора данных
# =====
# на вход подаётся датафрейм df
# на выходе:
#   - 10 случайных строк df
#   - информация df.info()
#   - количество явных дубликатов в строках df
#   - процент пропусков данных в столбцах df
# =====
def data_observe(df):
    row_num = 5      # количество отображаемых строк таблицы

    print('Произвольные строки таблицы:')
    print('=====')
    if len(df) >= row_num:
        display(df.sample(row_num))
    else:
        display(df)

    print('\nИнформация о таблице:')
    print('=====')
    df.info()

    print('\nКоличество явных дубликатов в таблице:')
    print('=====')
    print(df.duplicated().sum())

    print('\nПроцент пропусков в столбцах:')
    print('=====')
    display(pd.DataFrame(
        round((df.isna().mean()*100),2), columns=['NaNs, %'])
        .sort_values(by='NaNs, %', ascending=False)
        )
    .style.format('{:.2f}')
    .background_gradient('coolwarm')
    )
```

```
In [3]: # определение кросплатформенной функции загрузки данных
# =====
# на вход подаётся:
#   file_name - имя файла
# на выходе - датафрейм с загруженными данными
# =====
def open_file(file_name, sep=','):
    try:
        df = pd.read_csv('/datasets/' + file_name, sep=sep)
    except:
        df = pd.read_csv(file_name, sep=sep)

    return df
```

Функции определения и работы с профилями пользователей

```
In [4]: # определение профилей пользователей
# =====
def get_profiles(
    sessions,      # данные о сессиях пользователей
    orders,        # данные о покупках
    ad_costs,      # стоимость привлечения
    events=None,   # данные о событиях для маркировки пользователей
    event_names=[] # названия событий для маркировки пользователей
):

    # находим параметры первых посещений
    profiles = (
        sessions.sort_values(by=['user_id', 'session_start'])
        .groupby('user_id')
        .agg(
            {
                'session_start': 'first',
                'channel': 'first',
                'device': 'first',
                'region': 'first',
            }
        )
        .rename(columns={'session_start': 'first_ts'})
        .reset_index()
    )

    # для когортного анализа определяем дату первого посещения
    # и первый день месяца, в который это посещение произошло
    profiles['dt'] = profiles['first_ts'].dt.date
    profiles['month'] = profiles['first_ts'].astype('datetime64[M]')

    # добавляем признак платящих пользователей
    profiles['payer'] = profiles['user_id'].isin(orders['user_id'].unique())

    # добавляем флаги для всех событий из event_names:
    for event in event_names:
        if event in events['event_name'].unique():
            profiles[event] = profiles['user_id'].isin(
```

```

        events.query('event_name == @event')['user_id'].unique()

    )

    # считаем количество уникальных пользователей
    # с одинаковыми источником и датой привлечения
    new_users = (
        profiles.groupby(['dt', 'channel'])
        .agg({'user_id': 'nunique'})
        .rename(columns={'user_id': 'unique_users'})
        .reset_index()
    )

    # объединяя траты на рекламу и число привлечённых пользователей
    ad_costs = ad_costs.merge(new_users, on=['dt', 'channel'], how='left')

    # делим рекламные расходы на число привлечённых пользователей
    ad_costs['acquisition_cost'] = ad_costs['costs'] / ad_costs['unique_users']

    # добавляем стоимость привлечения в профили
    profiles = profiles.merge(
        ad_costs[[['dt', 'channel', 'acquisition_cost']],
        on=['dt', 'channel'],
        how='left',
    )

    # стоимость привлечения органических пользователей равна нулю
    profiles['acquisition_cost'] = profiles['acquisition_cost'].fillna(0)

    return profiles

```

```

In [5]: # изучение профилей плательщиков в разрезе
# =====
def payers_study(prf, by):
    payers = (
        prf
        .groupby(by=by) # сгруппируем данные профиля
        .agg({'user_id': 'nunique', # посчитаем количество уникальных пользователей в группировке
              'payer': 'sum'}) # а также - количество плательщиков
        .rename(columns={'user_id': 'user_count',
                        'payer': 'payer_count'})
    )

    # посчитаем процент плательщиков
    payers['payer_percent'] = (
        payers['payer_count']
        .div(payers['user_count'], axis=0) * 100
    ).round(2)

    return payers.sort_values(by='payer_percent', ascending=False)

```

Функции расчёта бизнес-метрик

Функция расчёта конверсии (Conversion Rate)

```

In [6]: # определение функции подсчёта конверсии Conversion Rate
# конверсия считается с накоплением по лайфтаймам
# =====
def get_conversion(
    profiles, # профили пользователей
    purchases, # данные о покупках
    observation_date, # момент анализа
    horizon_days, # горизонт анализа
    dimensions=[], # признаки когорт
    ignore_horizon=False # игнорировать горизонт
):

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

    # добавляем данные о первых покупках в профили
    result_raw = result_raw.merge(
        ( # определяем дату и время первой покупки для каждого пользователя
            purchases.sort_values(by=['user_id', 'event_dt'])
            .groupby('user_id')
            .agg({'event_dt': 'first'})
            .reset_index()
        ),
        on='user_id',
        how='left'
    )

    # рассчитываем лайфтайм для каждой покупки
    result_raw['lifetime'] = (
        result_raw['event_dt'] - # из момента события конверсии
        result_raw['first_ts'] # вычитаем момент привлечения
    ).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):
        # строим «треугольную» таблицу конверсии
        result = df.pivot_table(
            index=dims,
            columns='lifetime',
            values='user_id',
            aggfunc='nunique'
        ).fillna(0).cumsum(axis=1) # считаем сумму с накоплением для каждой строки

        # вычисляем размеры когорт

```

```

cohort_sizes = (
    df.groupby(dims)
    .agg({'user_id': 'nunique'})
    .rename(columns={'user_id': 'cohort_size'})
)

# добавляем размеры когорт в таблицу конверсии
result = cohort_sizes.merge(result, on=dims, how='left').fillna(0)

# делим каждую «ячейку» в строке на размер когорты
# и получаем CR
result = result.div(result['cohort_size'], axis=0)

# исключаем все лайфтаймы, 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)

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

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

# возвращаем обе таблицы и сырье данные
return (
    result_raw,      # сырье данные
    result_grouped, # таблица конверсии
    result_in_time  # динамика конверсии
)

```

Функция расчёта удержания (Retention Rate)

```

In [7]: # определение функции подсчёта удержания Retention Rate
# конверсия считается с накоплением по лайфтаймам
# =====
def get_retention(
    profiles,          # профили пользователей
    sessions,          # данные о сессиях
    observation_date,  # момент анализа
    horizon_days,      # горизонт анализа
    dimensions=[],     # признаки когорт
    ignore_horizon=False # игнорировать горизонт
):
    # добавляем столбец payer в передаваемый список признаков dimensions
    dimensions = ['payer'] + dimensions

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

    # собираем «сырые» данные для расчёта удержания
    result_raw = result_raw.merge(
        sessions[['user_id', 'session_start']], on='user_id', how='left'
    )

    # вычисляем лайфтайм для каждой сессии активности в днях
    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):
        # строим «треугольную» таблицу удержания (количество активных в лайфтайм)
        result = df.pivot_table(
            index=dims, columns='lifetime', values='user_id', aggfunc='nunique'
        )

        # вычисляем размеры когорт
        cohort_sizes = (
            df.groupby(dims)
            .agg({'user_id': 'nunique'})
            .rename(columns={'user_id': 'cohort_size'})
        )

        # добавляем размеры когорт в таблицу конверсии
        result = cohort_sizes.merge(result, on=dims, how='left').fillna(0)

        # делим каждую «ячейку» в строке на размер когорты
        # и получаем RetentionRate
        result = result.div(result['cohort_size'], axis=0)

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

        # восстанавливаем размеры когорт
        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 + ['dt'], horizon_days
    )

```

```
# возвращаем обе таблицы и сырье данные
return (
    result_raw,      # "сырые данные"
    result_grouped, # таблица удержания
    result_in_time  # динамика удержания
)
```

Функция расчёта пожизненной ценности (Lifetime Value), стоимости привлечения (Customer Acquisition Cost) и возврата на инвестиции (Return On Investment)

```
In [8]: # определение функции подсчёта пожизненной ценности Lifetime Value
# и возврата на инвестиции Return On Investment
# конверсия считается с накоплением по лайфтаймам
# =====
def get_ltv(
    profiles,          # профили пользователей
    purchases,         # данные о покупках
    observation_date, # момент анализа
    horizon_days,     # горизонт анализа
    dimensions=[],    # признаки когорт
    ignore_horizon=False # игнорировать горизонт
):
    # исключаем пользователей, не «доживших» до горизонта анализа
    last_suitable_acquisition_date = observation_date
    if not ignore_horizon:
        last_suitable_acquisition_date -= timedelta(
            days=(horizon_days - 1)
        )
    result_raw = profiles.query('dt <= @last_suitable_acquisition_date')

    # добавляем данные о покупках в профили
    result_raw = result_raw.merge(
        purchases[['user_id', 'event_dt', 'revenue']], on='user_id', how='left'
    )

    # рассчитываем лайфтайм 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 (по аналогии с CR)
    if len(dimensions) == 0:
        result_raw['cohort'] = 'All users'
        dimensions = dimensions + ['cohort']

    # функция группировки по желаемым признакам
    def group_by_dimensions(df, dims, horizon_days):
        # строим «треугольную» таблицу выручки
        result = df.pivot_table(
            index=dims, columns='lifetime', values='revenue', aggfunc='sum'
        )

        # находим сумму выручки с накоплением
        result = result.fillna(0).cumsum(axis=1)

        # вычисляем размеры когорт
        cohort_sizes = (
            df.groupby(dims)
            .agg({'user_id': 'nunique'})
            .rename(columns={'user_id': 'cohort_size'})
        )

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

        # считаем LTV: делим каждую «ячейку» в строке на размер когорты
        result = result.div(result['cohort_size'], axis=0)

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

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

        # собираем датасет с данными пользователей и значениями SAC,
        # добавляя параметры из dimensions
        cac = df[['user_id', 'acquisition_cost']] + dims.drop_duplicates()

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

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

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

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

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

        # в финальной таблице оставляем размеры когорт, SAC
        # и ROI в лайфтаймы, 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_raw, dimensions, horizon_days
    )
```

```

    )

# для таблицы 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,      # сырье данные
    result_grouped, # таблица LTV
    result_in_time, # таблица динамики LTV
    roi_grouped,   # таблица ROI
    roi_in_time,    # таблица динамики ROI
)

```

Функции визуализации бизнес-метрик

```
In [9]: # функция для сглаживания фрейма методом скользящего среднего
# =====
def filter_data(
    df,      # данные
    window   # размер скользящего окна
):
    # для каждого столбца применяем скользящее среднее
    for column in df.columns.values:
        df[column] = df[column].rolling(window).mean()
    return df
```

Функция визуализации удержания (Retention Rate) и его динамики

```
In [10]: # функция для визуализации удержания (Retention Rate)
# =====
def plot_retention(
    retention,          # таблица удержания
    retention_history, # таблица динамики удержания
    horizon, window=7, # размер окна сглаживания
    figsize=(15, 10)   # общий размер сетки графиков
):

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

    # исключаем размеры когорт и удержание первого дня
    retention = retention.drop(columns=['cohort_size', 0])

    # 8 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
    # 1. 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')

    # 2. 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')

    # 3. 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 ['dt', 'payer']
    ]
    # фильтруем данные и строим график
    filtered_data = retention_history.query('payer == True').pivot_table(
        index='dt', 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)
    )

    # 4. 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='dt', 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()

```

Функция визуализации конверсии (Conversion Rate) и её динамики

```

In [11]: # функция для визуализации конверсии (Conversion Rate)
# =====
def plot_conversion(
    conversion,           # таблица конверсии
    conversion_history,   # таблица динамики конверсии
    horizon, window=7,    # размер окна сглаживания
    figsize=(15, 10)      # общий размер сетки графиков
):

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

    # 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]]
    )

    # одна строка, две ячейки с общей осью y
    # 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')

    # 2. 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='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 конверсии пользователей на {}-й день'.format(horizon))

    plt.tight_layout()
    plt.show()

```

Функция визуализации пожизненной ценности (Lifetime Value), стоимости привлечения (Customer Acquisition Cost) и возврата на инвестиции (Return On Investment) и их динамики

```

In [12]: # функция для визуализации пожизненной ценности (Lifetime Value),
# возврата на инвестиции (Return On Investment)
# и динамики стоимости привлечения (Customer Acquisition Cost)
# =====
def plot_ltv_roi(
    ltv,                  # таблица пожизненной ценности
    ltv_history,          # таблица динамики пожизненной ценности
    roi,                 # таблица возврата на инвестиции
    roi_history,          # таблица динамики возврата на инвестиции
    horizon,              # горизонт анализа
    window=7,             # размер окна сглаживания
    figsize=(20, 10)      # общий размер сетки графиков
):

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

    # 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]]
    )

    # две строки по 3 ячейки:
    # 1. первая строка:
    # 1.1 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')

    # 1.2 второй график – динамика ltv

```

```

ax2 = plt.subplot(2, 3, 2, sharey=ax1)
# столбцами свободной 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))

# 1.3 третий график – динамика cac
ax3 = plt.subplot(2, 3, 3, sharey=ax1)
# столбцами свободной 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')

# 2. Вторая строка:
# 2.1 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')

# 2.2 пятый график – динамика roi
ax5 = plt.subplot(2, 3, 5, sharey=ax4)
# столбцами свободной 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()

```

Обзор данных

Обзор данных о посещениях пользователей

Обзор начнём с данных о посещениях пользователей, которые хранятся в файле `visits_info_short.csv`. Откроем данный файл, загрузим данные в таблицу `visits` и выведем первичную информацию о ней:

```
In [13]: visits = open_file('visits_info_short.csv')
data.observe(visits)
```

Произвольные строки таблицы:

User Id	Region	Device	Channel	Session Start	Session End
221319	65164586023	Germany	PC	WahooNetBanner	2019-05-29 07:06:15 2019-05-29 07:35:38
253354	279147101635	France	Android	OppleCreativeMedia	2019-08-01 05:36:35 2019-08-01 05:44:58
204726	641183000682	United States	Android	organic	2019-10-27 16:29:57 2019-10-27 16:39:23
231780	314266718699	France	Android	LeapBob	2019-06-21 10:34:19 2019-06-21 10:52:31
288919	942632965454	France	Android	WahooNetBanner	2019-09-28 04:30:56 2019-09-28 04:58:56

Информация о таблице:

```
=====
<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
```

Количество явных дубликатов в таблице:

```
=====
0
```

Процент пропусков в столбцах:

```
=====
```

	NaNs, %
User Id	0.00
Region	0.00
Device	0.00
Channel	0.00
Session Start	0.00
Session End	0.00

Таблица `visits` (лог сервера с информацией о посещениях сайта) содержит 309901 запись и состоит из 6 столбцов. Типы данных в столбцах - int64 и object.

Согласно описанию данных, столбцы хранят следующую информацию:

- `User Id` — уникальный идентификатор пользователя
- `Device` — категория устройства пользователя
- `Session start` — дата и время начала сессии
- `Session End` — дата и время окончания сессии
- `Channel` — идентификатор рекламного источника, из которого пришел пользователь
- `Region` - страна пользователя

Столбцы поименованы в смешанном стиле, нименования содержат пробелы. *На этапе предварительной обработки целесообразно переименовать столбцы к стилю snake_case для удобства дальнейшей работы.*

Столбцы `Device`, `Channel`, `Region` по своей природе содержат текстовые данные - приведение типов не требуется.

Столбец `User Id` содержит целые числа большой длины, поэтому приводить его к целому меньшей размерности для экономии памяти - также не целесообразно.

Столбцы `Session start` и `Session End` содержат текстовое представление даты и времени. *На этапе предварительной обработки целесообразно привести их к типу datetime.*

Явных дубликатов и пропусков данных в таблице не обнаружено.

Обзор данных о покупках пользователей

Перейдём к рассмотрению данных о покупках пользователей, которые хранятся в файле `orders_info_short.csv`. Откроем данный файл, загрузим данные в таблицу `orders` и выведем первичную информацию о ней:

```
In [14]: orders = open_file('orders_info_short.csv')
data.observe(orders)
```

Произвольные строки таблицы:

User Id	Event Dt	Revenue	
28309	656086686122	2019-05-17 08:50:04	4.99
29189	235802108235	2019-06-08 22:48:03	4.99
31402	675654762503	2019-07-20 07:51:21	4.99
10163	507041255670	2019-07-31 06:33:05	4.99
21587	766555279180	2019-10-20 14:55:26	4.99

Информация о таблице:

```
=====
<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
```

Количество явных дубликатов в таблице:

```
=====
0
```

Процент пропусков в столбцах:

	NaNs, %
User Id	0.00
Event Dt	0.00
Revenue	0.00

Таблица `orders` (информация о заказах) содержит 40212 записей и состоит из 3 столбцов. Типы данных в столбцах - int64, object и float64.

Согласно описанию данных, столбцы хранят следующую информацию:

- `User Id` — уникальный id пользователя, который сделал заказ
- `Event Dt` — дата и время покупки
- `Revenue` — выручка

Столбцы поименованы в смешанном стиле, нименования содержат пробелы. *На этапе предварительной обработки целесообразно переименовать столбцы к стилю snake_case для удобства дальнейшей работы.*

Столбец `User Id` содержит целые числа большой длины, поэтому приводить его к целому меньшей размерности для экономии памяти не целесообразно.

Столбец `Event Dt` содержит текстовое представление даты и времени. *На этапе предварительной обработки целесообразно привести его к типу datetime.*

Столбец `Revenue` содержит вещественные числа. Насколько можно судить по образцу данных, выручка по отдельному заказу, как правило, небольшая. Следовательно **на этапе предварительной обработки целесообразно оценить максимальный размер выручки в столбце `Revenue` и при возможности изменить его тип на `float32` для экономии памяти.**

Явных дубликатов и пропусков данных в таблице не обнаружено.

Обзор данных о рекламных расходах

Рассмотрим данные о рекламных расходах компании, которые хранятся в файле `costs_info_short.csv`. Откроем данный файл, загрузим данные в таблицу `costs` и выведем первичную информацию о ней:

```
In [15]: costs = open_file('costs_info_short.csv')
data.observe(costs)
```

Произвольные строки таблицы:

	dt	Channel	costs
1208	2019-09-06	LeapBob	10.71
1142	2019-07-02	LeapBob	10.71
1486	2019-06-16	WahooNetBanner	33.60
258	2019-07-18	MediaTornado	2.40
1606	2019-10-14	WahooNetBanner	30.60

Информация о таблице:

```
=====
<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
```

Количество явных дубликатов в таблице:

```
=====
0
```

Процент пропусков в столбцах:

```
=====
```

	NanS, %
dt	0.00
Channel	0.00
costs	0.00

Таблица `costs` (информация о затратах на маркетинг) содержит 1800 записей и состоит из 3 столбцов. Типы данных в столбцах - `object` и `float64`.

Согласно описанию данных, столбцы хранят следующую информацию:

- `Channel` — идентификатор рекламного источника
- `dt` — дата
- `costs` — затраты на этот рекламный источник в этот день

Столбцы поименованы в смешанном стиле. **На этапе предварительной обработки целесообразно переименовать столбцы к стилю `snake_case` для удобства дальнейшей работы.**

Столбец `Channel` по своей природе содержит текстовые данные - приведение типов не требуется.

Столбец `Dt` содержит текстовое представление даты. **На этапе предварительной обработки целесообразно привести его к типу `date`.**

Столбец `Costs` содержит вещественные числа. Затраты на рекламу могут выражаться большими числами, поэтому приведение типов для экономии памяти в данном случае не целесообразно.

Явных дубликатов и пропусков данных в таблице не обнаружено.

Выводы

1. Представленные для анализа данные описывают количественные (стоимость привлечения, выручка) и качественные (канал привлечения, регион, устройство) характеристики поведения пользователей приложения Procrastinate Pro+ в течение некоторого промежутка времени.

1. Данные имеют недостатки, которые следует, по возможности, устраниить на этапе предварительной обработки:

- смешаный стиль и пробелы в наименованиях столбцов;
- некорректные типы данных.

1. Предварительно можно утверждать, что данных достаточно для проведения исследования, в том числе, для расчёта бизнес-метрик.

1. В ходе первичного обзора технологических проблем с данными не обнаружено.

Замечание: в описании данных не указана валюта, в которой измеряется выручка и затраты на маркетинг. Для определённости в дальнейших исследованиях будем предполагать, что обе эти величины измеряются в одинаковых уровнях единицах (УЕ).

Однако **представляется целесообразным обратить внимание инженеров по данным на неполноту контекста исследования.**

Предварительная обработка данных

Переименование столбцов

Переименуем столбцы в таблицах в стиле `snake_case`:

```
In [16]: visits.rename(columns={'User Id':'user_id',
                             'Region':'region',
                             'Device':'device',
                             'Channel':'channel',
                             'Session Start':'session_start',
                             'Session End':'session_end'},
                             inplace=True)
orders.rename(columns={'User Id':'user_id',
                      'Event Dt':'event_dt',
                      'Revenue':'revenue'},
                      inplace=True)
costs.rename(columns={'Channel':'channel'},
                      inplace=True)

# проверка
for df in [visits, orders, costs]:
    print(df.columns)

Index(['user_id', 'region', 'device', 'channel', 'session_start',
       'session_end'],
      dtype='object')
Index(['user_id', 'event_dt', 'revenue'], dtype='object')
Index(['dt', 'channel', 'costs'], dtype='object')
```

Итак, столбцы переименованы в хорошем стиле `snake_case`.

Перейдём к приведению типов данных в таблицах.

Приведение типов данных

На этапе обзора данных мы отметили целесообразность приведения типов в следующих столбцах:

- в таблице `visits` столбцы `'session_start'` и `'session_end'` - к типу `datetime`;
- в таблице `orders` столбец `'event_dt'` - к типу `datetime`;
- в таблице `costs` столбец `'dt'` - к типу `date`.

Кроме того в таблице `orders` столбец `'revenue'` содержит вещественные числа, возможно, сильно ограниченные по абсолютной величине, поэтому признано целесообразным оценить максимальный размер выручки в данном столбце и при возможности изменить его тип на `float32` для экономии памяти.

```
In [17]: # приведение типов дат в таблице visits
visits['session_start'] = pd.to_datetime(visits['session_start'])
visits['session_end'] = pd.to_datetime(visits['session_end'])

# приведение типов дат в таблице orders
orders['event_dt'] = pd.to_datetime(orders['event_dt'])

# приведение типов дат в таблице costs
costs['dt'] = pd.to_datetime(costs['dt']).dt.date
```

```
In [18]: # проверка результатов
print(visits['session_start'].dtype, visits['session_end'].dtype)
print(orders['event_dt'].dtype)
print(costs['dt'].dtype)
costs.head(1)
```

```
datetime64[ns] datetime64[ns]
datetime64[ns]
object
```

```
Out[18]:   dt   channel  costs
0  2019-05-01  FaceBoom  113.3
```

Типы данных для столбцов с датами и временем приведены. Рассмотрим значения столбца `'revenue'` таблицы `orders`:

```
In [19]: orders['revenue'].describe()
```

```
Out[19]: count    40212.000000
mean      5.370608
std       3.454208
min      4.990000
25%     4.990000
50%     4.990000
75%     4.990000
max     49.990000
Name: revenue, dtype: float64
```

Очевидно, размер типа данных `float64` избытен для хранения значений этого столбца, приведём его к типу `float32` для экономии памяти:

```
In [20]: orders['revenue'] = orders['revenue'].astype('float32')
orders['revenue'].dtype
```

```
Out[20]: dtype('float32')
```

Проверка категориальных столбцов на неявные дубликаты

Проверим значения столбцов `'region'`, `'device'`, `'channel'` таблицы `visits` на наличие неявных дубликатов:

```
In [21]: visits['region'].sort_values().unique()
```

```
Out[21]: array(['France', 'Germany', 'UK', 'United States'], dtype=object)
```

```
In [22]: visits['device'].sort_values().unique()
```

```
Out[22]: array(['Android', 'Mac', 'PC', 'iPhone'], dtype=object)
```

```
In [23]: visits['channel'].sort_values().unique()
```

```
In [23]: array(['AdNonSense', 'FaceBoom', 'LeapBob', 'MediaTornado',
   'OppleCreativeMedia', 'RocketSuperAds', 'TipTop', 'WahooNetBanner',
   'YRabbit', 'lambdaMediaAds', 'organic'], dtype=object)
```

В категориальных столбцах таблицы `visits` неявных дубликатов не обнаружено.

Проверим на наличие неявных дубликатов категориальный столбец `'channel'` таблицы `costs`:

```
In [24]: costs['channel'].sort_values().unique()
```

```
Out[24]: array(['AdNonSense', 'FaceBoom', 'LeapBob', 'MediaTornado',
   'OppleCreativeMedia', 'RocketSuperAds', 'TipTop', 'WahooNetBanner',
   'YRabbit', 'lambdaMediaAds'], dtype=object)
```

В категориальных столбцах таблицы `costs` неявных дубликатов не обнаружено.

Проверим наличие всех каналов таблицы `costs` среди источников привлечения реальных пользователей:

```
In [25]: for channel in costs['channel'].unique():
    if channel in visits['channel'].unique():
        print(f'Канал {channel} присутствует')
    else:
        print(f'---> Канал {channel} ОТСУТСТВУЕТ!')
```

Канал FaceBoom присутствует
 Канал MediaTornado присутствует
 Канал RocketSuperAds присутствует
 Канал TipTop присутствует
 Канал YRabbit присутствует
 Канал AdNonSense присутствует
 Канал LeapBob присутствует
 Канал OppleCreativeMedia присутствует
 Канал WahooNetBanner присутствует
 Канал lambdaMediaAds присутствует

Итак, все каналы, в которых компания размещала рекламу, привлекли реальных пользователей.

Выводы

1. В ходе предварительной обработки данных мы выполнили:

- переименование столбцов;
- изменение типов данных, связанных с датой и временем событий;
- проверку на неявные дубликаты категориальных столбцов таблиц `visits` и `costs`.

1. Колонки таблиц теперь поименованы корректно, в соответствии со стилем "snake_case".

1. Столбцы, содержащие дату и время событий приведены к типу `datetime64[ns]`.

1. Неявных дубликатов категориальных столбцов в таблицах `visits` и `costs` не выявлено.

1. Все каналы, в которых компания размещала рекламу, привлекли реальных пользователей.

Исследовательский анализ данных

Создание профилей пользователей

Для создания профилей пользователей используем данные журнала посещений `visits`, а также журнала заказов `orders`:

```
In [26]: profile = get_profiles(visits, orders, costs)
profile.sample(10)
```

	user_id	first_ts	channel	device	region	dt	month	payer	acquisition_cost
112863	752096258930	2019-09-10 04:22:40	OppleCreativeMedia	Android	France	2019-09-10	2019-09-01	False	0.243750
135869	906107313634	2019-06-27 05:28:10		TipTop	iPhone	United States	2019-06-27	2019-06-01	False
145828	972645191346	2019-08-14 02:38:24		WahooNetBanner	Android	France	2019-08-14	2019-08-01	False
5150	34367105313	2019-08-08 01:54:58		TipTop	Mac	United States	2019-08-08	2019-08-01	False
113400	755873697861	2019-08-25 22:58:26		organic	PC	UK	2019-08-25	2019-08-01	False
122589	817348874608	2019-05-11 17:59:55		lambdaMediaAds	Android	UK	2019-05-11	2019-05-01	False
51874	344046622415	2019-10-21 06:53:34		organic	iPhone	United States	2019-10-21	2019-10-01	False
58771	389988613944	2019-10-08 22:58:33		organic	PC	United States	2019-10-08	2019-10-01	False
145841	972738730998	2019-09-11 20:59:12		WahooNetBanner	PC	France	2019-09-11	2019-09-01	True
92971	618879486393	2019-05-16 12:19:44		TipTop	iPhone	United States	2019-05-16	2019-05-01	False

Для дальнейшего анализа определим минимальную и максимальную даты привлечения пользователей:

```
In [27]: min_acq_date = profile['dt'].min()
max_acq_date = profile['dt'].max()

print('Минимальная дата привлечения:', min_acq_date)
print('Максимальная дата привлечения:', max_acq_date)
```

Минимальная дата привлечения: 2019-05-01

Максимальная дата привлечения: 2019-10-27

В имеющихся данных содержится информация о пользователях, привлечённых с 1 мая 2019 года по 27 октября 2019 года.

Изучение профилей в разрезе регионов привлечения

Выясним, из каких стран привлечены пользователи приложения, а также какие страны дают большее количество платящих пользователей:

```
In [28]: payers_study(profile, by='region')
```

region	user_count	payer_count	payer_percent
United States	100002	6902	6.90
Germany	14981	616	4.11
UK	17575	700	3.98
France	17450	663	3.80

По количеству пользователей и доле плательщиков лидируют США. Германия, несмотря на то, что в ней привлечено меньше всего пользователей, находится на втором месте по доле платящих. Замыкают список регионов Великобритания и Франция, обладающие близкими значениями как общего количества привлечённых пользователей, так и доли плательщиков.

Изучение профилей в разрезе используемых устройств

Выясним, какими устройствами пользуются клиенты, а также с каких устройств чаще всего заходят платящие пользователи:

```
In [29]: payers_study(profile, by='device')
```

device	user_count	payer_count	payer_percent
Mac	30042	1912	6.36
iPhone	54479	3382	6.21
Android	35032	2050	5.85
PC	30455	1537	5.05

Очевидно, чаще всего деньги в приложении тратят пользователи устройств Apple. За ними следуют пользователи Android и PC. При этом любопытно, что пользователи Mac, несмотря на наименьшее их абсолютное количество, наиболее часто склонны совершать покупки в приложении.

Изучение профилей в разрезе каналов привлечения

Изучим, по каким рекламным каналам шло привлечение пользователей, а также какие каналы приносят больше всего платящих пользователей:

```
In [30]: payers_study(profile, by='channel')
```

channel	user_count	payer_count	payer_percent
FaceBoom	29144	3557	12.20
AdNonSense	3880	440	11.34
lambdaMediaAds	2149	225	10.47
TipTop	19561	1878	9.60
RocketSuperAds	4448	352	7.91
WahooNetBanner	8553	453	5.30
YRabbit	4312	165	3.83
MediaTornado	4364	156	3.57
LeapBob	8553	262	3.06
OppleCreativeMedia	8605	233	2.71
organic	56439	1160	2.06

Компания размещала рекламу в 10 рекламных агентствах. Наибольшее количество пользователей (top-5) привлекли агентства FaceBoom, TipTop, OppleCreativeMedia, LeapBob и WahooNetBanner.

Однако, не все привлечённые этими каналами пользователи хорошо сконвертировались в плательщиков: лидерами по конвертации (top-5) являются FaceBoom, AdNonSense, lambdaMediaAds, TipTop и RocketSuperAds.

Как видим, только два из пяти лидеров в абсолютном привлечении попали в top-5 по конверсии.

Наибольшее количество пользователей (56439) пришло в приложение естественным путём (organic). Вместе с тем, данный канал демонстрирует наихудшую конвертацию в покупки.

Изучение регионов деятельности каналов привлечения

Составим своеобразную "карту ответственности" рекламных каналов:

```
In [31]: profile.pivot_table(index='channel',
                           columns='region',
                           values='user_id',
                           aggfunc='nunique').fillna(0)
```

	region	France	Germany	UK	United States
channel					
AdNonSense	1366.0	1219.0	1295.0	0.0	
FaceBoom	0.0	0.0	0.0	29144.0	
LeapBob	2988.0	2519.0	3046.0	0.0	
MediaTornado	0.0	0.0	0.0	4364.0	
OppleCreativeMedia	2991.0	2562.0	3052.0	0.0	
RocketSuperAds	0.0	0.0	0.0	4448.0	
TipTop	0.0	0.0	0.0	19561.0	
WahooNetBanner	2971.0	2579.0	3003.0	0.0	
YRabbit	0.0	0.0	0.0	4312.0	
lambdaMediaAds	742.0	649.0	758.0	0.0	
organic	6392.0	5453.0	6421.0	38173.0	

Очевидно, каналы делятся на 2 больших группы:

- на европейскую аудиторию работают AdNonSense, LeapBob, OppleCreativeMedia, WahooNetBanner, lambdaMediaAds;
- на американскую - FaceBoom, MediaTornado, RocketSuperAds, TipTop, YRabbit.

Представляется целесообразным учитывать это в дальнейших исследованиях.

Кроме того, отметим, что большая часть "органических" пользователей пришла в регионе США.

Определение наиболее часто используемых устройств по регионам

Определим, в каких странах какие устройства более популярны:

```
In [32]: profile.pivot_table(index='device',
                           columns='region',
                           values='user_id',
                           aggfunc='nunique').fillna(0)
```

	region	France	Germany	UK	United States
device					
Android	5252	4440	5141	20199	
Mac	1838	1584	1778	24842	
PC	6940	5945	7138	10432	
iPhone	3420	3012	3518	44529	

Во всех регионах мобильные устройства более популярны по сравнению с настольными платформами:

- в Европе отдают предпочтение Android;
- в США, естественно, - Apple.

В разрезе настольных систем ситуация аналогична: европейские пользователи предпочитают PC, американские - Mac.

Выводы

В процессе изучения профилей пользователей приложения Procrastinate Pro+ мы выяснили, что:

- В анализируемых данных временной диапазон привлечения пользователей составляет с 2019-05-01 по 2019-10-27.
- По количеству привлечённых пользователей и доле плательщиков лидируют США. Германия, несмотря на то, что в ней привлечено меньше всего пользователей, находится на втором месте по доле платящих. Замыкают список регионов Великобритания и Франция, обладающие близкими значениями как общего количества привлечённых пользователей, так и доли плательщиков.
- Чаще всего деньги в приложении тратят пользователи устройств Apple. За ними следуют пользователи Android и PC. При этом пользователи Mac, несмотря на наименьшее их абсолютное количество, наиболее часто склонны совершать покупки в приложении.
- Компания размещала рекламу в 10 рекламных агентствах. Только два из пяти лидеров в абсолютном привлечении попали в top-5 по конверсии.
- Наибольшее количество пользователей пришло в приложение естественным путём. Вместе с тем, данный канал демонстрирует наихудшую конвертацию в покупки.
- Рекламные каналы делятся на две равнозначные группы по региональному признаку - с целевой аудиторией в США и Европе.
- Во всех регионах мобильные устройства более популярны по сравнению с настольными платформами.
- В Европе отдают предпочтение Android и PC, в США - iPhone и Mac.

Исследование маркетинговых затрат

Общая сумма и распределение по источникам затрат на маркетинг

Посчитаем общую сумму затрат на маркетинг:

```
In [33]: total_ad_costs = costs['costs'].sum().round(2)
```

```
Out[33]: 105497.3
```

За весь период на рекламу было потрачено 105497.30 YЕ. Проверим, какая доля из них была потрачена в Европе, а какая - в США:

```
In [34]: usa_channels = ['FaceBoom', 'MediaTornado', 'RocketSuperAds',
                     'TipTop', 'YRabbit']
europe_channels = ['AdNonSense', 'LeapBob', 'OppleCreativeMedia',
                   'WahooNetBanner', 'lambdaMediaAds']

print('Доля США в общих затратах (%):',
      (costs.query('channel in @usa_channels')['costs'].sum() * 100 /
       total_ad_costs).round(2))
)

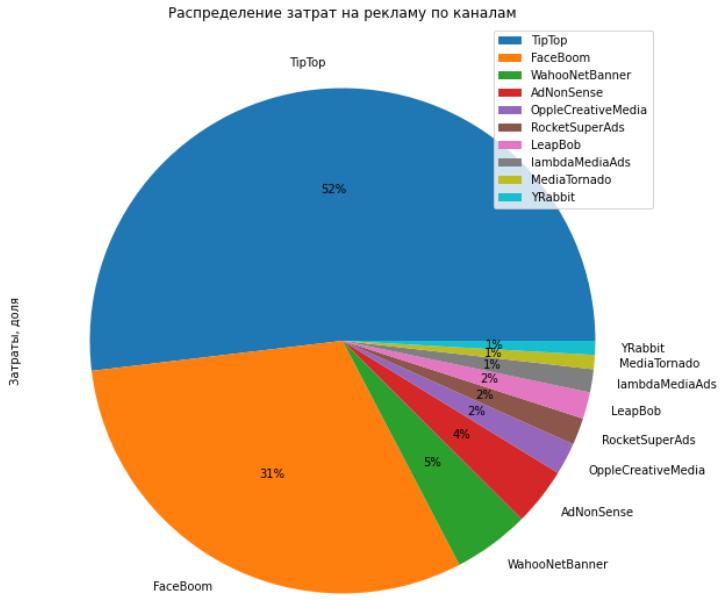
print('Доля Европы в общих затратах (%):',
      (costs.query('channel in @europe_channels')['costs'].sum() * 100 /
       total_ad_costs).round(2))
)
```

Доля США в общих затратах (%): 86.19
Доля Европы в общих затратах (%): 13.81

Более 86% рекламного бюджета потрачено в **США!** Этот регион **будем считать первым под подозрением о неэффективном расходовании рекламных средств.**

Построим распределение рекламных расходов по каждому из каналов:

```
In [35]: (
    costs
    .groupby(by='channel')
    .agg({'costs':'sum'})
    .sort_values(by='costs', ascending=False)
).plot(y='costs',
       kind='pie',
       figsize=(10,10),
       autopct='%1.0f%%', # визуализация долей на диаграмме
       title='Распределение затрат на рекламу по каналам',
       ylabel='Затраты, доля')
plt.show()
```



Более половины рекламного бюджета было потрачено в канале TipTop, около трети - в канале FaceBoom (оба действуют на американском рынке), остальное, предварительно, более-менее равномерно распределено между прочими рекламными каналами.

Построим распределения отдельно для Европы и Америки:

```
In [36]: # задаём размер сетки для графиков
plt.figure(figsize=(20, 10))

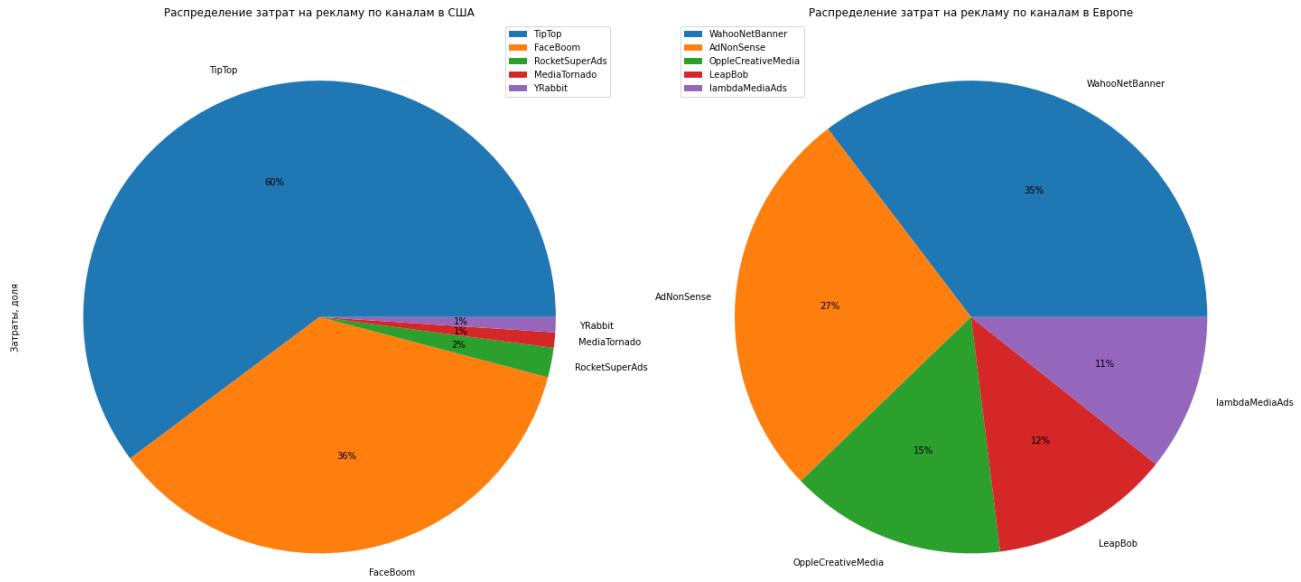
# 1. первый график – Распределение в США
ax1 = plt.subplot(1, 2, 1)
(
    costs.query('channel in @usa_channels')
    .groupby(by='channel')
    .agg({'costs':'sum'})
    .sort_values(by='costs', ascending=False)
).plot(y='costs',
       kind='pie',
       title='Распределение затрат на рекламу по каналам в США',
       ylabel='Затраты, доля',
       autopct='%1.0f%%', # визуализация долей на диаграмме
       ax=ax1)
plt.tight_layout()

# 2. второй график – Распределение в Европе
ax2 = plt.subplot(1, 2, 2)
(
    costs.query('channel in @europe_channels')
    .groupby(by='channel')
    .agg({'costs':'sum'})
    .sort_values(by='costs', ascending=False)
).plot(y='costs',
       kind='pie',
       title='Распределение затрат на рекламу по каналам в Европе',
       ylabel='',
       autopct='%1.0f%%', # визуализация долей на диаграмме
       ax=ax2)
```

```

ax=ax2)
plt.tight_layout()
plt.show()

```



Львиная доля рекламного бюджета в США тратится в каналах **TipTop** и **FaceBoom** (60% и 36% соответственно) - наши главные подозреваемые в этом регионе.

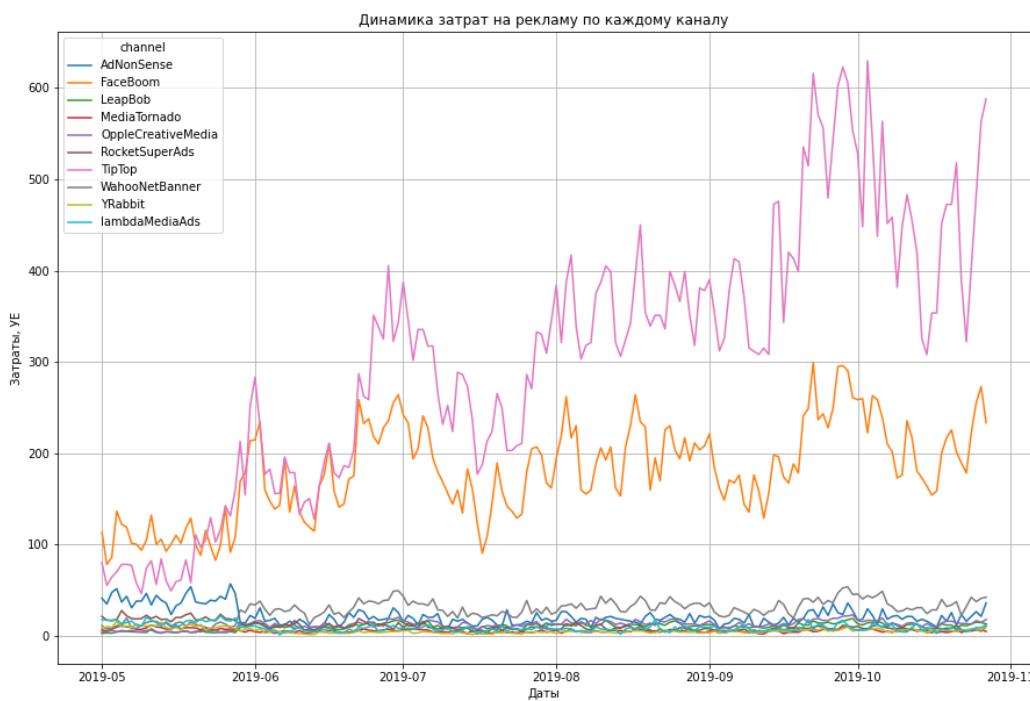
В Европе большая часть бюджета расходуется в каналах **AdNonSense** и **WahooNetBanner** (27% и 35% соответственно).

Построим распределение по времени рекламных расходов по каждому из каналов:

```

In [37]: (
    costs
    .pivot_table(index='dt',
                 columns='channel',
                 values='costs',
                 aggfunc='sum')
    .plot(grid=True,
          figsize=(15,10),
          title='Динамика затрат на рекламу по каждому каналу',
          xlabel='Даты',
          ylabel='Затраты, УЕ')
plt.show()

```



Из полученного графика распределений следует, что больше всего денег компания вкладывала в рекламу на каналах TipTop и FaceBoom, причём затраты на TipTop растут опережающими темпами.

Затраты на рекламу в канале AdNonSense занимали 3-ю позицию в мае 2019 года, но потом были урезаны в пользу WahooNetBanner.

Затраты по остальным каналам в среднем остаются постоянными в диапазоне до 30 УЕ на протяжении всего периода.

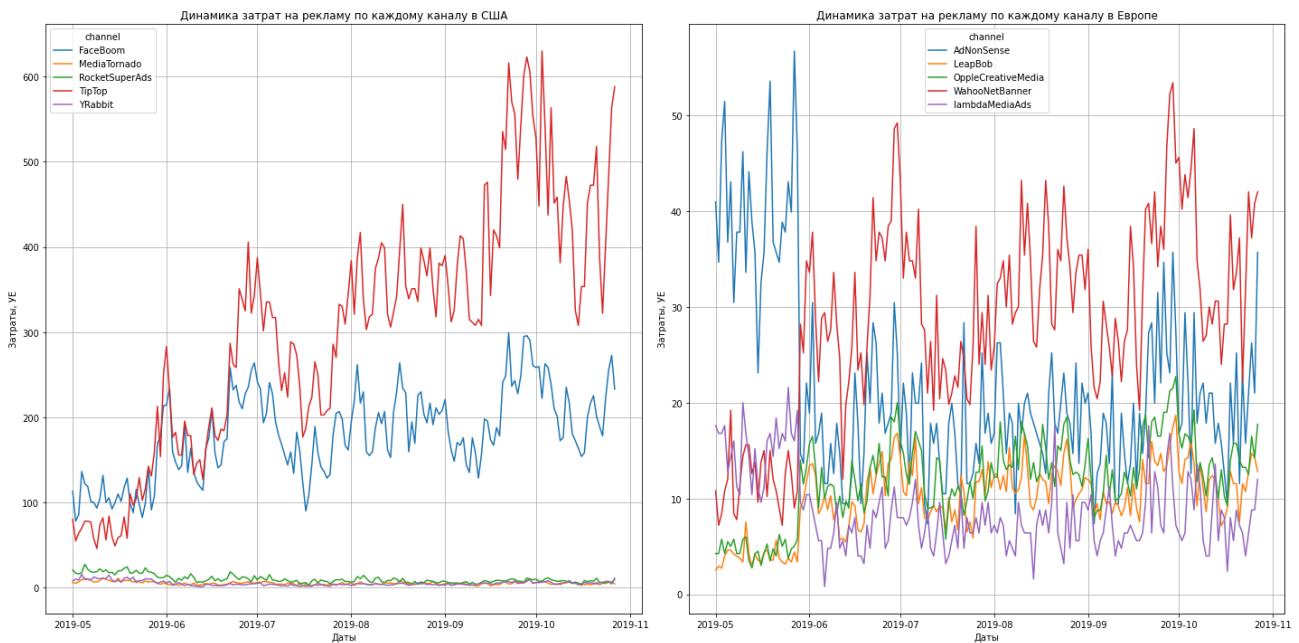
Следует отметить, что, как было отмечено ранее, источник FaceBoom привлек большее количество пользователей, нежели TipTop, а WahooNetBanner занимает 5-ю позицию по количеству привлечённых пользователей. С позиции же конверсии пользователей в плательщиков лидером является FaceBoom, а TipTop занимает только 4-ю позицию.

Рассмотрим динамику затрат по каналам в привязке к регионам:

```
In [38]: # задаём размер сетки для графиков
plt.figure(figsize=(20, 10))

# 1. первый график – Распределение в США
ax1 = plt.subplot(1, 2, 1)
(
    costs.query('channel in @usa_channels')
    .pivot_table(index='dt',
                 columns='channel',
                 values='costs',
                 aggfunc='sum')
).plot(grid=True,
#           figsize=(15,10),
#           title='Динамика затрат на рекламу по каждому каналу в США',
#           xlabel='Даты',
#           ylabel='Затраты, УЕ',
#           ax=ax1)
plt.tight_layout()

# 2. второй график – Распределение в Европе
ax2 = plt.subplot(1, 2, 2)
(
    costs.query('channel in @europe_channels')
    .pivot_table(index='dt',
                 columns='channel',
                 values='costs',
                 aggfunc='sum')
).plot(grid=True,
#           figsize=(15,10),
#           title='Динамика затрат на рекламу по каждому каналу в Европе',
#           xlabel='Даты',
#           ylabel='Затраты, УЕ',
#           ax=ax2)
plt.tight_layout()
plt.show()
```



Наши предварительные наблюдения по рынкам США и Европы подтвердились.

Расчёт средней стоимости привлечения (CAC)

Стоимость привлечения рассчитывается путём деления общих расходов на рекламу на количество привлечённых с её помощью пользователей. Общие расходы на весь рекламный проект мы посчитали выше. Разделим их на количество привлеченных пользователей (исключим пользователей канала organic, поскольку они пришли не из рекламы):

```
In [39]: cac_total = (
    costs['costs'].sum() /
    profile.query('channel != "organic"')['user_id'].nunique()
).round(2)

cac_total
```

```
Out[39]: 1.13
```

Итак, средняя стоимость привлечения пользователя в рамках всего рекламного проекта составляет 1.13 УЕ. Посмотрим в разбивке по регионам:

```
In [40]: print('Средняя стоимость привлечения в США:',
        (costs.query('channel in @usa_channels')['costs'].sum() /
         profile.query('channel in @usa_channels')['user_id'].nunique()).round(2)
        )

print('Средняя стоимость привлечения в Европе:',
        (costs.query('channel in @europe_channels')['costs'].sum() /
         profile.query('channel in @europe_channels')['user_id'].nunique()).round(2)
        )
```

```
Средняя стоимость привлечения в США: 1.47
Средняя стоимость привлечения в Европе: 0.46
```

Средняя стоимость привлечения "неорганического" пользователя в США более чем втрое превышает стоимость привлечения в Европе.

Посчитаем среднюю стоимость привлечения пользователя каждым каналом:

```
In [41]: cac_by_channel = (
    profile.query('channel != "organic"') # сгруппируем "неорганических"
    .groupby(by='channel') # пользователей по каналам
    .agg({'user_id': 'nunique'})
    .join(
        costs # сгруппируем затраты по каналам
        .groupby(by='channel')
        .agg({'costs': 'sum'}))
)
.rename({'user_id': 'user_count',
         'costs': 'total_costs'},
        axis=1)
)

cac_by_channel['cac'] = ( # Вычислим САС
    cac_by_channel['total_costs']
    .div(cac_by_channel['user_count'])
    .round(2)
)

cac_by_channel.sort_values(by='cac', ascending=False)
```

Out[41]:

channel	user_count	total_costs	cac
TipTop	19561	54751.30	2.80
FaceBoom	29144	32445.60	1.11
AdNonSense	3880	3911.25	1.01
LambdaMediaAds	2149	1557.60	0.72
WahooNetBanner	8553	5151.00	0.60
RocketSuperAds	4448	1833.00	0.41
OppleCreativeMedia	8605	2151.25	0.25
MediaTornado	4364	954.48	0.22
YRabbit	4312	944.22	0.22
LeapBob	8553	1797.60	0.21

Значение САС в канале TipTop более чем вдвое превышает удельные затраты в остальных каналах. При этом уже для FaceBoom, занимающего второе место по значению САС, это самое значение примерно равно среднему по рекламному проекту.

В этой связи можно заключить, что наши подозрения относительно нерациональных рекламных расходов в канале TipTop только укрепляются.

Выводы

- За весь период на рекламу было потрачено 105497.30 УЕ.
- Более 86% рекламного бюджета потрачено в **США!** Этот регион **будем считать первым под подозрением о неэффективном расходовании рекламных средств.**
 - Более 50% рекламного бюджета было потрачено в канале **TipTop**, около трети - в канале **FaceBoom**. Это **главные подозреваемые в регионе США.**
 - в ходе дальнейшего исследования **целесообразно оценить эффект от перераспределения затрат на рекламу в каналах AdNonSense и WahooNetBanner в Европе.**
- Больше всего денег компания вкладывала в рекламу на каналах TipTop и FaceBoom, причём затраты на TipTop растут с течением времени опережающими темпами.
- Значение САС в канале TipTop более чем вдвое превышает удельные затраты в остальных каналах. При этом уже для FaceBoom, занимающего второе место по значению САС, это самое значение примерно равно среднему по рекламному проекту.

Оценка окупаемости рекламы

Для оценки окупаемости рекламы будем применять когортный анализ таких бизнес-метрик, как "пожизненная ценность" (Lifetime value, LTV), стоимость привлечения (Customer Acquisition Cost, CAC), а также возврат на инвестиции (Return On Investment, ROI).

Для расчёта этих метрик используются следующие формулы:

$$LTV = \frac{CDR}{CS},$$

где:

- CDR - общая выручка на текущий день (вычисляется с накоплением);
- CS - размер когорты.

$$CAC = \frac{AC}{CS},$$

где:

- AC - общие расходы на рекламу на день привлечения;
- CS - размер когорты.

$$ROI = \frac{LTV}{CAC}.$$

Замечание: Легко видеть, что учёт при расчёте ROI пользователей, пришедших из канала "organic", приведёт к потенциально бесконечному росту данной метрики, поскольку для таких пользователей стоимость привлечения CAC = 0. С другой стороны, ранее мы отмечали, что, несмотря на то, что данному каналу принадлежит наибольшее количество новых пользователей, они обладают минимальной конверсией в покупатели и, следовательно, потенциально вносят минимальный вклад в LTV. Возникает неопределённая ситуация.

Поэтому, без ограничения общности, считать LTV, CAC и ROI будем **без учёта "органических" пользователей**.

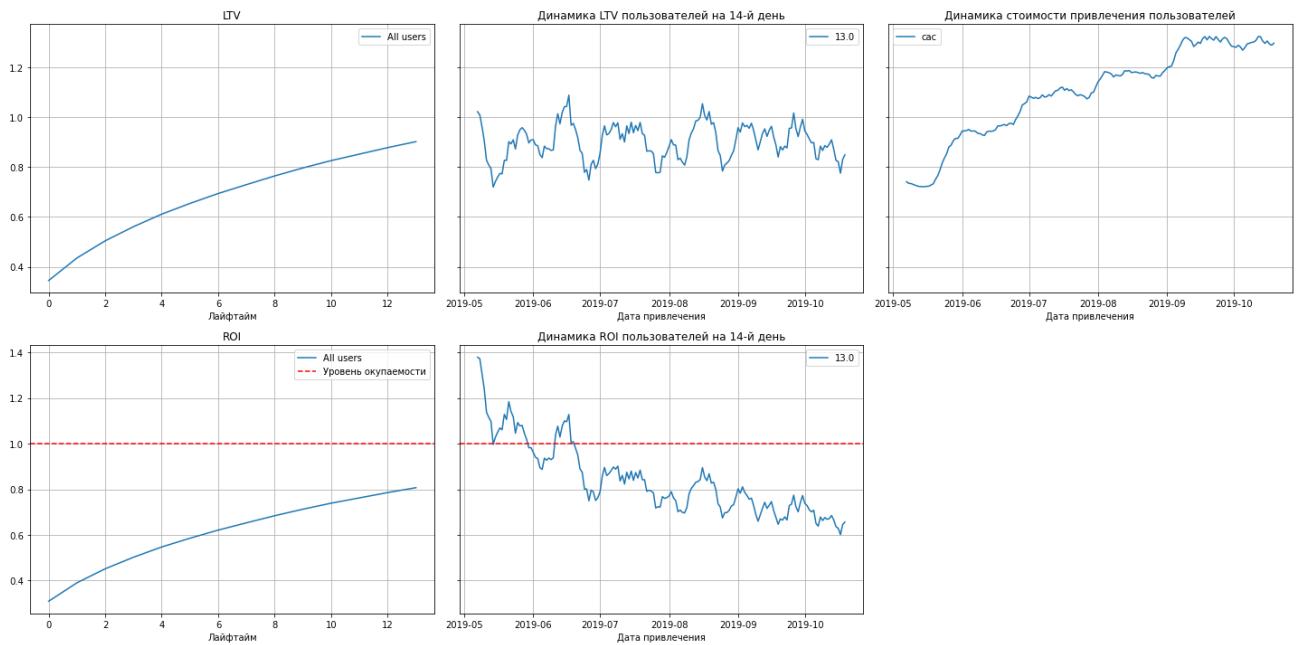
Момент анализа, согласно контекста, установим **1 ноября 2019 года**, горизонт анализа - **2 недели (14 дней)**, за которые предположительно должны окупаться вложения в рекламу.

Анализ общей окупаемости рекламы

Ранее, при формировании профилей пользователей, мы уже посчитали удельную стоимость привлечения по каждому источнику для каждой даты.

Воспользуемся функцией `get_ltv()` для расчёта всех трёх метрик в абсолютных значениях и в динамике, которые затем визуализируем с помощью функции `plot_ltv_roi()`:

```
In [42]: ltv_raw, ltv, ltv_din, roi, roi_din = get_ltv()
    profile.query('channel != "organic"'), # профили "неорганических" пользователей
    orders,                                # данные о покупках
    datetime(2019,11,1).date(),            # момент анализа
    14,                                     # горизонт анализа
)
plot_ltv_roi(
    ltv,                                    # таблица пожизненной ценности
    ltv_din,                                # таблица динамики пожизненной ценности
    roi,                                    # таблица возврата на инвестиции
    roi_din,                                # таблица динамики возврата на инвестиции
    14,                                     # горизонт анализа
    window=7,                               # размер окна сглаживания
    figsize=(20, 10)                         # общий размер сетки графиков
)
```



Из графиков следует, что:

- расходы на привлечение по всем "неорганическим" пользователям с течением времени растут быстрее, чем прибыль от их покупок;
- в этой связи динамика ROI на 14-й день демонстрирует отрицательный тренд, несмотря на то, что динамика LTV на 14-й день стабильно колеблется в районе 0.9 ЕЕ по всем когортам;
- среднее значение прибыли на 1 пользователя когорты на границу второй недели менее 1 ЕЕ;
- в целом окупаемость инвестиций на границу второй недели находится на уровне 80%.

Общий вывод - расходы на рекламных пользователей в целом **не окупаются** в течение первых двух недель.

Анализ окупаемости рекламы по каналам привлечения

Воспользуемся функцией `get_ltv()` для расчёта LTV, CAC и ROI в разрезе каналов привлечения в абсолютных значениях и в динамике, которые затем визуализируем с помощью функции `plot_ltv_roi()`:

```
In [43]: # анализ окупаемости по каналам в целом
ltv_raw, ltv, ltv_din, roi, roi_din = get_ltv()
    profile.query('channel != "organic"'), # профили "неорганических" пользователей
    orders,                                # данные о покупках
    datetime(2019,11,1).date(),            # момент анализа
    14,                                     # горизонт анализа
    dimensions=['channel']                 # признаки когорт
)
plot_ltv_roi(
    ltv,                                    # таблица пожизненной ценности
    ltv_din,                                # таблица динамики пожизненной ценности
    roi,                                    # таблица возврата на инвестиции
    roi_din,                                # таблица динамики возврата на инвестиции
    14,                                     # горизонт анализа
    window=14,                             # размер окна сглаживания
)
```



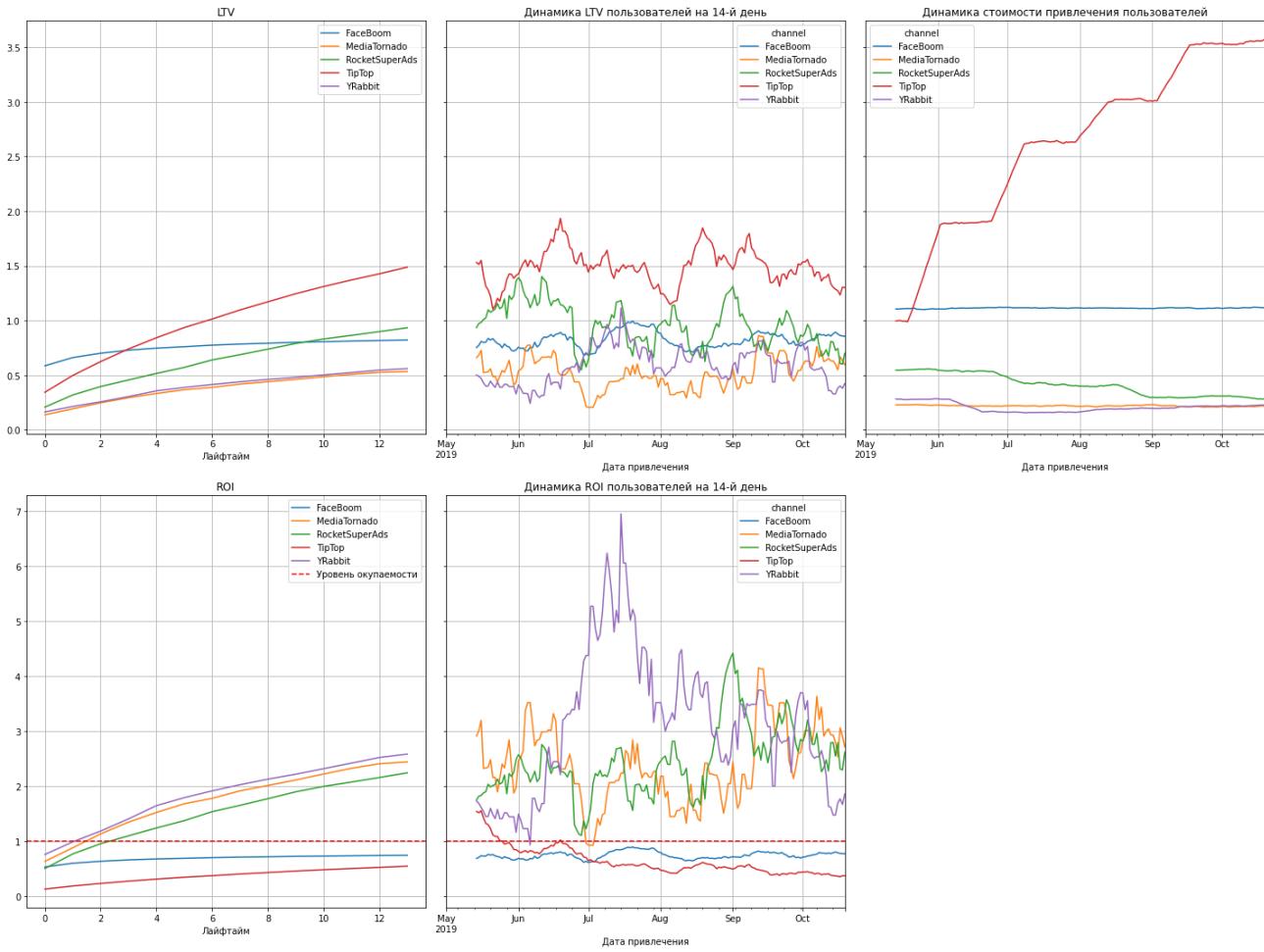
Анализируя графики LTV, CAC и ROI в разрезе каналов привлечения, можно видеть:

- из всех каналов привлечения только lambdaMediaAds и TipTop получают на конец второй недели среднюю выручку с пользователя выше 1 Е;
- при этом в динамике LTV канал lambdaMediaAds стабильно опережает другие каналы привлечения на протяжении всего периода наблюдений;
- канал TipTop стабильно занимает вторую позицию в динамике LTV, однако ежемесячные повышения стоимости привлечения на данном канале (на 0.5-1 Е на пользователя) не позволяют ему выйти на окупаемость (на конец второй недели ROI составляет около 50%);
- большинство каналов (за исключением TipTop, FaceBoom и AdNonSense) окупается в течение первой недели и далее начинают приносить прибыль;
- это подтверждает и динамика ROI - для TipTop, FaceBoom и AdNonSense на протяжении практически всего периода наблюдений ROI 14-го дня не выходит за уровень окупаемости.

Проверим наши наблюдения отдельно для каналов, действующих в нашем главном подозреваемом регионе (США) и в Европе:

```
In [44]: # анализ окупаемости по каналам в США
ltv_raw, ltv, ltv_din, roi, roi_din = get_ltv(
    profile.query('channel in @usa_channels'), # профили "неорганических" пользователей в США
    orders, # данные о покупках
    datetime(2019, 11, 1).date(), # момент анализа
    14, # горизонт анализа
    dimensions=['channel'] # признаки когорт
)

plot_ltv_roi(
    ltv, # таблица пожизненной ценности
    ltv_din, # таблица динамики пожизненной ценности
    roi, # таблица возврата на инвестиции
    roi_din, # таблица динамики возврата на инвестиции
    14, # горизонт анализа
    window=14, # размер окна сглаживания
    figsize=(20, 15) # общий размер сетки графиков
)
```



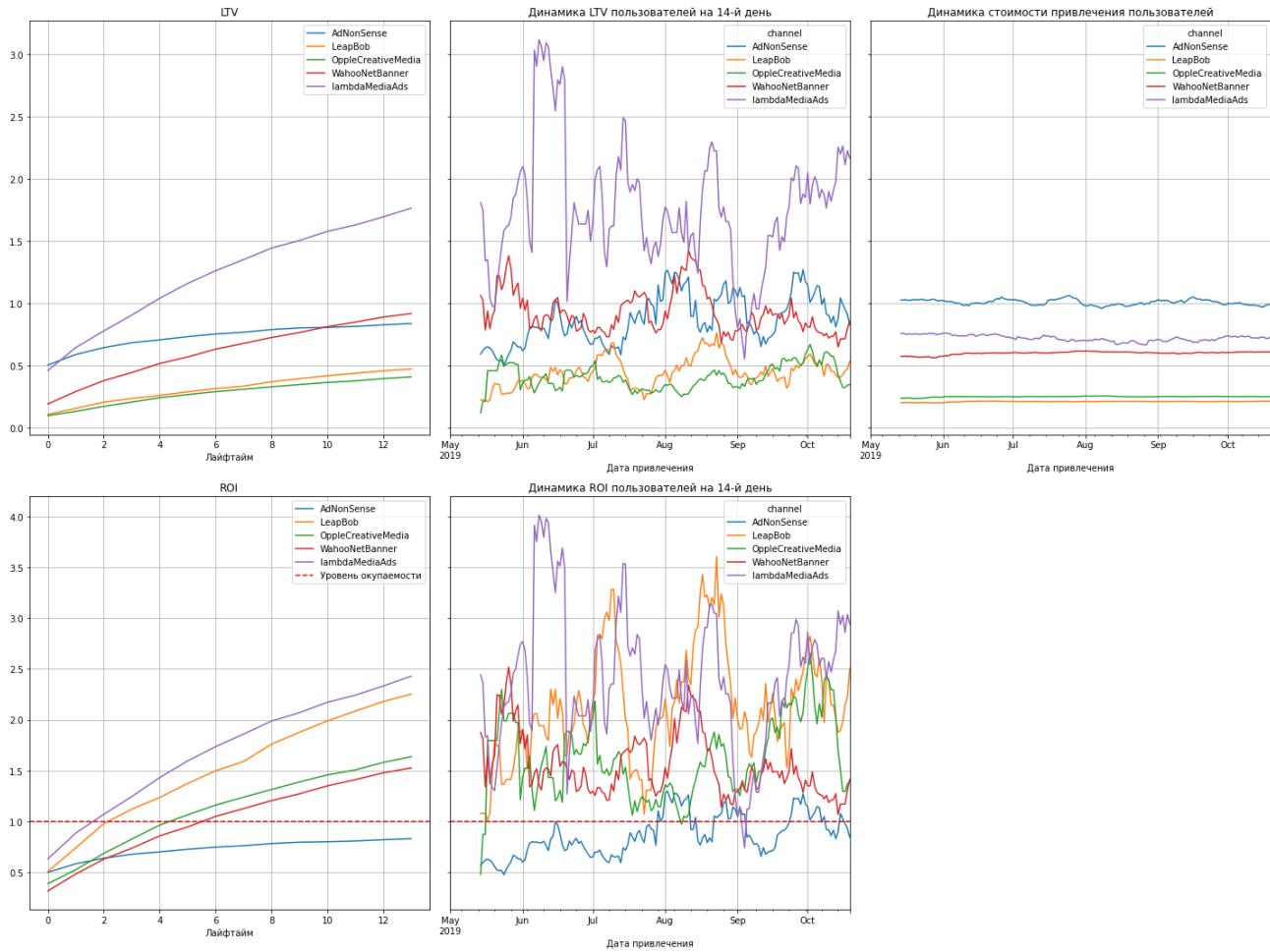
Данные графиков подтверждают, что в США не окупаются два канала привлечения: TipTop и FaceBoom. Однако причины неокупаемости для этих каналов различны:

- TipTop не окупается из-за постоянно возрастающей стоимости рекламы;
- стоимость привлечения в канале FaceBoom постоянна, слабый рост LTV в течение первых двух недель (возможно, пользователи, привлечённые этим источником, уходят из приложения).

Остальные каналы в США окупаются за первые 4 дня.

```
In [45]: # анализ окупаемости по каналам в Европе
ltv_raw, ltv, ltv_din, roi, roi_din = get_ltv(
    profile.query('channel in @europe_channels'), # профили "неорганических" пользователей в Европе
    orders, # данные о покупках
    datetime(2019, 11, 1).date(), # момент анализа
    14, # горизонт анализа
    dimensions=['channel'] # признаки когорт
)

plot_ltv_roi(
    ltv, # таблица пожизненной ценности
    ltv_din, # таблица динамики пожизненной ценности
    roi, # таблица возврата на инвестиции
    roi_din, # таблица динамики возврата на инвестиции
    14, # горизонт анализа
    window=14, # размер окна склаживания
    figsize=(20, 15) # общий размер сетки графиков
)
```



Данные графиков подтверждают, что в Европе не окупается один канал привлечения: AdNonSense. Причины неокупаемости, как и в случае с FaceBoom, скорее всего кроются в слабом росте LTV в течение первых двух недель, обусловленном плохим удержанием пользователей, привлеченных этим источником.

Остальные каналы в Европе окупаются за первые 7 дней.

Основные выводы на данном этапе:

- Наиболее неудачными рекламными каналами следует признать TipTop, FaceBoom и AdNonSense.
- При этом TipTop привлекает изрядную долю выручки, но регулярно возрастающие затраты на рекламу на данном канале мешают окупаемости данного канала.
- Каналы FaceBoom и AdNonSense имеют стабильную стоимость рекламы, однако выручка (LTV) с этих каналов растёт медленнее к концу второй недели. Возможно, пользователи, привлечённые данными каналами, **плохо удерживаются в приложении**.
- Остальные каналы имеют стабильно невысокую (или даже снижающуюся со временем) стоимость привлечения и окупаются в среднем в течение первой недели.

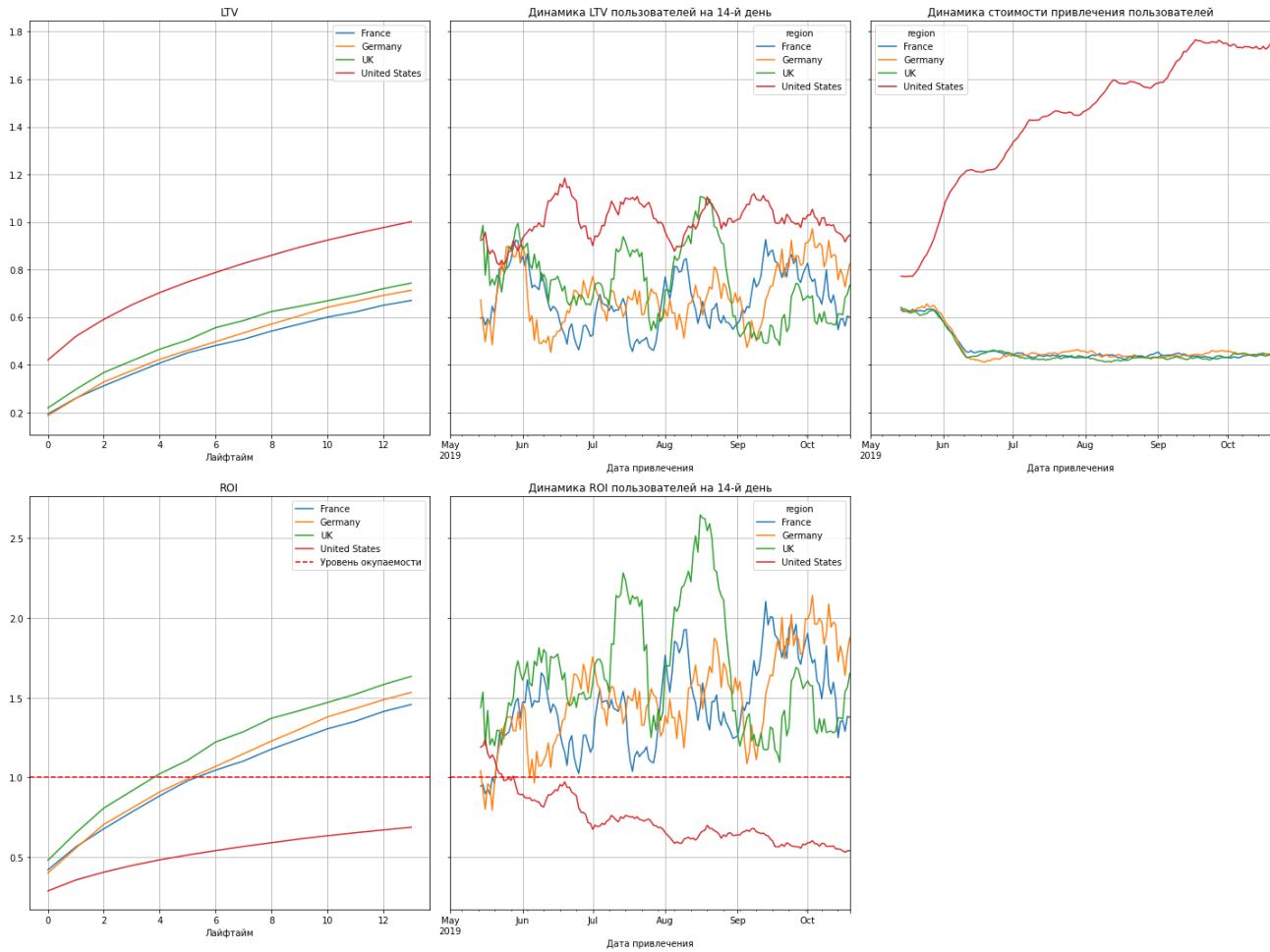
Анализ окупаемости рекламы по странам привлечения

Мы выявили три проблемных канала привлечения и описали возможные причины проблем с каждым из них. Проверим, во всех ли странах реклама окупается одинаково.

Воспользуемся функцией `get_ltv()` для расчёта LTV, CAC и ROI в разрезе региона привлечения в абсолютных значениях и в динамике, которые затем визуализируем с помощью функции `plot_ltv_roi()`:

```
In [46]: ltv_raw, ltv, ltv_din, roi, roi_din = get_ltv()
profile.query('channel != "organic"', # профили "неорганических" пользователей
orders, # данные о покупках
datetime(2019, 11, 1).date(), # момент анализа
14, # горизонт анализа
dimensions=['region'] # признаки когорт
)

plot_ltv_roi(
    ltv, # таблица пожизненной ценности
    ltv_din, # таблица динамики пожизненной ценности
    roi, # таблица возврата на инвестиции
    roi_din, # таблица динамики возврата на инвестиции
    14, # горизонт анализа
    window=14, # размер окна сглаживания
    figsize=(20, 15) # общий размер сетки графиков
)
```



Анализируя графики LTV, САС и ROI в разрезе регионов привлечения, можно видеть:

- для всех регионов наклон графиков LTV примерно одинаков - это, скорее всего, свидетельствует об **одинаковом среднем удержании пользователей-платильщиков в различных регионах**;
- при этом в динамике LTV 14-го дня все регионы в среднем стабильны - отсутствуют растущие и падающие тренды на протяжении всего периода наблюдений;
- регион США стабильно занимает лидирующую позицию в динамике LTV, однако ежемесячное увеличение затрат на рекламу в данном регионе (подозрительно напоминающее рост стоимости привлечения в канале TipTop) не позволяют ему выйти на окупаемость (на конец второй недели ROI составляет около 70%);
- схожесть поведения и последствий изменения САС для региона United States и канала TipTop позволяют предположить, что **при размещении рекламы в США по неясным причинам основная ставка была сделана именно на этот канал**;
- кроме того, бросается в глаза резкое перераспределение рекламных бюджетов в июне 2019 года, что может свидетельствовать об изменении ориентации рекламной политики в сторону рынка США;
- в большинстве регионов (за исключением США) рекламные затраты окупаются в течение первой недели и далее начинают приносить прибыль;
- это подтверждает и динамика ROI - для региона United States на протяжении всего периода наблюдений ROI 14-го демонстрирует устойчивый ниспадающий тренд, для остальных регионов - сезонно колеблется выше уровня окупаемости.

Основные выводы на данном этапе:

- Наиболее неудачными являются рекламные вложения в США.
- Динамика затрат на рекламу в США** подозрительно напоминает рост стоимости привлечения в канале TipTop и **не позволяет ему выйти на окупаемость**.
- Резкое перераспределение рекламных бюджетов в июне 2019 года может свидетельствовать об изменении ориентации рекламной политики в сторону рынка США.
- Остальные регионы имеют стабильно невысокую стоимость привлечения, которая окупается в среднем во второй половине первой недели, несмотря на наличие в них отдельных неокупаемых каналов.

Анализ окупаемости рекламы по устройствам

Ранее мы выявили три проблемных канала привлечения и описали возможные причины проблем с каждым из них. Кроме того удалось связать самые проблемные каналы с регионом США.

Проверим, для всех ли пользовательских устройств реклама окупается одинаково.

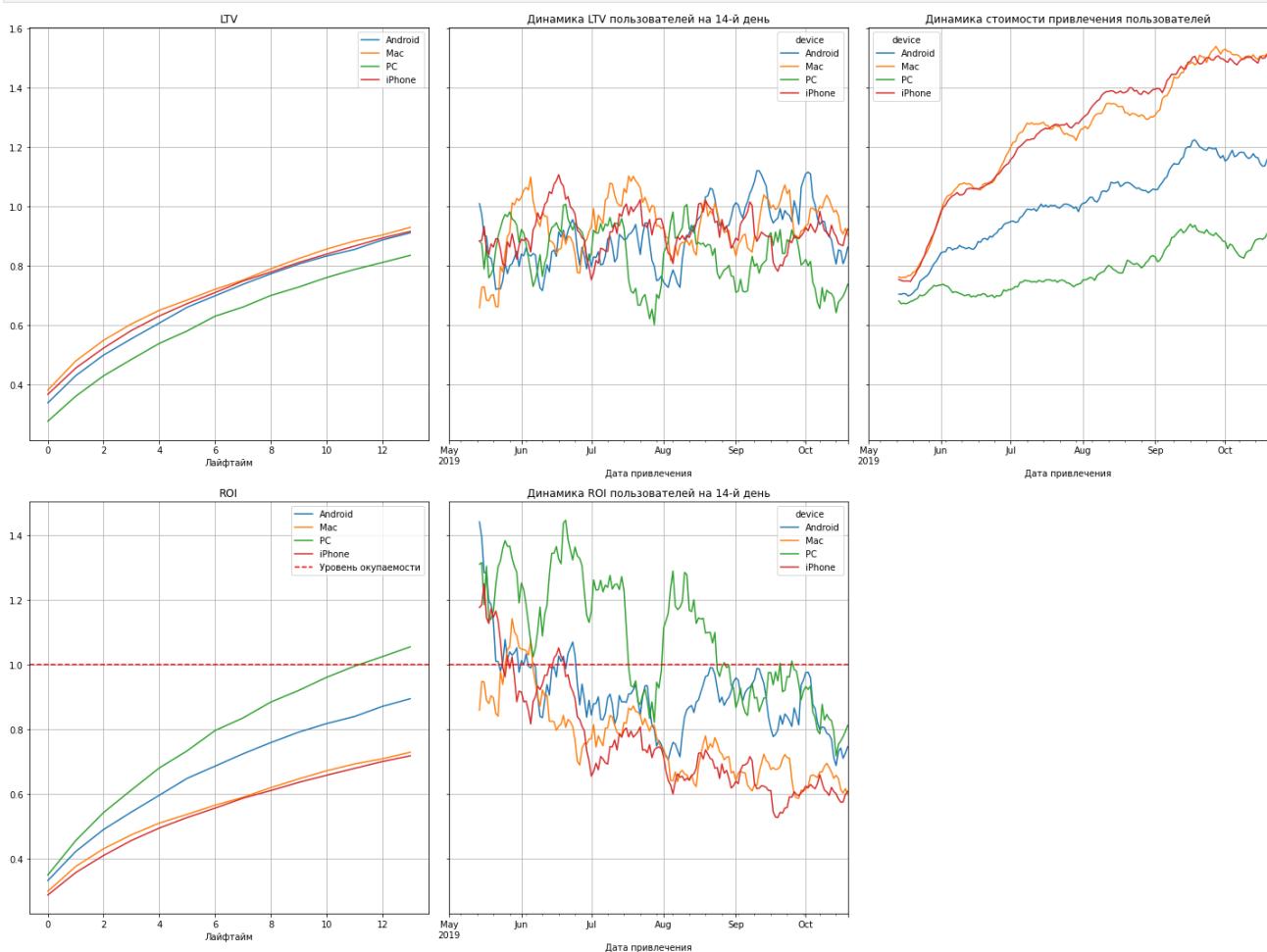
Воспользуемся функцией `get_ltv()` для расчёта LTV, САС и ROI в разрезе устройства в абсолютных значениях и в динамике, которые затем визуализируем с помощью функции `plot_ltv_roi()`:

```
In [47]: ltv_raw, ltv, ltv_din, roi, roi_din = get_ltv()
profile.query('channel != "organic"'), # профили "неорганических" пользователей
orders, # данные о покупках
datetime(2019, 11, 1).date(), # момент анализа
14, # горизонт анализа
dimensions=['device'] # признаки когорт
)
plot_ltv_roi(
    ltv, # таблица пожизненной ценности
    ltv_din, # таблица динамики пожизненной ценности
)
```

```

roi,          # таблица возврата на инвестиции
roi_din,      # таблица динамики возврата на инвестиции
14,          # горизонт анализа
window=14,    # размер окна сглаживания
figsize=(20, 15) # общий размер сетки графиков
)

```



Анализируя графики LTV, САС и ROI в разрезе пользовательских устройств, можно видеть:

- мобильные устройства и устройства на базе ОС Apple обеспечивают стабильно большую выручку по сравнению с PC;
- самая большая стоимость привлечения - на устройствах на базе ОС Apple, на втором месте - Android, дешевле всего обходится реклама на PC;
- единственный платформой, которая окупается в течение первых трёх недель, является PC;
- все устройства имеют отрицательный тренд динамики ROI 14 дня с течением времени, который легко увязывается с ростом затрат на рекламу при стабильной средней выручке 14-го дня.

Основные выводы на данном этапе:

- Изучение окупаемости рекламы в разрезе устройств не привносит в наше расследование новых находок - стоимость рекламы по всем устройствам возрастает, что при стабильном среднем LTV приводит к снижению ROI.
- Единственной окупаемой в течение первых двух недель платформой является PC в силу того, что реклама для них дорожает медленнее всего.

Изучение конверсии и удержания пользователей

Для подтверждения выдвинутых гипотез изучим поведение пользователей в разрезе каналов привлечения, регионов и устройств.

Поведение пользователей, как правило, описывается следующими бизнес-метриками:

- конверсия Conversion Rate, CR (доля пользователей, совершивших переход из одного состояния в другое, например "первое посещение - просмотр страницы товара");
- удержание Retention Rate, RR (доля пользователей, вернувшихся в приложение в последующие дни).

Для расчёта этих метрик используются следующие формулы:

$$CR_i = \frac{C_i}{CS},$$

где:

- C_i - количество пользователей, "сконвертировавшихся" в i -й день;
- CS - изначальный размер когорты.

$$RR_i = \frac{R_i}{CS},$$

где:

- R_i - количество пользователей, вернувшихся в i -й день;
- CS - изначальный размер когорты.

Как и раньше, считать CR и RR будем **без учёта "органических" пользователей**.

Момент анализа установим **1 ноября 2019 года**, горизонт анализа - **2 недели (14 дней)**.

Конверсия и удержание пользователей в разрезе рекламных каналов в США

В ходе анализа окупаемости рекламы в разрезе каналов привлечения мы отмечали, что выручка (LTV) с каналов FaceBoom и AdNonSense растёт медленнее к концу второй недели. Мы предположили, что пользователи, привлечённые данными каналами, плохо удерживаются в приложении. Проверим эту гипотезу, рассчитав конверсию и удержание в разрезе каналов.

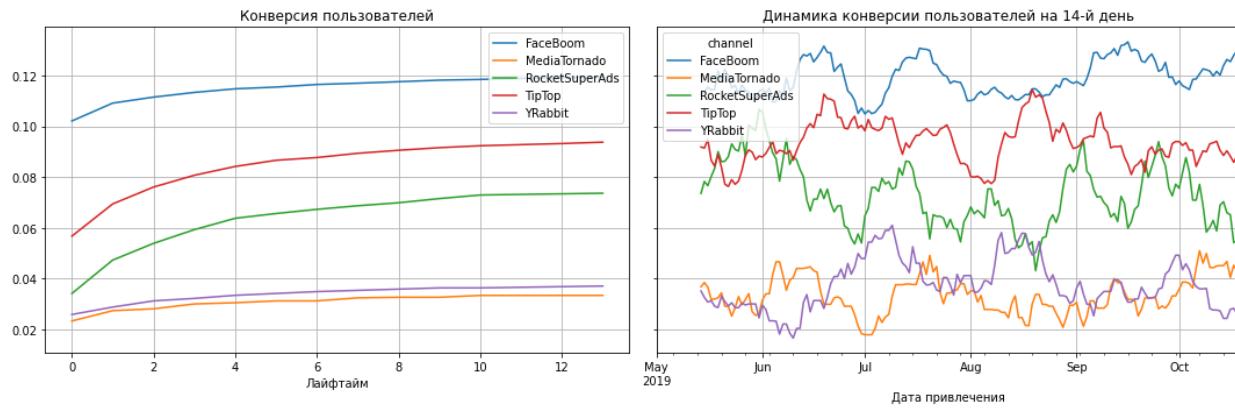
С учётом явного разделения каналов по региональному признаку будем отдельно рассматривать каналы в США и Европе.

Воспользуемся функцией `get_conversion()` для расчёта конверсии в разрезе рекламных каналов США в абсолютных значениях и в динамике, которые затем визуализируем с помощью функции `plot_conversion()`:

```
In [48]: # анализ конверсии по каналам в США
conv_raw, conv, conv_history = get_conversion(
    profile.query('channel in @usa_channels'), # профили "неорганических" пользователей в США
    orders, # данные о покупках
    datetime(2019, 11, 1).date(), # момент анализа
    14, # горизонт анализа
    dimensions=['channel']) # признаки когорт

)

plot_conversion(
    conv, # таблица конверсии
    conv_history, # таблица динамики конверсии
    14, # горизонт анализа
    window=14, # размер окна сглаживания
    figsize=(15, 5) # общий размер сетки графиков
)
```

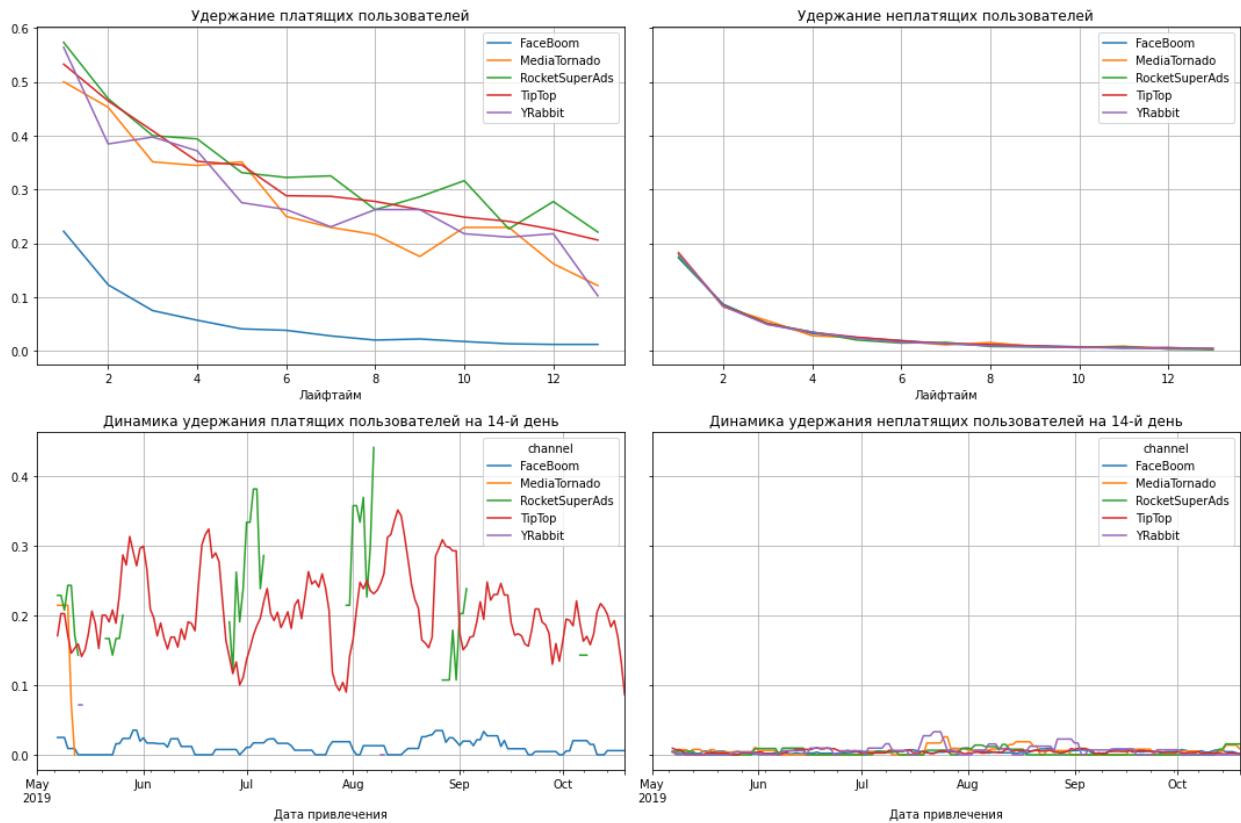


Воспользуемся функцией `get_retention()` для расчёта удержания в разрезе рекламных каналов в абсолютных значениях и в динамике, которые затем визуализируем с помощью функции `plot_retention()`:

```
In [49]: # анализ удержания по каналам в США
ret_raw, ret, ret_history = get_retention(
    profile.query('channel in @usa_channels'), # профили "неорганических" пользователей в США
    visits, # данные о сессиях
    datetime(2019, 11, 1).date(), # момент анализа
    14, # горизонт анализа
    dimensions=['channel']) # признаки когорт

)

plot_retention(
    ret, # таблица удержания
    ret_history, # таблица динамики удержания
    14, # горизонт анализа
    window=7, # размер окна сглаживания
    figsize=(15, 10) # общий размер сетки графиков
)
```



Графики конверсии и удержания свидетельствуют о том, что пользователи, привлеченные через канал FaceBoom лучше конвертируются в первые дни после привлечения, затем конверсия сильно замедляется. В целом конверсия с данного канала выше, чем с прочих.

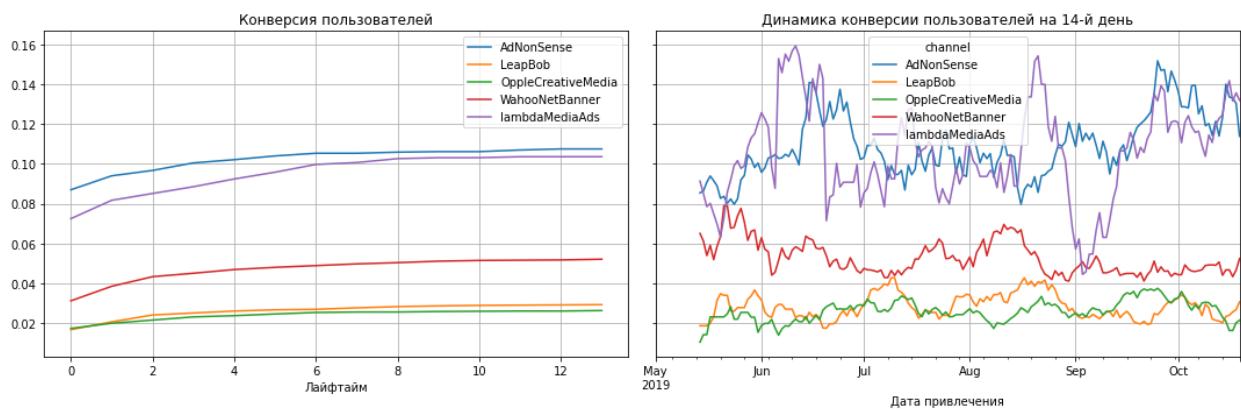
При этом удержание указанных пользователей к концу второй недели практически достигает нуля, причем для канала FaceBoom это наблюдалось на протяжении всего периода наблюдений.

Наиболее вероятной причиной такого положения дел представляется неправильный выбор целевой аудитории каналом FaceBoom: новые клиенты сперва активно интересуются приложением, но оно оказывается для них бесполезно, и к концу второй недели они перестают им пользоваться. Это же является причиной плохой окупаемости данного рекламного канала.

Конверсия и удержание пользователей в разрезе рекламных каналов в Европе

```
In [50]: # анализ конверсии по каналам в Европе
conv_raw, conv, conv_history = get_conversion(
    profile.query('channel in @europe_channels'), # профили "неорганических" пользователей в Европе
    orders, # данные о покупках
    datetime(2019, 11, 1).date(), # момент анализа
    14, # горизонт анализа
    dimensions=['channel'] # признаки когорт
)

plot_conversion(
    conv, # таблица конверсии
    conv_history, # таблица динамики конверсии
    14, # горизонт анализа
    window=14, # размер окна слаживания
    figsize=(15, 5) # общий размер сетки графиков
)
```



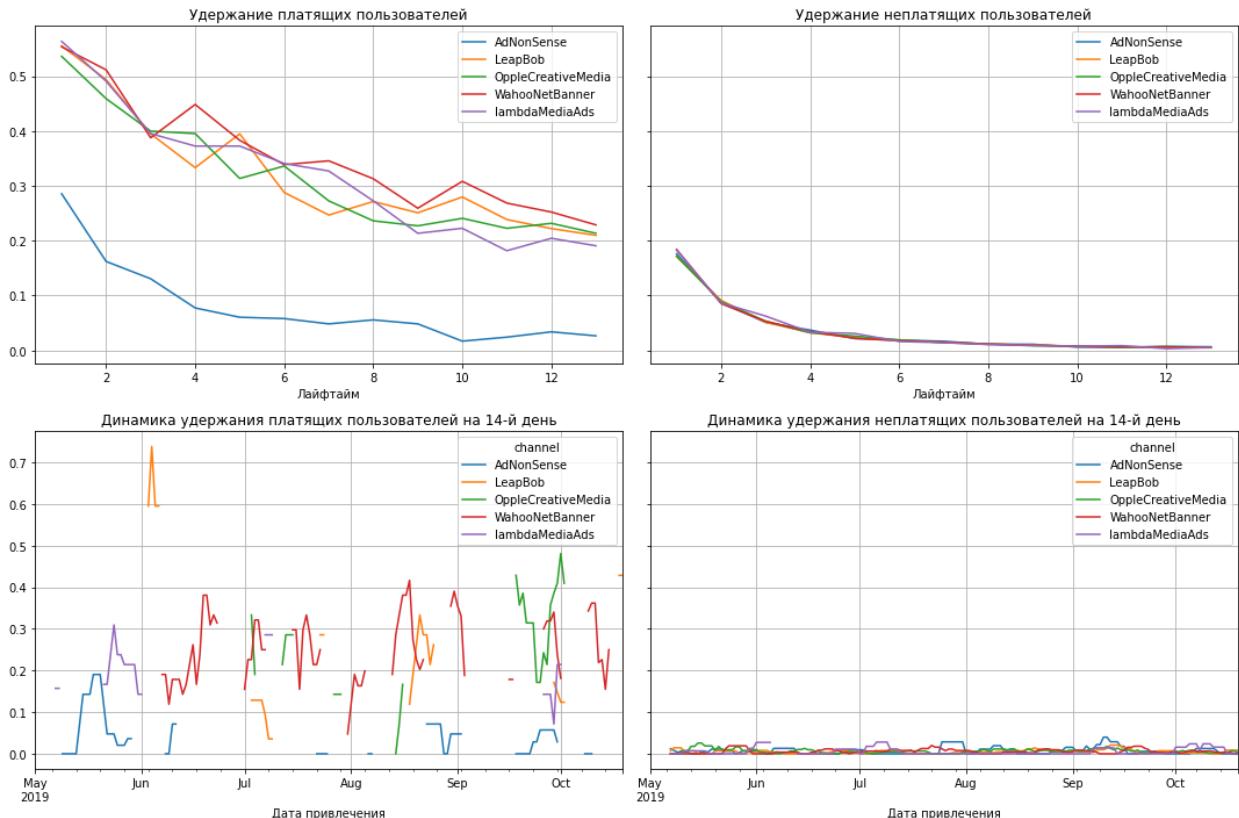
```
In [51]: # анализ удержания по каналам в Европе
ret_raw, ret, ret_history = get_retention(
    profile.query('channel in @europe_channels'), # профили "неорганических" пользователей в Европе
    visits, # данные о сессиях
    datetime(2019, 11, 1).date(), # момент анализа
    14, # горизонт анализа
    dimensions=['channel'] # признаки когорт
)

plot_retention(
    ret, # таблица удержания
    #
```

```

ret_history,          # таблица динамики удержания
14,                  # горизонт анализа
window=7,            # размер окна сглаживания
figsize=(15, 10)     # общий размер сетки графиков
)

```



Как следует из графиков конверсии и удержания каналов в Европе, ситуация с AdNonSense полностью повторяет FaceBoom, за исключением того, что для канала AdNonSense в мае удержание 14-го дня составляло среднестатистические 20%, а затем снизилось к нулю.

Наиболее вероятной причиной такого положения дел, как и в случае с FaceBoom, представляется неправильный выбор целевой аудитории.

Конверсия и удержание пользователей в разрезе регионов

Плохую окупаемость ещё одного рекламного канала TipTop мы ранее предварительно связали с плохой окупаемостью в регионе United States, предположив, что данный канал является в нём доминирующим. Исследуем поведение пользователей в разрезе регионов.

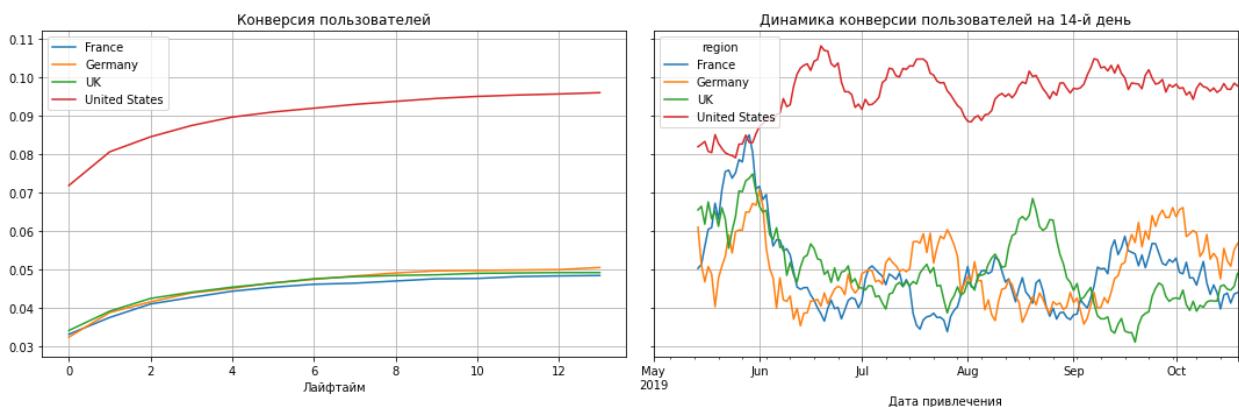
Воспользуемся функцией `get_conversion()` для расчёта конверсии в разрезе регионов в абсолютных значениях и в динамике, которые затем визуализируем с помощью функции `plot_conversion()`:

```

In [52]: conv_raw, conv, conv_history = get_conversion(
    profile.query("channel != \"organic\""), # профили "неорганических" пользователей
    orders,                                # данные о покупках
    datetime(2019, 11, 1).date(),           # момент анализа
    14,                                     # горизонт анализа
    dimensions=['region']                  # признаки когорт
)

plot_conversion(
    conv,                                  # таблица конверсии
    conv_history,                          # таблица динамики конверсии
    14,                                    # горизонт анализа
    window=14,                            # размер окна сглаживания
    figsize=(15, 5)                        # общий размер сетки графиков
)

```



Воспользуемся функцией `get_retention()` для расчёта удержания в разрезе регионов в абсолютных значениях и в динамике, которые затем визуализируем с помощью функции `plot_retention()`:

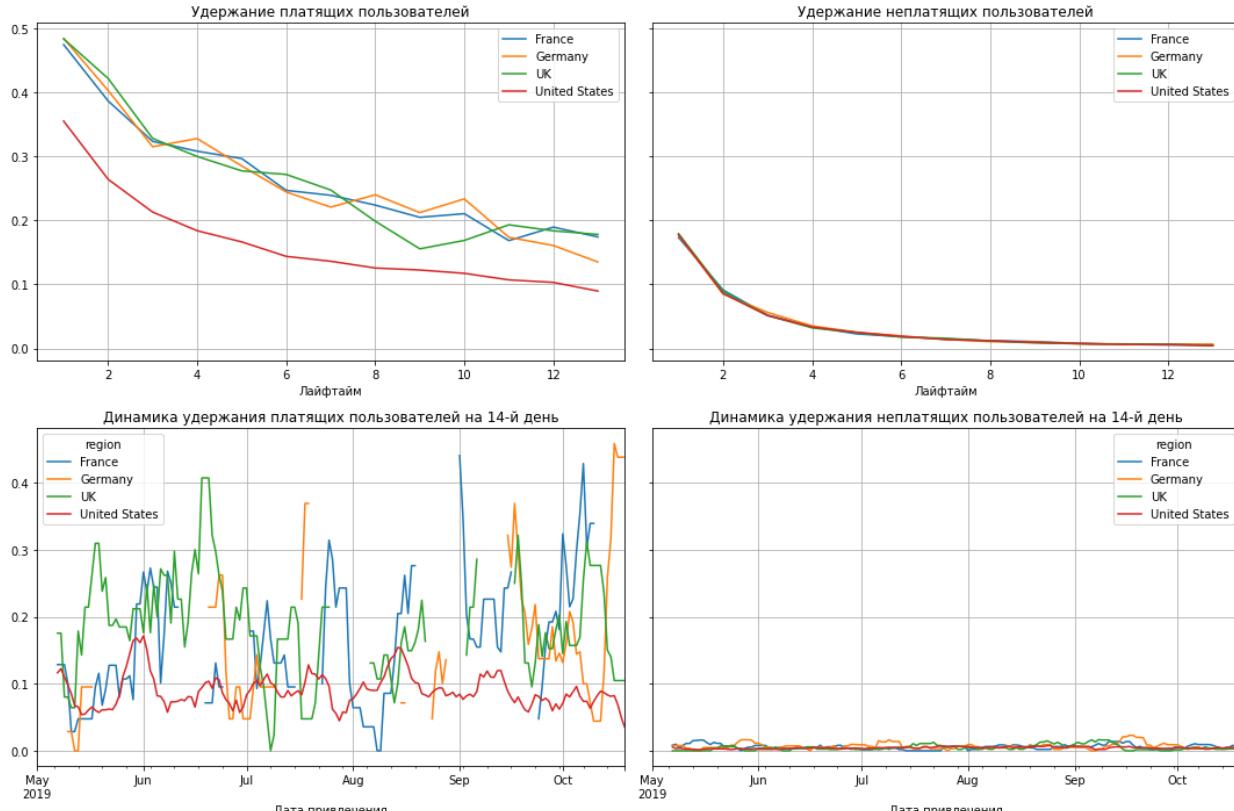
```
In [53]: ret_raw, ret, ret_history = get_retention()
```

```

profile.query('channel != "organic"'), # профили "неорганических" пользователей
visits,                                # данные о сессиях
datetime(2019,11,1).date(),             # момент анализа
14,                                     # горизонт анализа
dimensions=['region']                   # признаки когорт
)

plot_retention(
    ret,                                 # таблица удержания
    ret_history,                         # таблица динамики удержания
    14,                                   # горизонт анализа
    window=7,                            # размер окна сглаживания
    figsize=(15, 10)                     # общий размер сетки графиков
)

```



Графики конверсии и удержания свидетельствуют о том, что огромные вложения в рекламу TipTop и FaceBoom, с одной стороны, принесли плоды в виде опережающего роста конверсии (превышающей 95% на конец второй недели). При этом динамика конверсии свидетельствует, что рост произошёл одновременно с перераспределением рекламного бюджета в пользу TipTop.

Однако на удержание пользователей увеличение рекламного бюджета не оказалось никакого влияния: мало того, что удержание в США - самое низкое среди регионов, так ещё и динамика удержания 14-го дня показывает, что размер рекламных вложений не повлиял на желание пользователей оставаться с приложением.

Наиболее вероятной причиной такого положения дел представляется совокупность двух факторов: неправильное переориентирование привлечения на американских пользователей, помноженное на постоянное увеличение цены рекламных услуг канала TipTop.

Конверсия и удержание пользователей в разрезе устройств

При анализе окупаемости рекламы в разрезе устройств мы не обнаружили каких-либо существенных аномалий. Единственной окупаемой платформой оказалась PC, реклама для которой стоила дешевле всего.

Тем не менее, представляется целесообразным изучить поведение пользователей в разрезе устройств, поскольку в разных регионах предпочитают разные устройства, и если недостаточную окупаемость по устройствам Apple можно увязать с регионом (США), то для Android ситуация остаётся непонятной.

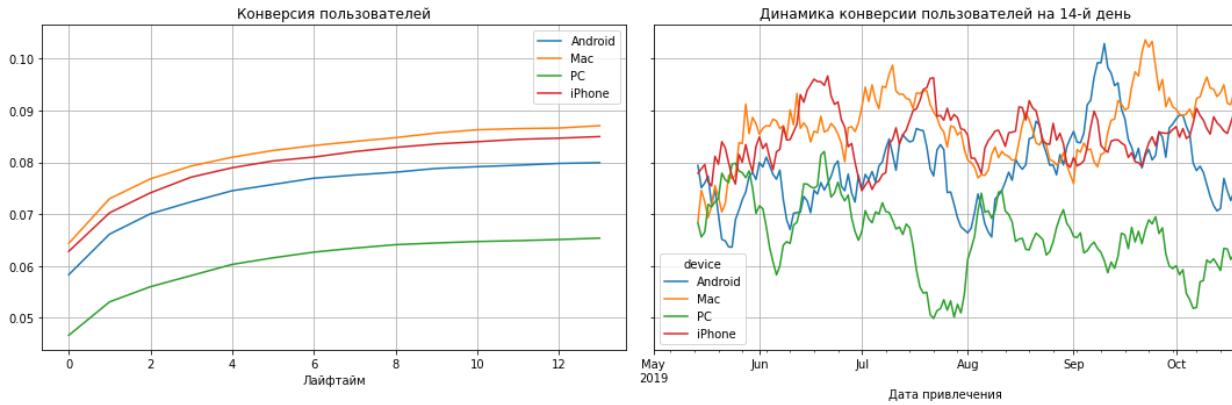
Рассчитаем и визуализируем конверсию:

```

In [54]: conv_raw, conv, conv_history = get_conversion()
profile.query('channel != "organic"'), # профили "неорганических" пользователей
orders,                                # данные о покупках
datetime(2019,11,1).date(),             # момент анализа
14,                                     # горизонт анализа
dimensions=['device']                   # признаки когорт
)

plot_conversion(
    conv,                                 # таблица конверсии
    conv_history,                         # таблица динамики конверсии
    14,                                   # горизонт анализа
    window=14,                            # размер окна сглаживания
    figsize=(15, 5)                      # общий размер сетки графиков
)

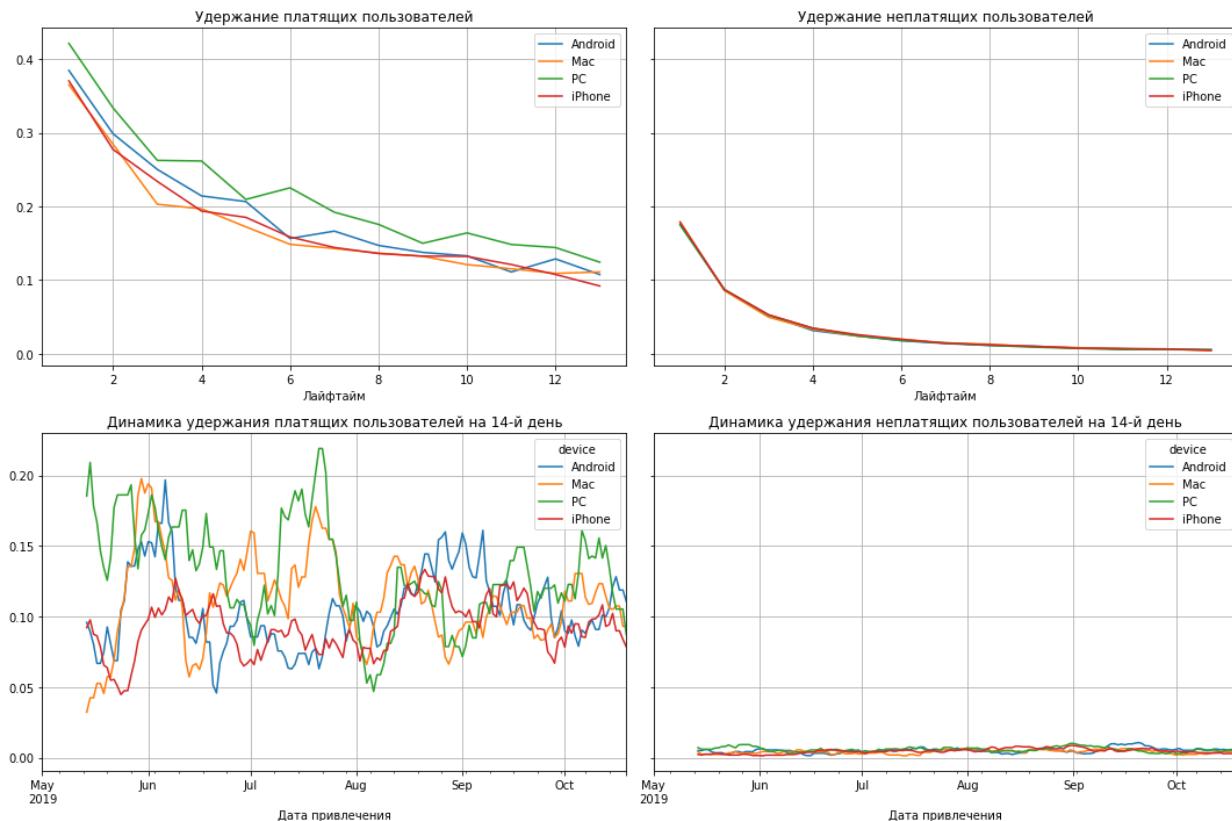
```



Рассчитаем и визуализируем удержание:

```
In [55]: ret_raw, ret, ret_history = get_retention(
    profile.query('channel != "organic"'), # профили "неорганических" пользователей
    visits, # данные о сессиях
    datetime(2019, 11, 1).date(), # момент анализа
    14, # горизонт анализа
    dimensions=['device'] # признаки когорт
)

plot_retention(
    ret, # таблица удержания
    ret_history, # таблица динамики удержания
    14, # горизонт анализа
    window=14, # размер окна скользящего
    figsize=(15, 10) # общий размер сетки графиков
)
```



Мы видим, что конверсия для мобильных устройств и компьютеров Apple, а также устройств Android, несколько выше PC. В то же время, удержание несколько выше для PC - пользователи Android, а за ними и Apple быстрее прекращают заходить в приложение.

Вероятно, это объясняется большим консерватизмом пользователей PC по сравнению с пользователями других платформ, либо различием функционала приложения под каждую из платформ. Целесообразно обсудить это с отделом разработки. Возможно удастся не только повысить привлекательность, но и полезность приложения для пользователей платформ Apple и Android.

Выводы

- В процессе анализа окупаемости рекламы и поведения пользователей в различных разрезах нами были получены следующие выводы:
 - В целом, средства, выделяемые на рекламу, не окупаются (общая окупаемость инвестиций на границу второй недели находится на уровне 80%).
 - В плохой окупаемости виноваты три канала привлечения:
 - TipTop и FaceBoom - в США;
 - AdNonSense - в Европе.
 - Перераспределение в июне 2019 года рекламного бюджета в пользу канала TipTop обеспечило высокую конверсию пользователей в США, однако не смогло обеспечить должное удержание, что, в совокупности с регулярным повышением цен на рекламу TipTop на американском рынке привело к самой низкой окупаемости рекламы в данном канале по сравнению с остальными.

- Недостаточная окупаемость рекламы в каналах FaceBoom и AdNonSense обусловлена низким удержанием пользователей (вплоть до 0 на конец второй недели) - следствием неправильно выбранной данными каналами целевой аудитории пользователей.
 - Пользователи РС демонстрируют меньшую конверсию при большем удержании по сравнению с платформами Apple и Android. Причины этого могут скрываться как в консерватизме пользователей РС, так и в различиях функционала приложения под различные платформы.
 - При этом, РС - единственная окупаемая платформа.
1. На основе полученных выводов в общем выводе исследования будут сформулированы рекомендации как рекламному отделу компании, так и отделу разработки.
-

Общий вывод исследования

Проведённое исследование посвящено выяснению причин убытков компании-разработчика, несмотря на огромные вложения в рекламу приложения, и включает в себя:

- задание функций расчёта и визуализации бизнес-метрик;
- обзор данных;
- предварительную обработку данных;
- исследовательский анализ данных (изучение составленных профилей пользователей приложения в различных регионах);
- исследование распределения маркетингового бюджета;
- оценку окупаемости рекламы в различных разрезах.

Основные выводы по каждому этапу приведены в соответствующих разделах.

Основываясь на них можно сформулировать следующий **ответ на основной вопрос исследования**:

- реклама в целом за целевой период (2 недели) окупается на 80%;
- виновниками недостаточной окупаемости являются три рекламных канала - TipTop, FaceBoom и AdNonSense;
- основной причиной недостаточной окупаемости рекламы в TipTop (свыше 50% общего рекламного бюджета) является регулярный опережающий рост её стоимости;
- недостаточная окупаемость рекламы в каналах FaceBoom (стабильно порядка 30% бюджета) и AdNonSense обусловлена низким удержанием пользователей (вплоть до 0 на конец второй недели) - следствием, вероятно, неправильно выбранной данными каналами целевой аудитории пользователей;
- в разрезе устройств наблюдается недостаточное удержание пользователей платформ Mac, iPhone и Android.

Рекомендации для маркетингового отдела:

- пересмотреть стратегию работы с американским рынком:
 - постепенный отказ от сотрудничества с каналом TipTop в пользу других каналов региона позволит существенно поднять в нём общую окупаемость рекламы:
 - полностью прекращать сотрудничество с огромной рекламной сетью TipTop не целесообразно - она привлекает достаточно большое количество платящих пользователей и демонстрирует наилучшую двухнедельную динамику выручки (LTV);
 - тем не менее, представляется целесообразным до половины текущего рекламного бюджета TipTop перераспределить в соотношении 40% x 30% x 30% в пользу сетей RocketSuperAds, YRabbit и MediaTornado соответственно - все три сети демонстрируют хорошую окупаемость, при этом RocketSuperAds по двухнедельной динамике конверсии и удержанию (качеству) пользователей ближе всего к секвестрируемой TipTop;
 - часть (до 20%, к примеру) освободившихся средств может быть целесообразно направить на рекламу в европейских каналах с целью увеличения притока клиентов в этом регионе:
 - реклама в Европе стоит существенно дешевле (и даже дешевеет);
 - подавляющее большинство европейских рекламных каналов окупаются в течение первой недели;
- перемотреть совместно с представителями каналов FaceBoom и AdNonSense продвигаемые ими рекламные стратегии, в частности, касательно определения целевой аудитории приложения, что позволит увеличить удержание клиентов по этим каналам и, как следствие, их окупаемость:
 - забирать деньги у FaceBoom не целесообразно, поскольку сеть является де-факто лидером по привлечению американских пользователей, а значит делает качественный рекламный продукт (к тому же данная сеть характеризуется стабильной ценовой политикой);
 - понимание причин оттока плательщиков, привлечённых FaceBoom, на имеющихся в наличии данных невозможно - необходим дополнительный сравнительный анализ целевой аудитории каналов FaceBoom, RocketSuperAds, YRabbit и MediaTornado;
 - для понимания причин плохого удержания в канале AdNonSense представляется целесообразным сравнить таргетирование его рекламы с каналом lambdaMediaAds - компании очень близки по качеству рекламного продукта (близость конверсии) и, кроме того, lambdaMediaAds демонстрирует не только достаточно хорошее удержание плательщиков, но и лидирует по двухнедельной динамике валовой выручки (LTV).

Рекомендации для отдела разработки: более детально изучить причины недостаточного удержания пользователей устройств Mac, iPhone и Android, а также принять меры к исправлению ситуации, что потенциально позволит увеличить популярность у них приложения и общую окупаемость инвестиций.

Вниманию инженеров по данным: при составлении описания данных целесообразно указывать валюту, в которой выражены выручка и рекламные расходы.