

Author: Denis Ershov

Case Study: Yammer Analytics Research

Source: [Mode SQL Tutorial](#)

Main Problems:

- 1) Investigation of a drop in the user engagement
- 2) Digging into the search functionality to revamp it
- 3) A/B tests for the release of new features

Short background:

Yammer is a social network for communication with coworkers. Individuals can share documents, updates, ideas and questions by posting them in groups. Yammer is used often in larger corporations at which coworkers are scattered across the world. The platform facilitates the knowledge sharing and staying up-to-date in an area of interest.

Although Yammer has definitely a team of data analysts to investigate various aspects and issues, the data is confidential. Therefore, the simulated, close to real data is used within this case study.

The database for this case study is stored at Mode platform, the link is above. Three most important tables used for all three problems:

- 1) Users. It includes one row per user with some descriptive information about that user's account, e.g., company_id, language, state, etc.
- 2) Events. For each activity a user does, a new row in this table is created saving when the event happened, what was the type and name of the activity, location, device, etc.
- 3) E-Mail Events. This table summarizes all events which are related to E-Mails which users obtain from Yammer to their professional E-Mail addresses. This tracks if the user opens an E-Mail, clicks the link inside of it.

All tables are related to each other in a way that each user id is present in all three tables. For the third problem there are additional tables which are relevant to deal with the given questions.

Problem 1. Investigation of a drop in the user engagement

I come to the office on September 2, 2014 and the head of the Yammer product team states that he noticed a weird activity in the user behaviour between July and August 2014. The trend for the weekly active unique users has the following pattern (see Figure 1).

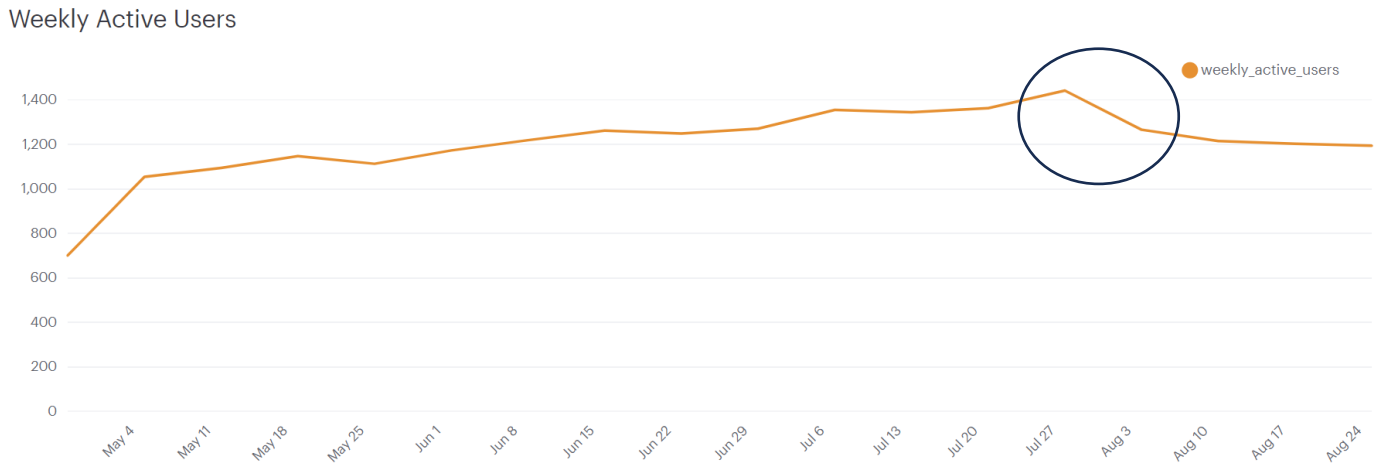


Figure 1. Weekly active users from the beginning of May till the end of August 2014

He asks me to investigate what could be the reasons behind this trend and what ideas and suggestions I could propose to improve the situation.

Hypotheses:

1. Weekly number of active users dropped as there was an issue of activation for new users.
2. August is usually a high holiday season in many countries, therefore, the number of active users dropped.
3. Retention of old users is getting low, while the new users are not too active.
4. Issues with some types of devices and users cannot fully use the service.
5. Issues with the Yammer distribution E-Mails, individuals cannot open them to get to Yammer.

Hypothesis 1.

The hypothesis states that the number of active users dropped as there was some issues with activation for new users. For this task I can work simply with the table that contains all users. This table has an activation date column (if activation is done – user is activated, if the cell is empty, the user is not active yet). I will use the creation date, count of all created users and activated users from this table to try to get some pattern.

Query:

```
SELECT DATE_TRUNC('day',created_at) AS day,  
COUNT(*) AS all_users,  
COUNT(CASE WHEN activated_at IS NOT NULL THEN u.user_id ELSE NULL END) AS activated_users  
FROM tutorial.yammer_users u  
WHERE created_at >= '2014-06-15'  
AND created_at < '2014-09-01'  
GROUP BY 1  
ORDER BY 1
```

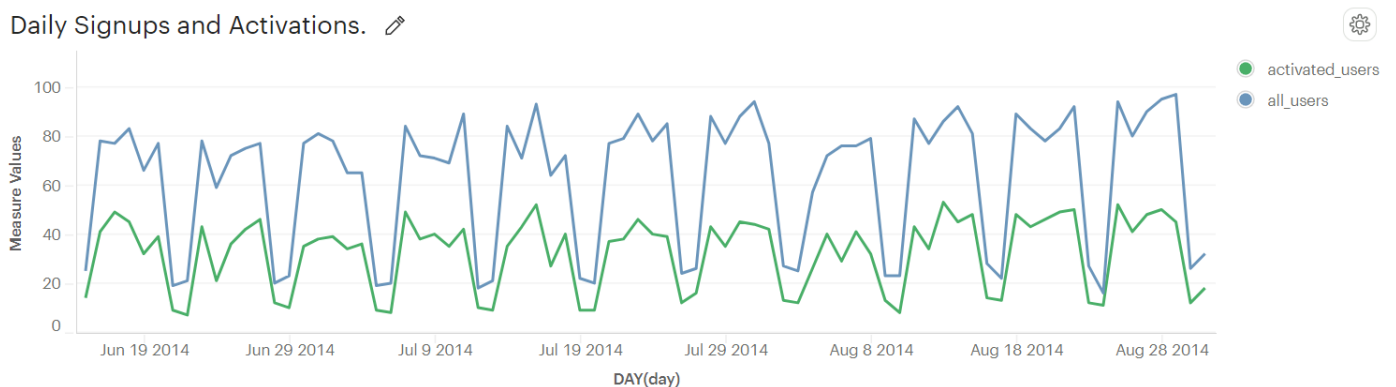


Figure 2. Daily Signups and Activation from middle of June to end of August 2014.

There is no any visible trend among the number of signed up and activated users every day. During the week, there are more newly created and activated users, during the weekend this number drops significantly. This pattern repeats within all the time frame before and after the user engagement drop. So, this hypothesis is false. There is definitely no issue in activation for new users.

Hypothesis 2.

This hypothesis states that the weekly number of users dropped in August much, as it was the holiday season in some countries. This is usually a case for most European and North American countries, while less relevant for Asia and other countries.

For checking this hypothesis, I need to group all users based on their country of activity and then check the activity within July-August 2014 for all groups. As I have already checked in the first hypothesis that there was no issue with account activation and usage, I will consider here only activated users and their activities (engagement). I would like to see only the countries where the number of events in August is by more than 15% less as compared to July.

Query:

```

SELECT * FROM
(SELECT DISTINCT(location),
COUNT(CASE WHEN DATE_TRUNC('month', occurred_at) = '2014-07-01' THEN 1 ELSE NULL END) AS events_july,
COUNT(CASE WHEN DATE_TRUNC('month', occurred_at) = '2014-08-01' THEN 1 ELSE NULL END) AS
events_august,
(COUNT(CASE WHEN DATE_TRUNC('month', occurred_at) = '2014-08-01' THEN 1 ELSE NULL END)::FLOAT /
COUNT(CASE WHEN DATE_TRUNC('month', occurred_at) = '2014-07-01' THEN 1 ELSE NULL END)::FLOAT) AS
difference
FROM tutorial.yammer_events
WHERE occurred_at BETWEEN '2014-07-01' AND '2014-08-31' AND event_type = 'engagement'
GROUP BY 1) sub_table
WHERE difference < 0.85

```

As seen below in Figure 3, there are major countries belonging to the North America (Canada, Mexico, USA), countries belonging to Europe (Austria, Belgium, France, Germany, Greece, Italy, Norway, UK, Sweden), countries belonging to the South America (Brazil, Chile, Venezuela) and the ones belonging to Asia (India, Indonesia, Korea, Saudi Arabia, Singapore, Taiwan). All of them have different holiday seasons, but even though the significant drop of more than 15% in user activities happened in all of them.

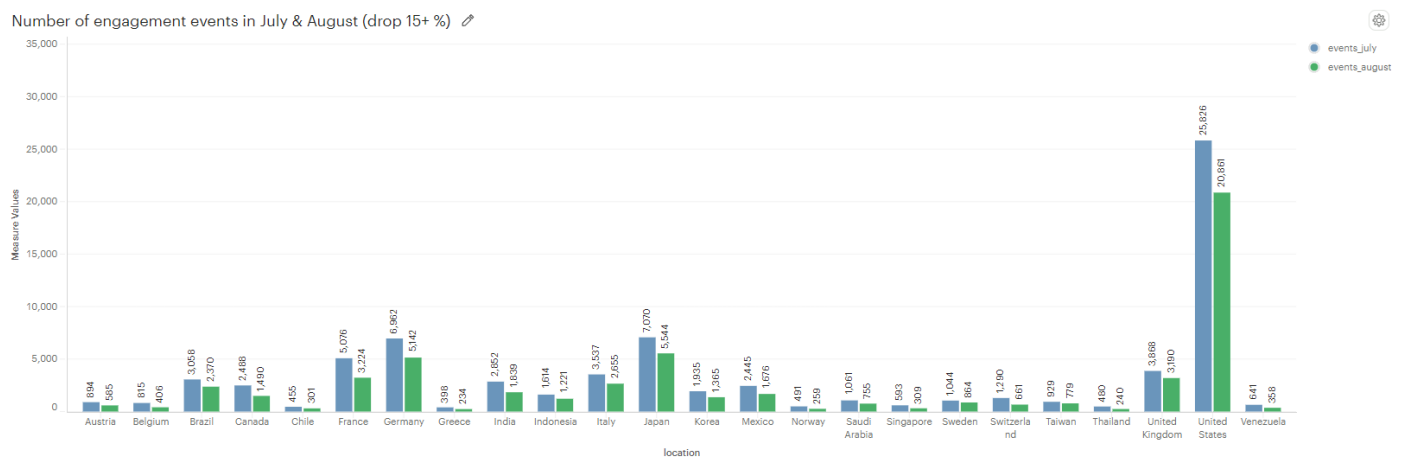


Figure 3. Number of engagement events in July and August for countries which have the drop of 15+ %

Let's now check the countries which had an increase in the user engagement by at least 10% in August as compared to July.

Query:

```

SELECT * FROM
(SELECT DISTINCT(location),
COUNT(CASE WHEN DATE_TRUNC('month', occurred_at) = '2014-07-01' THEN 1 ELSE NULL END) AS events_july,
COUNT(CASE WHEN DATE_TRUNC('month', occurred_at) = '2014-08-01' THEN 1 ELSE NULL END) AS
events_august,
(COUNT(CASE WHEN DATE_TRUNC('month', occurred_at) = '2014-08-01' THEN 1 ELSE NULL END)::FLOAT /
COUNT(CASE WHEN DATE_TRUNC('month', occurred_at) = '2014-07-01' THEN 1 ELSE NULL END)::FLOAT) AS
difference
FROM tutorial.yammer_events
WHERE occurred_at BETWEEN '2014-07-01' AND '2014-08-31' AND event_type = 'engagement'
GROUP BY 1) sub_table
WHERE difference > 1.10

```

Number of engagement events in July & August (increase 10+ %) [🔗](#)

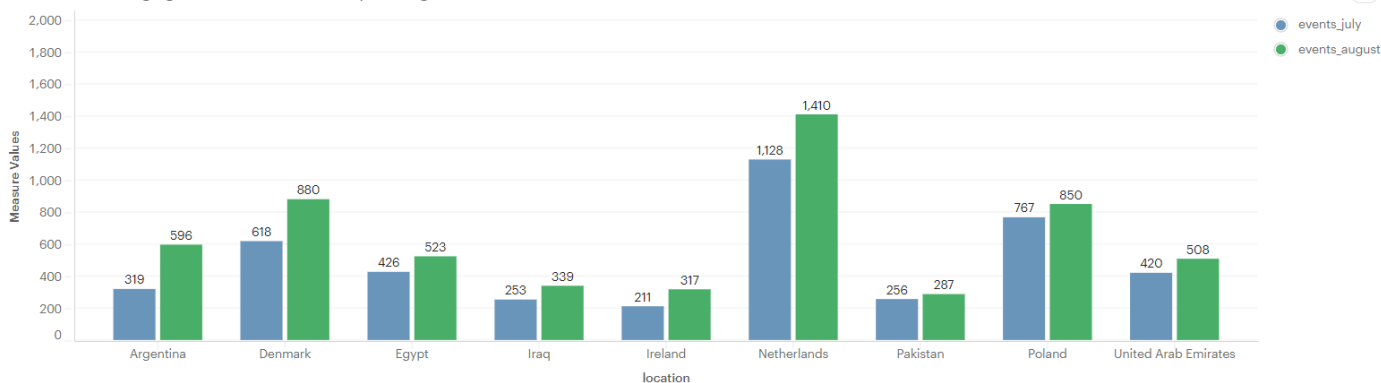


Figure 4. Number of engagement events in July and August for countries which have the increase of 10+ %

As seen from these results, there are several countries from Europe, South America, Asia which had a considerable growth in the number of events by 10+ % in August as compared to July. These two figures prove that there is no correlation between the drop of engagement and the usual holiday season in August in certain countries.

Hypothesis 3.

This hypothesis states that there are issues with the retention of old users (e.g., 2+ months since activation in Yammer) and the new users are not much interested in using the service.

In order to prove this, I can check how the number of users was changing since beginning of June until the end of August. In this case I will consider only "login" activities and I will split users based on their age (time passed between the login event and the registration at the platform) into 6 categories. For each week I will check how many users from each of the age groups logged in the platform. Even if a user logged in multiple times, only one login session is considered, while the age of this use is averaged to assign him/her to a certain age group.

Query:

```
SELECT DATE_TRUNC('week',z.occurred_at) AS "week",
AVG(z.age_at_event) AS "Average age during week",

-- Creating separate columns for the users based on how long they are registered at the platform
COUNT(DISTINCT CASE WHEN z.age_at_event > 70 THEN z.user_id ELSE NULL END) AS "> 10 weeks",
COUNT(DISTINCT CASE WHEN z.age_at_event <= 70 AND z.age_at_event > 56 THEN z.user_id ELSE NULL END)
AS "9-10 weeks",
COUNT(DISTINCT CASE WHEN z.age_at_event <= 56 AND z.age_at_event > 42 THEN z.user_id ELSE NULL END)
AS "7-8 weeks",
COUNT(DISTINCT CASE WHEN z.age_at_event <= 42 AND z.age_at_event > 28 THEN z.user_id ELSE NULL END)
AS "5-6 weeks",
COUNT(DISTINCT CASE WHEN z.age_at_event <= 28 AND z.age_at_event > 14 THEN z.user_id ELSE NULL END)
AS "3-4 weeks",
COUNT(DISTINCT CASE WHEN z.age_at_event <= 14 AND z.age_at_event > 0 THEN z.user_id ELSE NULL END)
AS "1-2 weeks"
FROM (SELECT e.occurred_at,
u.user_id,
DATE_TRUNC('week',u.activated_at) AS activation_week,
```

-- Calculating the age of a user when an event happened

```
EXTRACT('day' FROM e.occurred_at - u.activated_at) AS age_at_event
```

-- Joining tables with users and events

```
FROM tutorial.yammer_users u
```

```
JOIN tutorial.yammer_events e
```

```
ON e.user_id = u.user_id
```

```
AND e.event_type = 'engagement'
```

```
AND e.event_name = 'login'
```

```
AND e.occurred_at >= '2014-06-02'
```

```
AND e.occurred_at < '2014-09-01'
```

```
WHERE u.activated_at IS NOT NULL) z
```

GROUP BY 1

ORDER BY 1

The results in Figure 5 demonstrate that the number of users who are older than 10 weeks at the platform was increasing steadily from June 1 till July 27, what makes sense, as there are more and more users at the platform and the amount of those falling into category “>10 weeks” was increasing all the time, too. In August this number started drastically decreasing, although this is not related to the worse retention of the older users. This is proven by the fact that the drop in this period happened also for other age groups, e.g., the number of new users who logged in dropped from 328 in the end of July to 295-298 in the first half of August. The same holds true for all other age groups, for some to a greater extent, for others to a lesser extent.

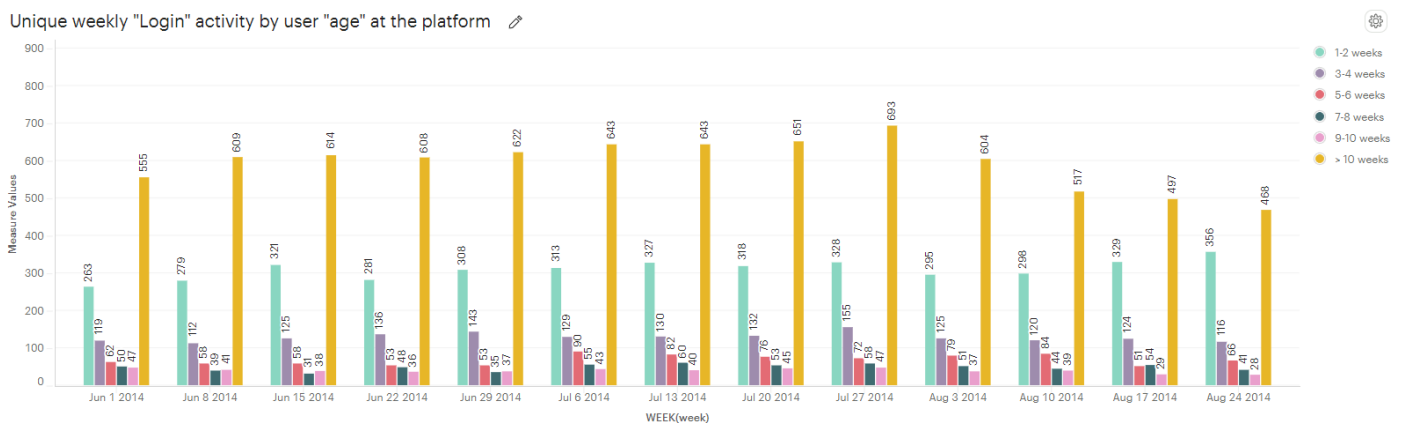


Figure 5. Unique weekly “Login” activity by user “age” at the platform between June and August 2014

Based on these results, I prone to reject the third hypothesis. User retention has nothing to do with the drop of unique users at Yammer in August 2014.

Hypothesis 4.

I have seen in the previous analysis that there were definitely no issues caused by the summer holiday season or the low retention of older users. Therefore, the lower user engagement (i.e., the number of unique logging sessions or in other words the number of unique users) may be related to some technical issues. This often happens when some updates were introduced, it may turn out that some devices started working

better after the updates and the app only improved, but there are also multiple cases when the updates cause only some destruction to certain users.

Let's check first what unique names of devices exist among users.

Query:

```
SELECT DISTINCT device FROM tutorial.yammer_events
```

The obtained list has the following 26 values which I split manually in 3 categories:

1) Laptops/Computers: dell inspiron desktop, macbook pro, asus chromebook, macbook air, lenovo thinkpad, mac mini, acer aspire desktop, acer aspire notebook, dell inspiron notebook, hp pavilion desktop

2) Mobile Devices: amazon fire phone, iphone 5, iphone 5s, nexus 5, htc one, iphone 4s, samsung galaxy note, nokia lumia 635, samsung galaxy s4

3) Tablets: nexus 10, ipad mini, samsung galaxy tablet, nexus 7, kindle fire, ipad air, windows surface

Let's now see how the number of unique users changed for each device type in the summer months (from June until the end of August). Again, only the "engagement" event type is considered (users that are not activated yet – out of interest), more specifically only the "login" event type is in focus, as I would like to count the number of unique users in the portal and the "login" is the first action that any user does when opening Yammer, all other actions are out of scope.

Query:

```
SELECT DATE_TRUNC('week', occurred_at) AS week,
```

```
-- Count total weekly users, computer/laptop users, smartphone users, tablet users
```

```
COUNT(DISTINCT e.user_id) AS total_weekly_users,
```

```
COUNT(DISTINCT CASE WHEN e.device IN ('macbook pro','lenovo thinkpad','macbook air','dell inspiron notebook',  
'asus chromebook','dell inspiron desktop','acer aspire notebook','hp pavilion desktop','acer aspire desktop','mac mini')  
THEN e.user_id ELSE NULL END) AS computer_users,
```

```
COUNT(DISTINCT CASE WHEN e.device IN ('iphone 5','samsung galaxy s4','nexus 5','iphone 5s','iphone 4s','nokia  
lumia 635', 'htc one','samsung galaxy note','amazon fire phone') THEN e.user_id ELSE NULL END) AS phone_users,
```

```
COUNT(DISTINCT CASE WHEN e.device IN ('ipad air','nexus 7','ipad mini','nexus 10','kindle fire','windows surface',  
'samsung galaxy tablet') THEN e.user_id ELSE NULL END) AS tablet_users
```

```
FROM tutorial.yammer_events e
```

```
WHERE e.event_type = 'engagement' AND e.event_name = 'login'
```

```
GROUP BY 1
```

```
ORDER BY 1
```

The results in Figure 6 demonstrate quite unusual trends. I can note that the number of unique computer/laptop users stayed about 900 during the whole month of July and increased a bit by the last week and then stayed close to 900 with only a slight drop. This means that the trend is stable and no major changes are seen for computer users. However, if I look at the number of unique mobile device users dropped significantly in the end of July from about 600 to about 430 by the middle of August, the same holds true for tablet users (drop from 250 to about 160). In both cases the drop is not slight and random, the drop is continuous, the trends stay stable during the whole month of August. It may mean that there were some issues with phone and tablet app. Users had difficulties using it, e.g., the app did not run stable and froze or kicked the user out, therefore users did not try to log in anymore, the number dropped and was stable low as compared to July.

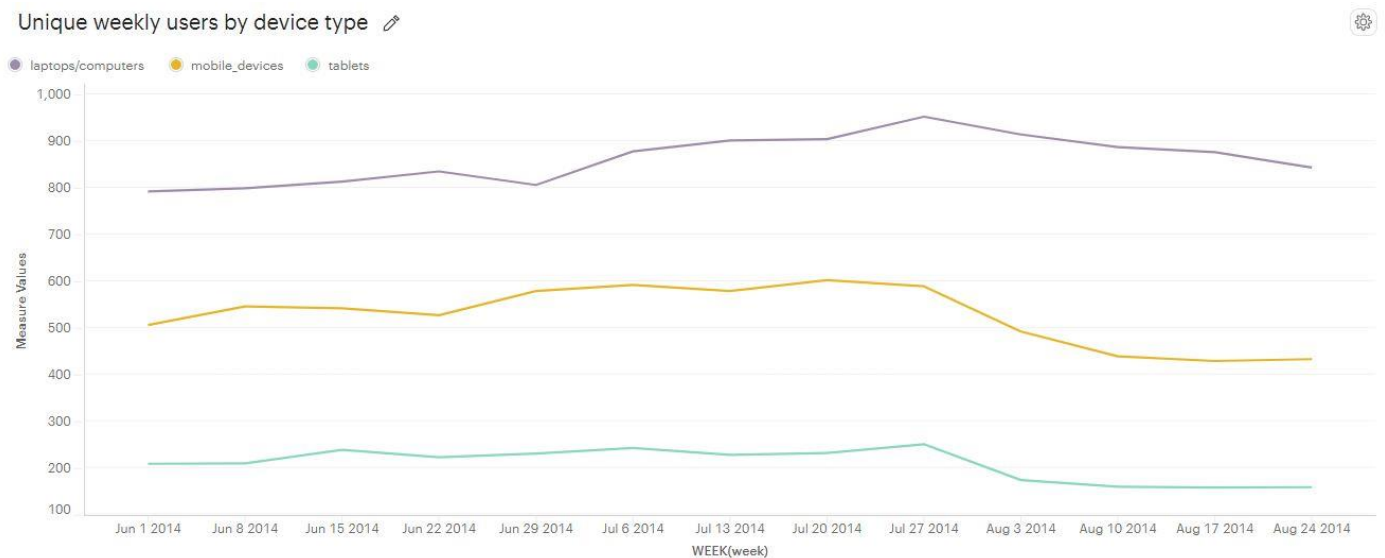


Figure 6. Unique weekly users by device type

Digging a bit deeper and analysing only Apple users, the mentioned above trends become even more obvious. As seen in Figure 7, for Apple macbook users there are no significant drops in user engagement, however, there is a notable reduction of logins among Apple iPhone users from 300 in July to close to 200 in August. The same is true for Apple iPad users with the drop from 90-100 in July to 65-70 users in August. However, the user drop was more (than only this cause by Apple users) for all device types meaning that the drops are observed in other non-Apple devices.

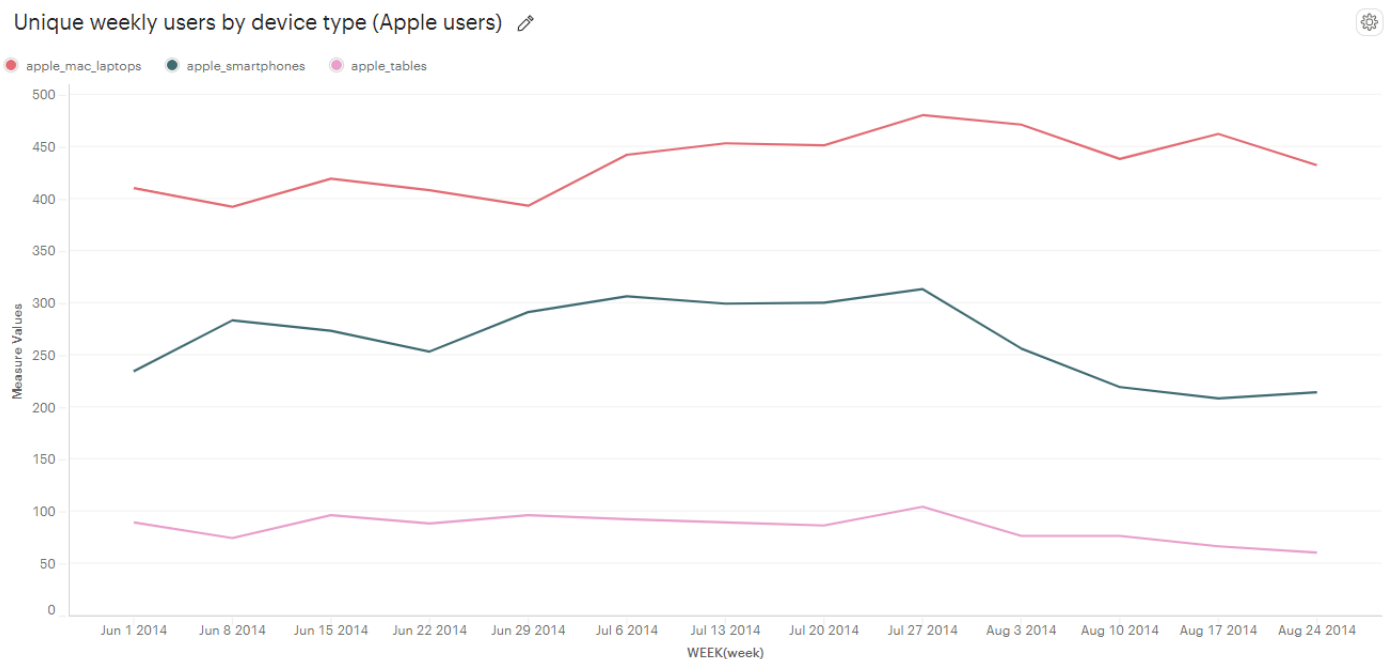


Figure 7. Unique weekly users by device type (Apple users)

This brings me to a thought that I should communicate to the product manager that the drop is mainly observed in mobile and tablet devices and he/she should check what could cause this in the August. Probably there was some update in the app, causing a malfunction. I would also recommend to check the App Store

and Google Play Store to review the comments of users on Yammer, they may mention what sort of problem they experience, this may help to tackle the problem faster and without digging into the code much.

Hypothesis 5.

The issues with the app for portable devices may be not the only one that caused a significant reduction in the number of users. As I have extra data about the E-Mails that are sent automatically by Yammer to the users, I can try to extract some useful information from there. My hypothesis is that there were some issues with these E-Mails and users simply cannot open the link in those E-Mails to get to Yammer. Quite often users are engaged to the platform thanks to such E-Mails as by receiving them, the users get back to the platform and stay active there by reading, liking, posting or even commenting something, looking for some information and many more.

Let's first check if the digest mails (weekly distribution E-Mails from Yammer) are sent properly and then read by users, as well as if the link in this E-Mails is clicked through properly. It is important to note that there are two subtypes of distribution E-Mails – weekly digest E-Mails and re-engagement E-Mails (specifically for those who have not opened the platform for a longer time). Results of the query are visualized in Figure 8.

Query:

```
SELECT DATE_TRUNC('week', occurred_at) AS week,  
COUNT(CASE WHEN e.action = 'sent_weekly_digest' THEN e.user_id ELSE NULL END) AS weekly_emails,  
COUNT(CASE WHEN e.action = 'sent_reengagement_email' THEN e.user_id ELSE NULL END) AS  
reengagement_emails,  
COUNT(CASE WHEN e.action = 'email_open' THEN e.user_id ELSE NULL END) AS email_opens,  
COUNT(CASE WHEN e.action = 'email_clickthrough' THEN e.user_id ELSE NULL END) AS link_clickthrough  
FROM tutorial.yammer_emails e  
GROUP BY 1  
ORDER BY 1
```

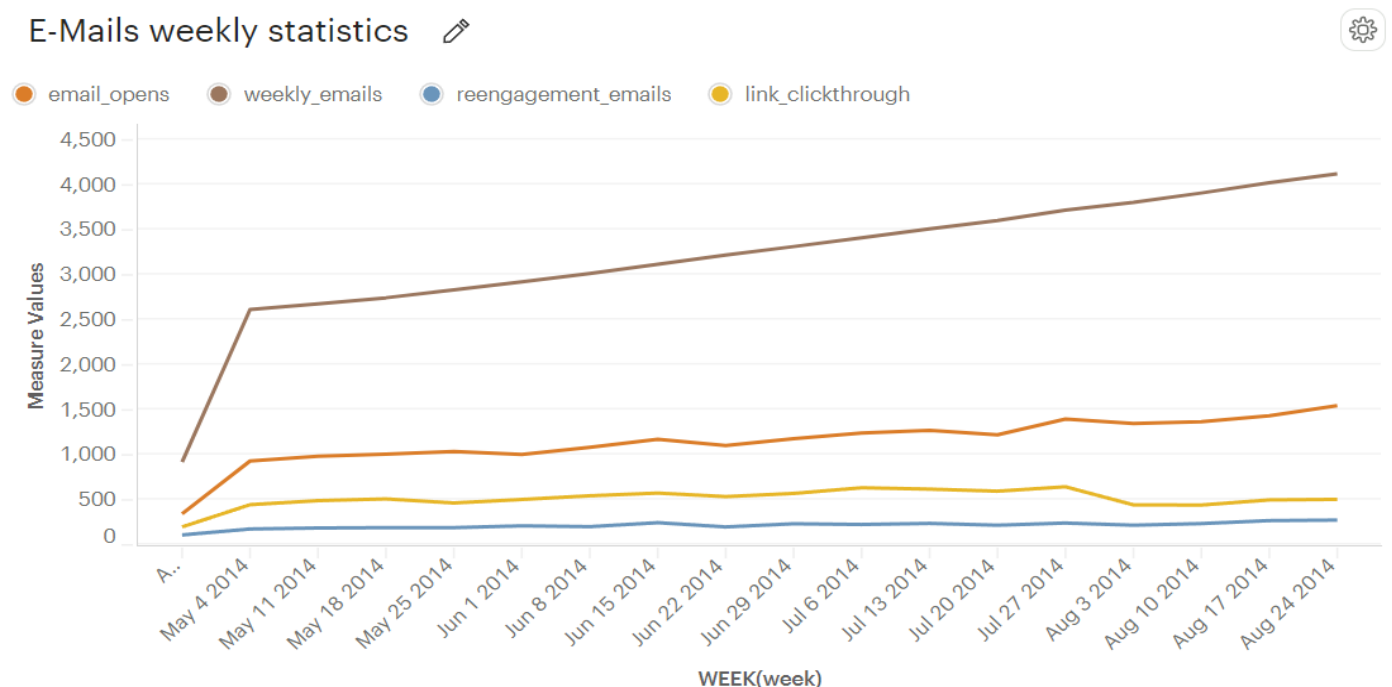


Figure 8. E-Mails weekly statistics (sent, opened, clicked through)

As seen in the results, the number of E-Mails sent increased weekly (as more and more activated users were present in the pool, see brown and blue lines). The number of opened E-Mails (both weekly digest and re-engagement) also increased on a weekly basis (dark orange line). At the same time, a sharp decrease from about 600 in July to approximately 430 in August was observed for link clickthrough. It means that the users did not click the link in the E-Mail to move to Yammer or there was some issue in clicking this link. Connecting this to the Hypothesis 4 it is possible that the link clickthrough did not work on some of the devices due to a technical problem.

I can dig into and check if the clickthrough rate fell only for weekly digest E-Mails or for re-engagement E-Mails, or even for both of them. The logic behind is to check weekly digest open rate, weekly digest clickthrough rate, reengagement open rate and reengagement clickthrough rate.

In order to obtain such data, I need firstly to count how many digest mails are sent, opened and clicked through weekly; as well as the same characteristics for the reengagement emails.

Query:

```
SELECT week,
weekly_opens/CASE WHEN weekly_emails = 0 THEN 1 ELSE weekly_emails END::FLOAT AS weekly_open_rate,
weekly_ctr/CASE WHEN weekly_opens = 0 THEN 1 ELSE weekly_opens END::FLOAT AS weekly_ctr,
retain_opens/CASE WHEN retain_emails = 0 THEN 1 ELSE retain_emails END::FLOAT AS retain_open_rate,
retain_ctr/CASE WHEN retain_opens = 0 THEN 1 ELSE retain_opens END::FLOAT AS retain_ctr
FROM (

-- Selecting all necessary metrics to obtain the statistics of interest later
SELECT DATE_TRUNC('week',e1.occurred_at) AS week,
COUNT(CASE WHEN e1.action = 'sent_weekly_digest' THEN e1.user_id ELSE NULL END) AS weekly_emails,
COUNT(CASE WHEN e1.action = 'sent_weekly_digest' THEN e2.user_id ELSE NULL END) AS weekly_opens,
COUNT(CASE WHEN e1.action = 'sent_weekly_digest' THEN e3.user_id ELSE NULL END) AS weekly_ctr,
COUNT(CASE WHEN e1.action = 'sent_reengagement_email' THEN e1.user_id ELSE NULL END) AS retain_emails,
COUNT(CASE WHEN e1.action = 'sent_reengagement_email' THEN e2.user_id ELSE NULL END) AS retain_opens,
COUNT(CASE WHEN e1.action = 'sent_reengagement_email' THEN e3.user_id ELSE NULL END) AS retain_ctr

FROM tutorial.yammer_emails e1
LEFT JOIN tutorial.yammer_emails e2
ON e2.occurred_at >= e1.occurred_at
AND e2.occurred_at < e1.occurred_at + INTERVAL '5 MINUTE'
AND e2.user_id = e1.user_id
AND e2.action = 'email_open'

LEFT JOIN tutorial.yammer_emails e3
ON e3.occurred_at >= e2.occurred_at
AND e3.occurred_at < e2.occurred_at + INTERVAL '5 MINUTE'
AND e3.user_id = e2.user_id
AND e3.action = 'email_clickthrough'
WHERE e1.occurred_at >= '2014-06-02'
AND e1.occurred_at < '2014-09-01'
AND e1.action IN ('sent_weekly_digest','sent_reengagement_email')
GROUP BY 1
) a
ORDER BY 1
```

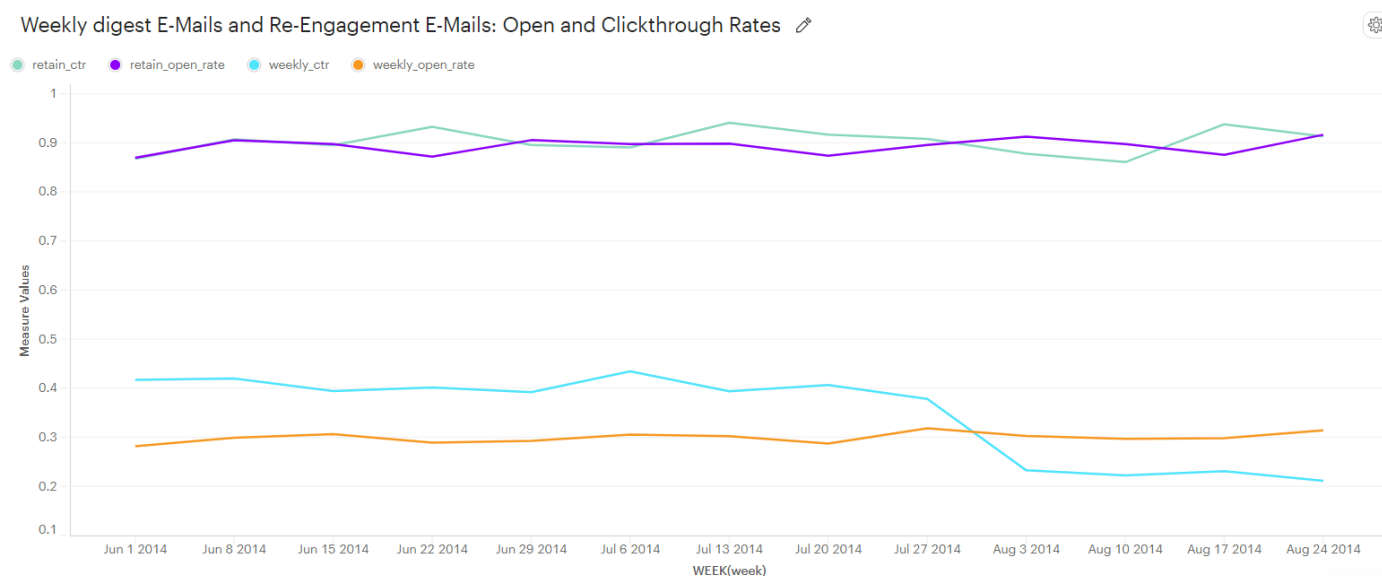


Figure 9. Weekly digest E-Mails and Re-Engagement E-Mails: Open and Clickthrough Rates

As seen from this figure, the weekly digest E-Mails are opened by about 30% of all users, the number is quite constant within all 13 weeks shown here. The re-engagement (retain) E-Mails are opened by approximately 90% of users that obtained these mails. The trend is stable, too. However, if I check how much of the users that opened these E-Mails could click through the link the E-Mail, I see some peculiarities. Approximately 85-95% of all those who opened the re-engagement E-Mails could click through the link, there was some fluctuation during the given 13 weeks, but the numbers are quite stable. In case of the weekly digest E-Mails, only about 40% of users that opened these mails could click through the link in the timeframe between the beginning of June and the end of July, then this number dropped by almost twice towards a bit more than 20% during all the weeks of August. This makes me think that there was indeed an issue with the clickthrough link of the weekly digest E-Mails in August. This confirms somehow the Hypothesis 5.

Conclusions for Problem 1:

Two out of five hypotheses were confirmed to some extent, while three were not of much success.

- major issues were seen with the user engagement of those using smartphones or tablets to log in to Yammer; the product team should check what kind of updates were released in the end of July 2014 to investigate why the issues appeared; it could be also useful to check the reviews of users at Google Play Store or Apple Store to get the direct feedback of users that had troubles at that time.
- some issue was observed with the click-through of weekly digest E-Mails in the end of July and whole August 2014; users could not get to Yammer from the weekly digest mails, giving a hint that there was some issue with the link to Yammer or the E-Mail was shown in a wrong way to a great share of users.

These are two main points that I would share to the Product Team to investigate further, why the user engagements dropped notably.

Problem 2. Digging into the search functionality to revamp it

The Product Team of Yammer is planning its activities for the coming months and would like to know if it is essential to focus on the search services of the platform. It is important at this stage to understand how users interact with the search, which results they obtain, if they are satisfied with the results or not. Based on this, the Product Team may enhance the service or rather invest its precious time into something else, as the improvement of one feature may come at expense of some other potential features.

The search service of Yammer back to 2014 had the following operational principle:

- In every page of the website there is a search box in the header. It prompts users to search for people, groups, and conversations.
- Once a user begins to type in the search box, a dropdown list with the most relevant results appears. The results are separated by category (people, conversations, files, etc.).
- If the user hits enter or selects “view all results” from the dropdown list, he/she is taken to a result page, where the results are separated by tabs for different categories. Each tab is ordered by relevance and chronology with more recent posts among the first results.

For this problem there are mainly two tables used: one with the users, another one is with the yammer events identical to the ones in the first problem. In the second table, I would focus primarily on the events, that have the following names:

- `search_autocomplete` – this happens when a user clicks on any of the search options coming from the dropdown list
- `search_run` – this happens when a user runs a proper search (by clicking at the search symbol) and sees the search results page
- `search_click_X` – this event describes every single action a user takes when he/she clicks on the specific X-th search result (X in this data may have values from 1 to 10)

It is crucial to give the following information based on the data I have:

- Are users generally satisfied with the search or not?
- Is the search worth working at all or is it better to invest the time somewhere else?
- If the search requires some work on it, what should be improved?

Step 1.

First of all, I would like to check if the search is used at all by anyone. For this, I can check, how many times `search_autocomplete` and `search_run` are used on a weekly basis. The first one refers to an event when a user types something in the bar and the drop-down list automatically appears and the user chooses something from it, while the second one refers specifically to those events, when the user clicks the search icon.

Query:

```
SELECT
DATE_TRUNC('week', e.occurred_at) AS week,
COUNT(CASE WHEN e.event_name = 'search_autocomplete' THEN e.event_name ELSE NULL END) AS
total_searches_autocomplete,
COUNT(CASE WHEN e.event_name = 'search_run' THEN e.event_name ELSE NULL END) AS total_searches
FROM tutorial.yammer_events e
GROUP BY 1
```

The results shown in Figure 10 demonstrate that both `search_autocomplete` and `search_run` had the increasing numbers from the middle of May until the end of July 2014. This correlates with the trends shown in the first problem that there were more and more active users during this period of time. Similarly to the number of unique users for August 2014 the large drop is seen in the search activities. On average, people used more `search_autocomplete` than ran the full searches.

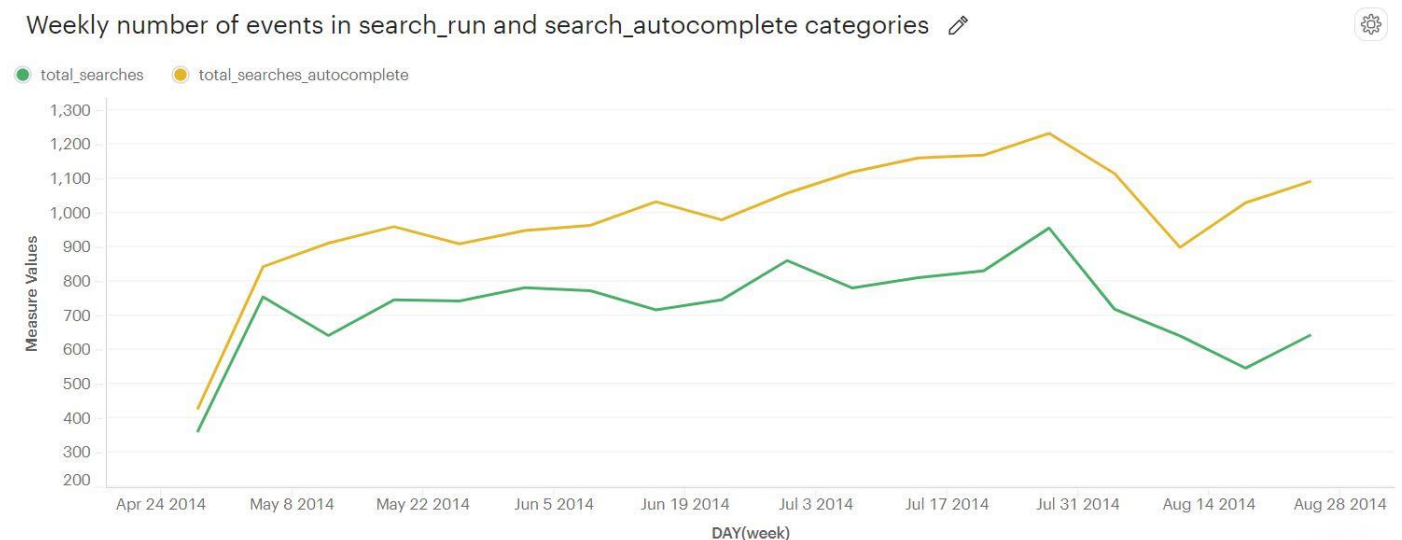


Figure 10. Weekly number of events for the `search_run` and `search_autocomplete` activities

However, only from these two lines it is almost impossible to estimate what type of the search was more efficient and appreciated by the users, as, for example, one user could be satisfied with one `search_autocomplete` event, while the other one had to click on the search button several times within a few minutes.

Therefore, I need to see how these search events are distributed within each session. It can be assumed that the average session does not last more than 10 minutes. If there were no actions happening within 10

minutes, then any action of the user is treated as a new session. The query for looking deeper into this problem is quite complex, but it will be separated with the blank lines to understand it better.

Query:

/ 6) From the obtained table z in the fifth step, I count the percentage of all sessions that had searches with autocomplete, and that had usual search runs. */*

```
SELECT DATE_TRUNC('week',z.session_start) AS week,
COUNT(CASE WHEN z.autocompletes > 0 THEN z.session ELSE NULL END)/COUNT(*)::FLOAT AS
with_autocompletes,
COUNT(CASE WHEN z.runs > 0 THEN z.session ELSE NULL END)/COUNT(*)::FLOAT AS with_runs
FROM (
```

/ 5) From the table obtained in the fourth step, I count the number of autosearches, searches and search clicks grouping results by session_start, session and user_id. */*

```
SELECT x.session_start,
x.session,
x.user_id,
COUNT(CASE WHEN x.event_name = 'search_autocomplete' THEN x.user_id ELSE NULL END) AS autocompletes,
COUNT(CASE WHEN x.event_name = 'search_run' THEN x.user_id ELSE NULL END) AS runs,
COUNT(CASE WHEN x.event_name LIKE 'search_click_%' THEN x.user_id ELSE NULL END) AS clicks
FROM (
```

/ 4) I take the initial yammer_events table and join left to it the obtained after the third block the "session" table. All events from the initial table get extra columns "session" (showing session number) and "session_start". Conditions are added in ON, AND clauses. */*

```
SELECT e.*,
session.session,
session.session_start
FROM tutorial.yammer_events e
LEFT JOIN (
```

/ 3) The obtained "final" table contains two rows having the same session number, I merge them by grouping by user_id and session, while selecting min and max occurred times as the start and the end of the sessions. Many other things could happen between these two times, but as all events were not more than 10 minutes away from each other, this is treated as one session. I am interested here in the number of unique sessions */*

```
SELECT user_id,
session,
MIN(occurred_at) AS session_start,
MAX(occurred_at) AS session_end
FROM (
```

/ 2) Column "Session" is created and only the rows are kept for which preceding or following search events are more or less by 10 minutes than the current row or one of them is empty, meaning the very first or very last action of the user in Yammer. This is reached with WHERE clauses. */*

```
SELECT bounds.*,
CASE WHEN last_event >= INTERVAL '10 MINUTE' THEN id
WHEN last_event IS NULL THEN id
ELSE LAG(id,1) OVER (PARTITION BY user_id ORDER BY occurred_at) END AS session
```

FROM (

-- 1) Three extra columns are added to show for each row what was the previous event, what will be the next event, and the row number.

```
SELECT user_id,
event_type,
event_name,
occurred_at,
occurred_at - LAG(occurred_at,1) OVER (PARTITION BY user_id ORDER BY occurred_at) AS last_event,
LEAD(occurred_at,1) OVER (PARTITION BY user_id ORDER BY occurred_at) - occurred_at AS next_event,
ROW_NUMBER() OVER () AS id
FROM tutorial.yammer_events e
WHERE e.event_type = 'engagement'
ORDER BY user_id,occurred_at
) bounds
```

-- 1) End of block 1.

```
WHERE last_event >= INTERVAL '10 MINUTE'
OR next_event >= INTERVAL '10 MINUTE'
OR last_event IS NULL
OR next_event IS NULL
) final
-- 2) End of block 2.
```

```
GROUP BY 1,2
) session
-- 3) End of block 3.
```

```
ON e.user_id = session.user_id
AND e.occurred_at >= session.session_start
AND e.occurred_at <= session.session_end
WHERE e.event_type = 'engagement'
) x
-- 4) End of block 4.
```

```
GROUP BY 1,2,3
) z
-- 5) End of block 5.
```

```
GROUP BY 1
-- 6) End of block 6.
```

The resulting table has the structure shown in Figure 11.

	week	with_autocompletes	with_runs
1	2014-04-28 00:00:00	0.2251	0.09129
2	2014-05-05 00:00:00	0.2204	0.08835
3	2014-05-12 00:00:00	0.2352	0.07956
4	2014-05-19 00:00:00	0.2319	0.08545
5	2014-05-26 00:00:00	0.2341	0.08560
6	2014-06-02 00:00:00	0.2273	0.08268

Figure 11. First six rows of the resulting table, showing shares of sessions in which search_autocomplete and search_run were used

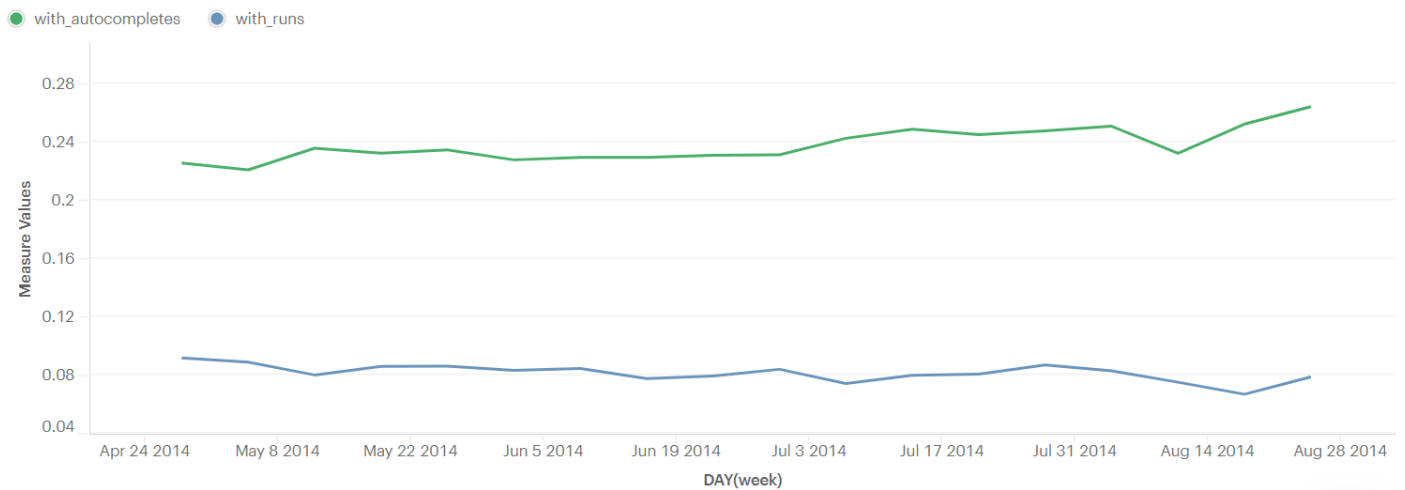


Figure 12. Percentage of sessions where search_autocomplete or search_run was used

Looking at the last Figure, I can infer that the search_autocomplete was used on average in 25% of all sessions, while the search_run was used three times less in about 8% of the sessions. During one session (subsequent activities are not more than 10 minutes apart from each other) each of the search could be used more than once. From these numbers, I can state that the search_autocomplete looks to be more popular among the users. It may also give a hint that such search is already quite successful (and efficient) and the users turn to the search_run only if the autosearch does not show relevant results in the dropdown menu. This could mean that only in about 1/3 of all cases that a person uses the search, he/she uses the search_run and in 2/3 of cases he/she is satisfied with the results of search_autocomplete. However, this is not a 100% guarantee, as a user may be used to clicking the search button immediately without clicking any of the results in the dropdown menu.

Step 2.

As the results of the previous step do not fully give the answers to the questions stated in the beginning, it makes sense to dig into the results of the first step.

Let's see how many times during one session the search_autocomplete is used. The query is similar to the one in Step 1, five core parts are the same, the last one is slightly modified. After the five parts, I got back then a table with the time frames of each session, user related to it, where each row matched one of the events (search_run, search_autocomplete, search_click_X). Then I count search_autocomplete events for each of the given sessions.

Query:

```
SELECT autocompletes,  
COUNT(*) AS sessions  
FROM (  
/*Parts 1-5*/  
GROUP BY 1
```

The results of the query are demonstrated in Figure 13.

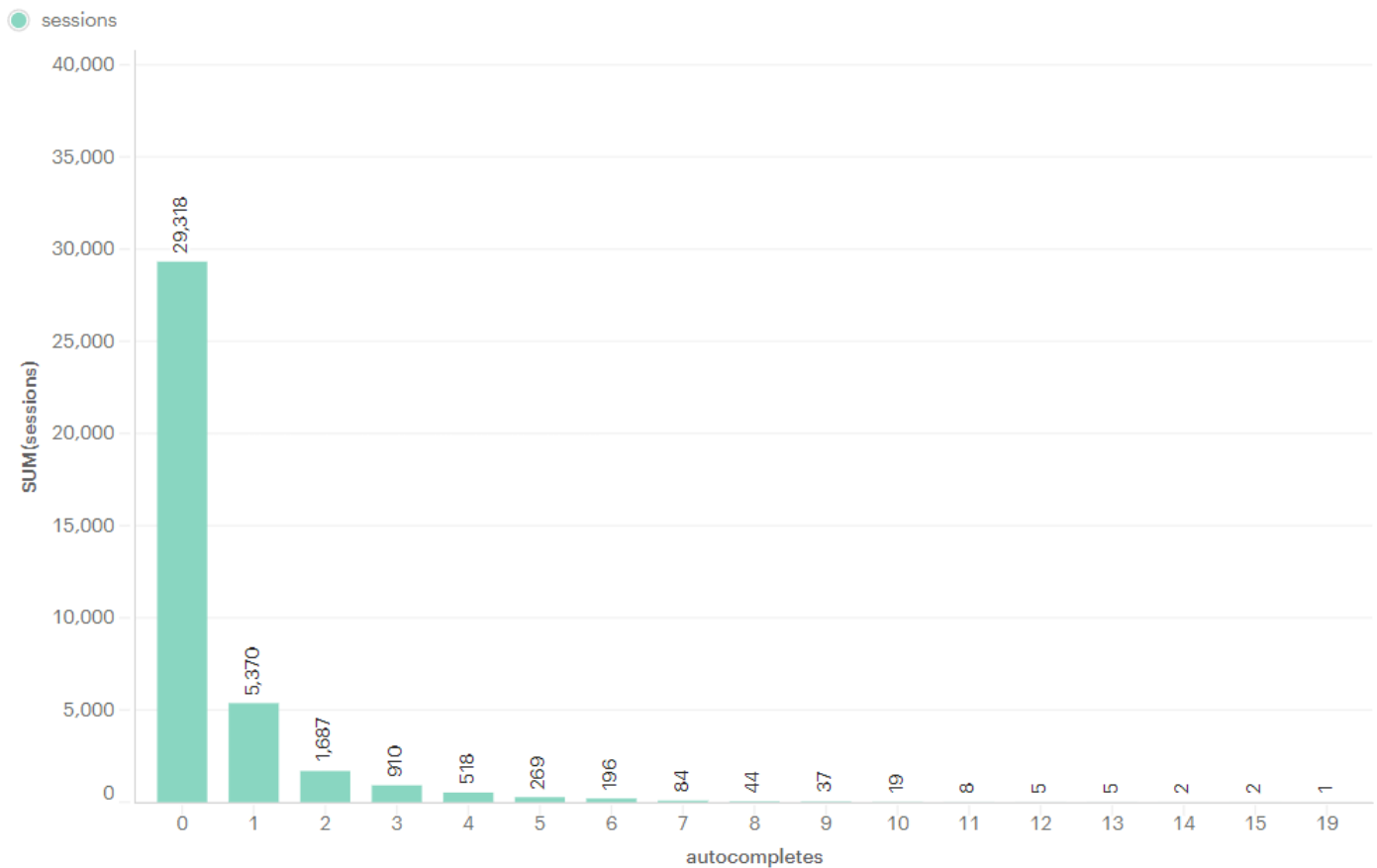


Figure 13. Number of sessions in which search_autocomplete was used vs. Number of times search_autocomplete was used


As seen from the graph, in most sessions search_autocomplete is not used at all. If it was used during a session, in most cases it was used once (5370 cases) and much less two or more times. In extremely rare cases it is run more than 3 times.

Let's now see similarly how the usage of search_run is distributed within sessions. The query is similar to the one in the beginning of the Step 2.

Query:

```
SELECT runs,
COUNT(*) AS sessions
FROM (
/*Parts 1-5*/
GROUP BY 1
```

The obtained results are summarized in Figure 14. They prove what was mentioned in Step 1: search_run was used more rarely than search_autocomplete. In most cases when the search was run, then it was done between 2 and 4 times during a session, while search_autocomplete was mostly run once within a session. This may give an idea that there is something wrong with the usual search and a user has to run it several times within a session to find the desired results. Maybe those who use the search functionality know that the search_run is worse than the search_autocomplete and therefore use it more rarely and if they use it, they have to make several runs within a session to receive the results they are looking for.

Number of sessions (y) vs. Number of search_run (x) was used within a session 

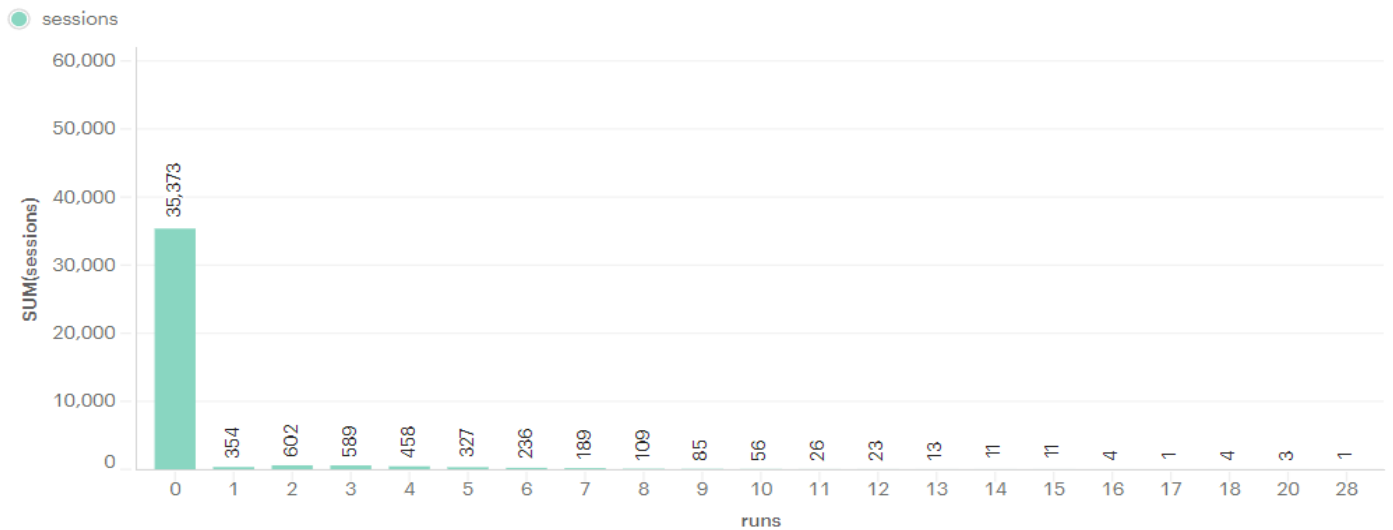



Figure 14. Number of sessions in which search_run was used vs. Number of times search_run was used

Step 3.

I can investigate a bit deeper how many clicks within a session a user does when he/she runs a search. Let's consider only the sessions, during which the search_run was done at least once and find out how many clicks (basically clicks at the results of the search) a user does within such one session. The query for this is similar to the one used in previous steps, the only difference is in the latest part of it, results are in Figure 15.

Query:

```
SELECT clicks,
COUNT(*) AS sessions
FROM (
/*Parts 1-5*/
WHERE runs > 0
GROUP BY 1
```

Number of sessions (y) vs. Number of clicks within a search_run (x) session 

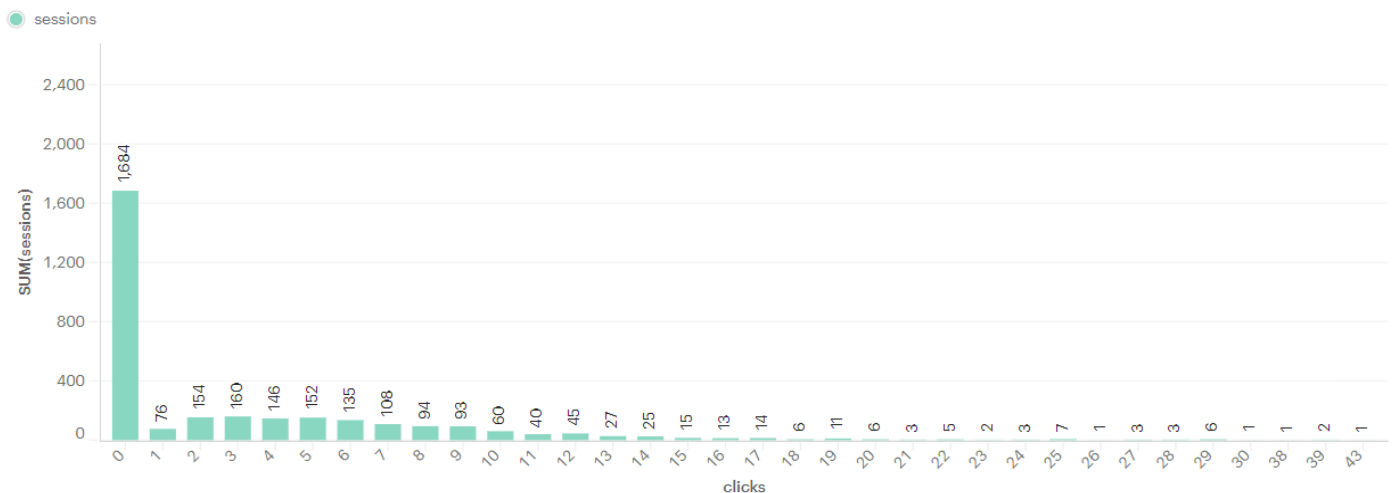


Figure 15. Number of sessions in which search_run was used vs. Number of clicks within each such search_run session

In the majority of the cases when the user clicks at the search button at least once during a session, then he or she never clicks at the obtained results. When he or she does it, it mostly takes for a user to check between 2 and 5 links within the search results, before this user can end up a session. One click is very rarely enough to find an appropriate result meaning that the first results of the search are mostly not something the user looks for.

Let's see if more search_run done by a user within one session would cause more clicks of the shown results. The query is similar to the one before, the difference in its external part is shown here.

Query:

```
SELECT runs,
AVG(clicks)::FLOAT AS avg_clicks
FROM (
/*Parts 1-5*/
WHERE runs > 0
GROUP BY 1
```

The results of this query shown in Figure 16 bring me to a conclusion that on average the trend is preserved: even at increasing number of runs within a session, the number of clicks is below the number of runs. This means that in some of the runs no clicks were done advising that the results could be not very relevant for a user. This may mean that even when more searches were run, the results were not relevant and very few clicks were done to check some of the shown search results.

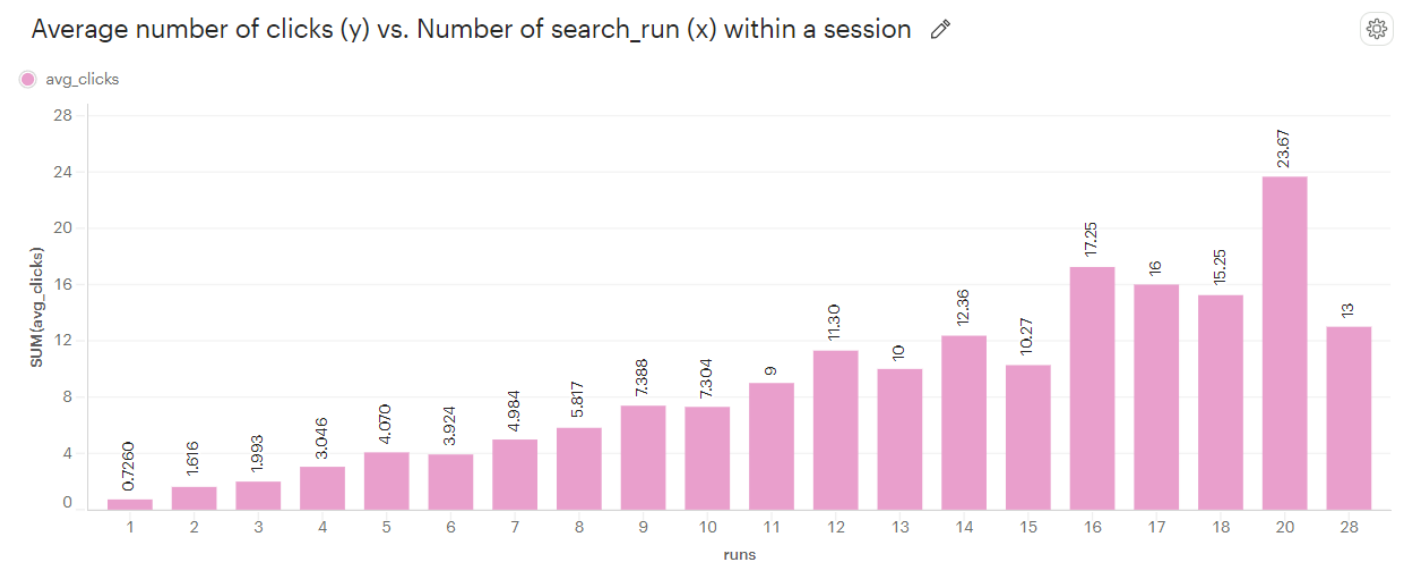


Figure 16. Average number of clicks vs. Number of search_run done within one session

Step 4.

Each search result that is obtained during a search_run has its own unique order number. Here for simplicity only the results having the order from 1 to 10 were considered. I can analyse how many times each of the search results was clicked when the search_run was done.

Query:

```
SELECT
```

```

TRIM('search_click_result_' FROM event_name)::INT AS clicked_link,
COUNT(event_name) AS clicks_counts
FROM tutorial.yammer_events WHERE event_name LIKE 'search_click_result_%%'
GROUP BY 1

```

The results are visualized in Figure 17.

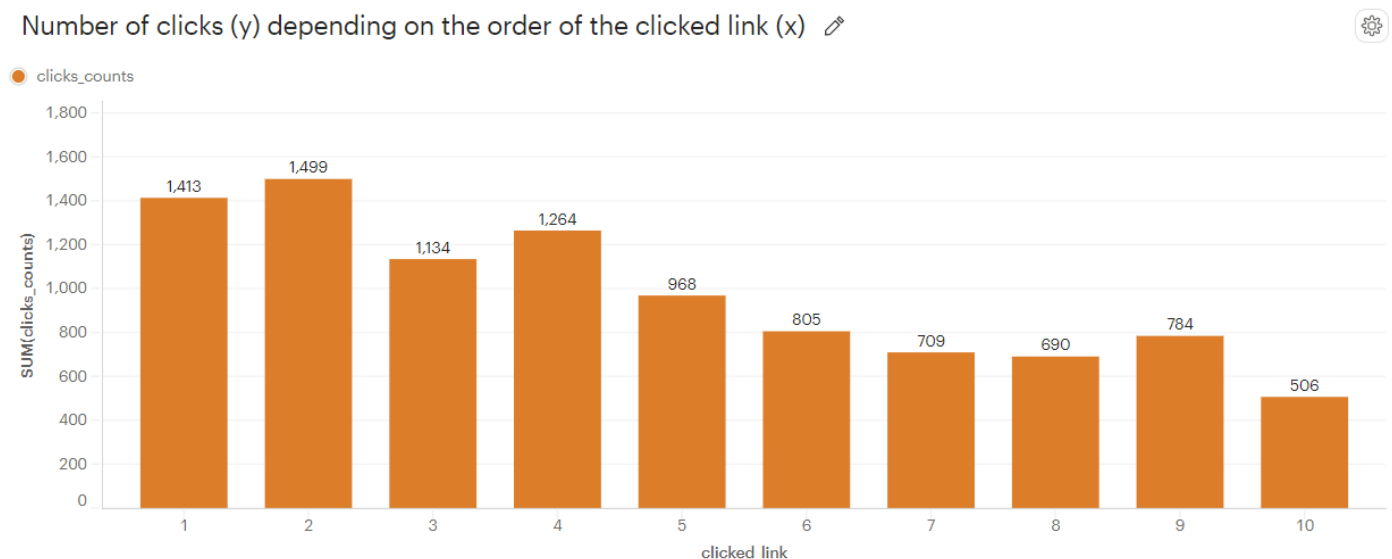


Figure 17. Number of clicks for each of the shown search_run results based on the result's order

If the search_run had brought the users the best results immediately, the users would have clicked the first three links mostly. When the results are not optimal, the users have to scroll down and click the results that have a lower order. Even though the first four results have more than 1000 clicks each, the results from 5 to 10 have also high numbers of clicks (between 506 and 968). This means that the lower ordered results were still clicked quite often and maybe the reason for this was that the first few results are not something what the users look for. It is quite possible that the search algorithms and demonstration of search results need some refinement.

Step 5.

The next interesting thing to check would be how many times do the users come back to the search_run or search_autocomplete within a month from their first search_run or search_autocomplete activities. The query used for this will remind the one in Step 1 of this problem. Three first blocks are the same and the next ones simply advance the resulting table in a way that the number of events where search_run or search_autocomplete was used for each user.

Query:

```

-- 7) The searches are grouped and the number of users is counted. */
SELECT searches,
COUNT(*) AS users
FROM (

```

/ 6) This counts those sessions for each user_id which have more than 0 searches. The results will show in how many sessions the search was used within a month from the very first usage of the search run. The very first search is inside this number, too. */*

```
SELECT user_id,  
COUNT(*) AS searches  
FROM (
```

/ 5) This "takes" x-table from block 4 and counts the number of search_runs for every user. It considers only "search_run" events and ignores the others. */*

```
SELECT x.session_start,  
x.session,  
x.user_id,  
x.first_search,  
COUNT(CASE WHEN x.event_name = 'search_run' THEN x.user_id ELSE NULL END) AS runs  
FROM (
```

/ 4) Table "First" is created to define the first usage of search_run for each user. Then it is joined to the initial "Events" table. The obtained table is connected to "session" table. The final table shows events that happened within a following month from the very first search done by user. */*

```
SELECT e.*,  
first.first_search,  
session.session,  
session.session_start  
FROM tutorial.yammer_events e  
JOIN (  
SELECT user_id,  
MIN(occurred_at) AS first_search  
FROM tutorial.yammer_events  
WHERE event_name = 'search_run'  
GROUP BY 1  
) first  
ON first.user_id = e.user_id  
AND first.first_search <= '2014-08-01'  
LEFT JOIN (
```

-- 1-3) These blocks result in a table with the sessions.

```
SELECT user_id,  
session,  
MIN(occurred_at) AS session_start,  
MAX(occurred_at) AS session_end  
FROM (  
SELECT bounds.*,  
CASE WHEN last_event >= INTERVAL '10 MINUTE' THEN id  
WHEN last_event IS NULL THEN id  
ELSE LAG(id,1) OVER (PARTITION BY user_id ORDER BY occurred_at) END AS session  
FROM (  
SELECT user_id,  
event_type,  
event_name,  
occurred_at,  
occurred_at - LAG(occurred_at,1) OVER (PARTITION BY user_id ORDER BY occurred_at) AS last_event,  
LEAD(occurred_at,1) OVER (PARTITION BY user_id ORDER BY occurred_at) - occurred_at AS next_event,  
ROW_NUMBER() OVER () AS id
```

```

FROM tutorial.yammer_events e
WHERE e.event_type = 'engagement'
ORDER BY user_id, occurred_at
) bounds
WHERE last_event >= INTERVAL '10 MINUTE'
OR next_event >= INTERVAL '10 MINUTE'
OR last_event IS NULL
OR next_event IS NULL
) final
GROUP BY 1,2
) session
-- End of blocks 1-3.

ON session.user_id = e.user_id
AND session.session_start <= e.occurred_at
AND session.session_end >= e.occurred_at
AND session.session_start <= first.first_search + INTERVAL '30 DAY'
) x
-- End of block 4.

GROUP BY 1,2,3,4
) z
-- End of block 5.

WHERE z.runs > 0
GROUP BY 1
) z
-- End of block 6.

GROUP BY 1
ORDER BY 1
-- End of block 7.

```

The results of this long query are shown in Figure 18.

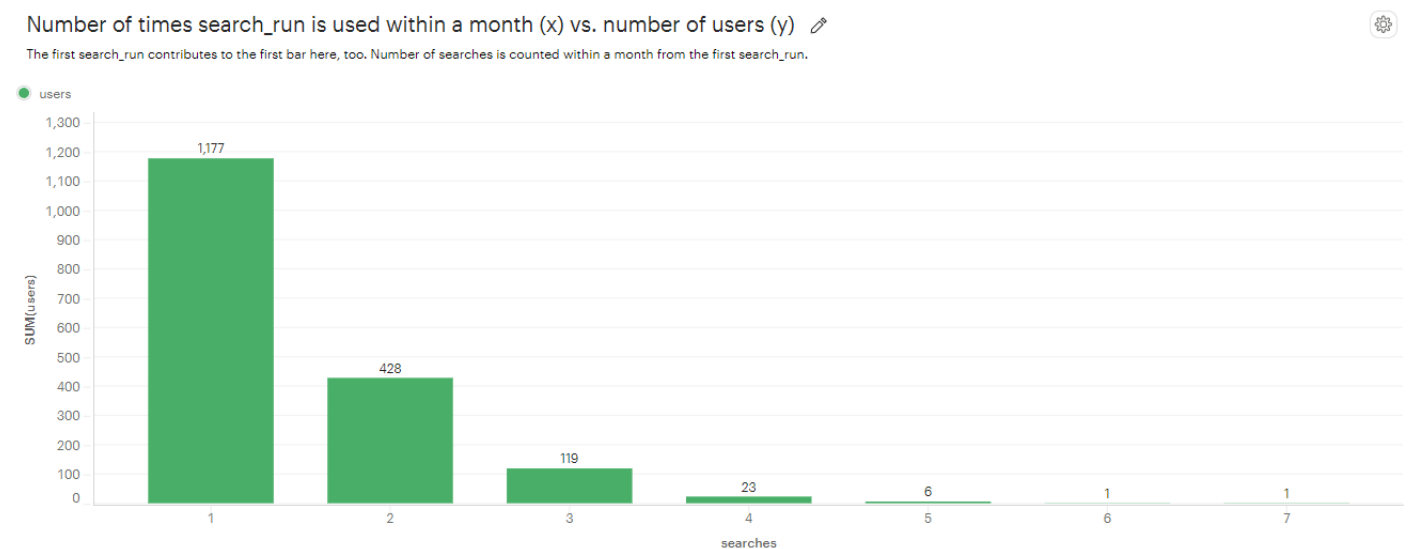


Figure 18. Number of times search_run is used by a user within a month since his/her first search_run vs. Number of such users

As it was calculated in earlier steps, only about 8% of the users have ever used the search_run. The graph above shows that about one third of them used this function at least once more within the month from the first activity (in total there are 1755 users who used the function at least once, 578 of them used twice or more). It means that less than 3% of all users run searches more than once within a month from their first search_run, which is quite a low value.

If I modify the query in this step in a way that the analysis is done for search_autocomplete, then I will obtain the data showing how many times a user comes back to this function within a month from the very first usage of it. The results are summarised in Figure 19.

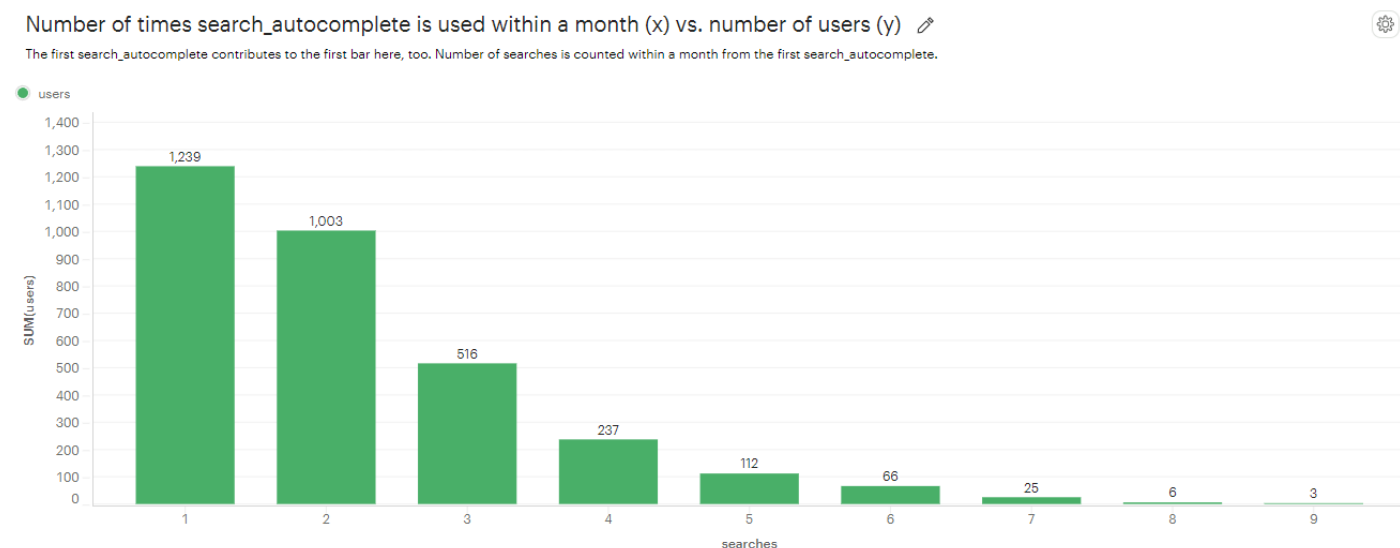


Figure 19. Number of times search_autocomplete is used by a user within a month since his/her first search_autocomplete vs. Number of such users

The results demonstrate that out of all users (3207 people), almost two third (1968) used it more than once. From the previous calculations, it was seen that search_autocomplete was used by 25% of all users, which means that in total about 16% of all users used it more than once within a month from their first usage.

Conclusions for Problem 2:

This all suggests that autocomplete is performing reasonably well, while search runs are not. The users are usually quite satisfied with the results provided by the search_autocomplete function, but less happy with the results of the full search_run. The most obvious place to focus on is the ordering of search results. It's important to consider that users likely run full searches when autocomplete does not provide the things they are looking for, so maybe changing the search ranking algorithm to provide results that are a bit different from the autocomplete results would help. Moreover, it is important to discuss with the Product Team, what expectations they have about the portion of people using the search functions, what results the other platforms have and many other deeper factors to understand if the whole story is worth investing time in it.

Problem 3. A/B tests for the release of new features

Like many companies, Yammer often tests new features on some group of users before releasing them to all of their customers. Such so-called A/B test help analysts and product managers better understand a feature's effect on user behaviour and the overall user experience.

The Product Team approached me on July 1, 2014 asking about the A/B tests and their results for the release of a new feature. The team would like to improve the module at the top of a Yammer feed where users type their messages – “publisher” core. For this, the team split users into 2 groups and run the tests from June 1 to June 30, 2014. During this period, a part of users who logged into Yammer was shown the old version of the publisher (“control group”), another part of users was shown the new version (“treatment group” or “test group”).

According to the results of the A/B test for this feature, the average amount of messages sent in the “treatment group” during the month of June was 4.08, meanwhile this was 2.67 in the “control group”. For such calculations the t-Test and the basics of statistics were applied. In the most tables for A/B tests the following columns are presented:

- **users:** The total number of users shown that version of the publisher.
- **total_treated_users:** The number of users who were treated in either group.
- **treatment_percent:** The number of users in a given group as a percentage of the total number of treated users.
- **total:** The total number of messages posted by the treatment or control group.
- **average:** The average number of messages per user a given group (total/users).
- **rate_difference:** The difference in posting rates between treatment groups (group average - control group average).
- **rate_lift:** The percent difference in posting rates between treatment groups ((group average / control group average) - 1).
- **stdev:** The standard deviation of messages posted per user for users in a group.
- **t_stat:** A test statistic for calculating if average of the treatment group is statistically different from the average of the control group. It is calculated using the averages and standard deviations of the treatment and control groups.
- **p_value:** Used to determine the test's statistical significance (if this value is 0 or very close to 0, then it means that the difference between two groups is very unlikely to be due to a chance).

Apart from two tables that are used in previous problems (Users, Events), two additional tables are used in this problem: experiments (contains data about assignment of a user to one of the groups for an experiment and some other extra information), normal distribution (this one is like from a statistics book with Z-scores).

Having the results of the specific A/B test done by the Product Team I should first of all make sure that the test they run was done correctly (approach itself) and the obtained results (someone's query) are correct; if there are some even minor inefficiencies, the way how to improve them can be proposed and if it is possible to do that having only the data I obtained from the Product Team, it should be done. It makes also sense to check other metrics (e.g., number of login session per user) to be sure that the positive result is not only seen for one metric. In the end, it will be useful to inform the team about my recommendation regarding the roll-out of a new feature or maybe I would even propose to conduct some extra tests.

Step 1.

Let's firstly see the query used to obtain the results indicated above (4.08 vs 2.67 messages per user posted in June 2014 in "test group" and "control group", respectively).

Query:

/ 5) t-analysis is conducted on the obtained data. All necessary statistics are found in a step-by-step procedure. */*

```
SELECT c.experiment,
c.experiment_group,
c.users,
c.total_treated_users,
ROUND(c.users/c.total_treated_users,4) AS treatment_percent,
c.total,
ROUND(c.average,4)::FLOAT AS average,
ROUND(c.average - c.control_average,4) AS rate_difference,
ROUND((c.average - c.control_average)/c.control_average,4) AS rate_lift,
ROUND(c.stdev,4) AS stdev,
ROUND((c.average - c.control_average) / SQRT((c.variance/c.users) + (c.control_variance/c.control_users)),4) AS
t_stat,
(1 - COALESCE(nd.value,1))*2 AS p_value
FROM (
```

/ 4) In this step new columns are created which repeat the control group, plus total number of users is added. */*

```
SELECT *,
MAX(CASE WHEN b.experiment_group = 'control_group' THEN b.users ELSE NULL END) OVER () AS control_users,
MAX(CASE WHEN b.experiment_group = 'control_group' THEN b.average ELSE NULL END) OVER () AS
control_average,
MAX(CASE WHEN b.experiment_group = 'control_group' THEN b.total ELSE NULL END) OVER () AS control_total,
MAX(CASE WHEN b.experiment_group = 'control_group' THEN b.variance ELSE NULL END) OVER () AS
control_variance,
MAX(CASE WHEN b.experiment_group = 'control_group' THEN b.stdev ELSE NULL END) OVER () AS control_stdev,
SUM(b.users) OVER () AS total_treated_users
FROM (
```

/ 3) Grouping users into 2 groups as they were in the experiments and calculating the statistics. */*

```
SELECT a.experiment,
a.experiment_group,
COUNT(a.user_id) AS users,
AVG(a.metric) AS average,
SUM(a.metric) AS total,
STDDEV(a.metric) AS stdev,
VARIANCE(a.metric) AS variance
FROM (
```

/ 2) Joining experiments, events and users tables. Counting events "send_message" and grouping them for each user. Metric in this query is the count of "send_message" events for each user. */*

```
SELECT ex.experiment,
ex.experiment_group,
ex.occurred_at AS treatment_start,
```

```

u.user_id,
u.activated_at,
COUNT(CASE WHEN e.event_name = 'send_message' THEN e.user_id ELSE NULL END) AS metric
FROM (

```

/ 1) Selecting the experiment-related data, excluding all experiments not relevant for the study, excluding excessive columns */*

```

SELECT user_id,
experiment,
experiment_group,
occurred_at
FROM tutorial.yammer_experiments
WHERE experiment = 'publisher_update'
) ex
-- End of block 1.

```

```

JOIN tutorial.yammer_users u
ON u.user_id = ex.user_id
JOIN tutorial.yammer_events e
ON e.user_id = ex.user_id
AND e.occurred_at >= ex.occurred_at
AND e.occurred_at < '2014-07-01'
AND e.event_type = 'engagement'
GROUP BY 1,2,3,4,5
) a
-- End of block 2.

```

```

GROUP BY 1,2
) b
-- End of block 3.

```

```

) c
-- End of block 4.

```

```

LEFT JOIN benn.normal_distribution nd
ON nd.score = ABS(ROUND((c.average - c.control_average)/SQRT((c.variance/c.users) +
(c.control_variance/c.control_users)),3))
-- End of block 5.

```

	experiment	experiment_group	users	total_treated_users	treatment_percent	total	average	rate_difference	rate_lift	stdev	t_stat	p_value
1	publisher_update	control_group	1746	2595	0.6728	4660	2.669	0	0	3.5586	0	1
2	publisher_update	test_group	849	2595	0.3272	3460	4.0754	1.4064	0.527	4.7676	7.6245	0

Figure 20. Results of A/B Test for the potential update of the “publisher” core (send_message events)

Step 2.

If I would like to compare two groups, I should apply other metrics, too. Posting a message in Yammer is not the only metric that could be used to estimate if the increase in user’s activity happened with the introduction of a new function. Similarly, the average number of logins per user during the test period of time (June 2014) can be used to check if the activity of the users in the “test group” indeed increased.

For this, the query from Step 1 can be modified in a way that instead of events “send_message”, events “login” are used. The results are summarized in Figure 21.

	experiment	experiment_group	users	total_treated_users	treatment_percent	total	average	rate_difference	rate_lift	stddev	t_stat	p_value
1	publisher_update	control_group	1746	2595	0.6728	5789	3.3156	0	0	2.5777	0	1
2	publisher_update	test_group	849	2595	0.3272	3481	4.1001	0.7845	0.2366	3.311	6.0676	0

Figure 21. Results of A/B Test for the potential update of the “publisher” core (login events)

On average, the increase from 3.31 to 4.1 logins per user in a given month is observed for those users who have got the new functionality. However, when the company introduce a new feature, it may lead to some other issues, e.g., the service kicks out the users in the “test group”, therefore, they have to log in more frequently.

It is meaningful to check if the number of distinct days when logging in was done increased for the users in the “test group”. For this, the same query is used as a basis, however, the COUNT line should be replaced with the following one in the second block of the query:

`COUNT(DISTINCT DATE_TRUNC('day',e.occurred_at)) AS metric`

Moreover, when joining tables in the second block “e.event_type” should be replaced with “e.event_name”, ‘engagement’ with ‘login’, respectively:

`AND e.event_name = 'login'`

The results of this query are shown in Figure 22. As seen, there were more distinct days with the login activity for each user on average: 3.6 in the “test group” and 3.03 in the “control group”.

	experiment	experiment_group	users	total_treated_users	treatment_percent	total	average	rate_difference	rate_lift	stddev	t_stat	p_value
1	publisher_update	control_group	1746	2595	0.6728	5296	3.0332	0	0	2.1518	0	1
2	publisher_update	test_group	849	2595	0.3272	3057	3.6007	0.5675	0.1871	2.6986	5.3551	0.00000008600

Figure 22. Results of A/B Test for the potential update of the “publisher” core (distinct days for login)

These results make me think from the first glance that there was indeed some improvement of the user posting activity, as well as more login events in the “test group” that had an improved functionality. However, there are still some things that should be checked.

Step 3.

It is reasonable to say that the groups should be formed in a more or less random way. For example, if there are significantly more new users (that registered during June 2014) in one of the groups, it would distort the results. The new users that fall in the “test group” were not familiar with the old functionality and therefore their behaviour is not demonstrating that they really like and use the new functionality more. Moreover, new users may be especially interested in exploring the service and its features, therefore they are more active than the old users. The reverse logic can be applied that the new users are shier and do not use the service much no matter what kind of functionality (old or new) it has. The new users also had less time than the other to be active in the portal during the month of June as their presence duration there was less than 30 days. In order to avoid this, when conducting A/B tests, the users should be split equally.

Let's check for this how the users are split into 2 groups for A/B tests based on their account activation date.

Query:

```
SELECT DATE_TRUNC('month', us.activated_at) AS account_activation_month,  
COUNT(CASE WHEN ex.experiment_group = 'control_group' THEN 1 ELSE NULL END) AS control_group_users,  
COUNT(CASE WHEN ex.experiment_group = 'test_group' THEN 1 ELSE NULL END) AS test_group_users  
FROM tutorial.yammer_experiments ex  
LEFT JOIN tutorial.yammer_users us  
ON ex.user_id = us.user_id  
GROUP BY 1  
ORDER BY 1
```

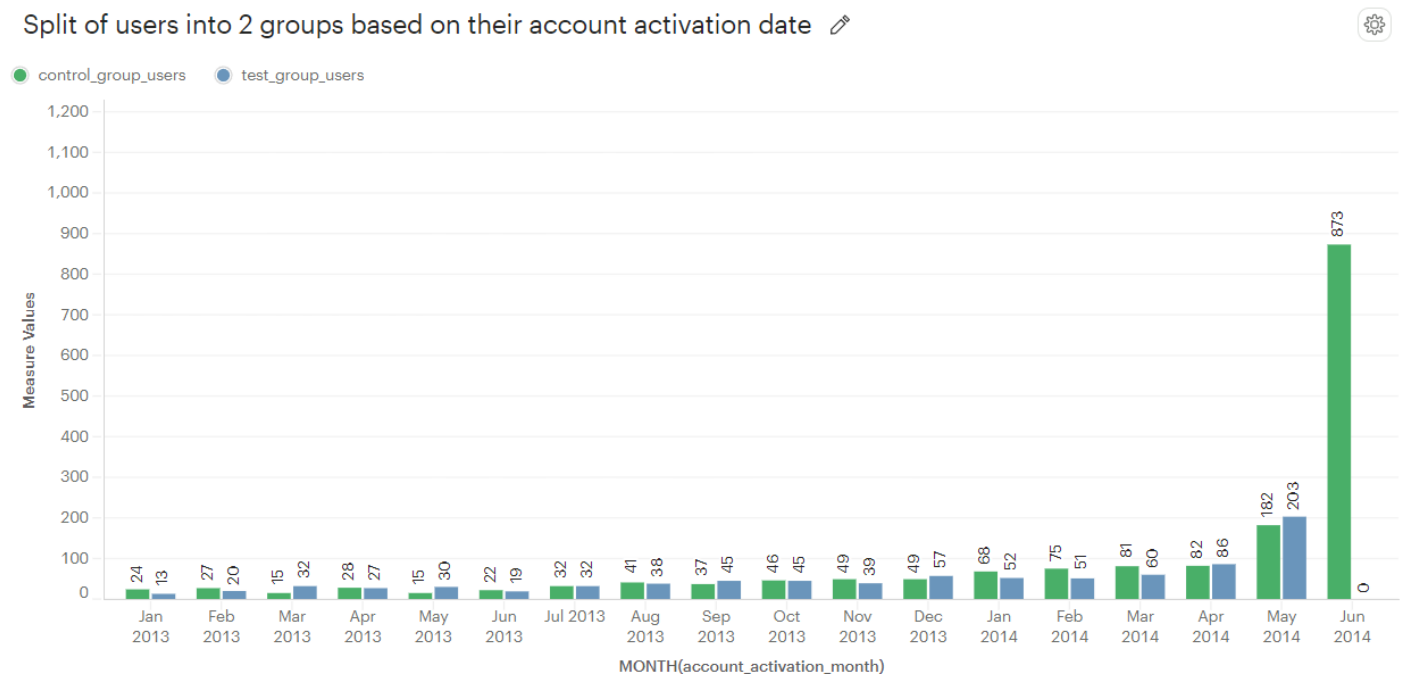


Figure 23. Split of users into 2 groups based on account activation date

The results shown above clearly indicate that for the users, whose accounts were activated before June 2014, the split was more or less equal. However, the new users that joined in June 2014 were fully assigned to the “control group” and did not have the new feature. The portion of the new users was giant in comparison to the other months. Maybe one of the companies introduced Yammer and made all employees to register there, even though most of them did not have any interest in using it, thus shifting the activity of the “control group” downwards and bringing a bias into results. Moreover, new users could have less time to use the service, as for them the user experience time was much less than 30 days.

It can be noted that for a better A/B test in June only employees registered before June should be considered. Let's do it in the next step.

Step 4.

In order to check carry out A/B test and analyse its results only for the users that were activated before June 2014, it is necessary simply to add an extra line in the initial query that was seen in Step 1. The extra line goes to the second block in the place where tables are joined:

AND u.activated_at < '2014-06-01'

The results of this query are in Figure 24.

	experiment	experiment_group	users	total_treated_users	treatment_percent	total	average	rate_difference	rate_lift	stdev	t_stat	p_value
1	publisher_upda...	control_group	873	1722	0.507	2546	2.9164	0	0	3.8758	0	1
2	publisher_upda...	test_group	849	1722	0.493	3460	4.0754	1.159	0.3974	4.7676	5.5266	0.00000003200

Figure 24. Results of A/B Test for the potential update of the “publisher” core (send_message events, users registered before June 2014)

Now the split of users in two groups looks much better, close 1:1. The rate difference between groups is only 1.159 (previously it was 1.406). The results narrowed down, but the result is still significant to neglect it thinking that the difference is due to a chance.

Let's now compare login attempts as before, but showing only users who activated the accounts before June 2014.

	experiment	experiment_group	users	total_treated_users	treatment_percent	total	average	rate_difference	rate_lift	stdev	t_stat	p_value
1	publisher_upda...	control_group	873	1722	0.507	3102	3.5533	0	0	2.8405	0	1
2	publisher_upda...	test_group	849	1722	0.493	3481	4.1001	0.5469	0.1539	3.311	3.674	0.0002388

Figure 25. Results of A/B Test for the potential update of the “publisher” core (login events, users registered before June 2014)

Again, the results narrowed down a bit, but the difference between two groups is still significant enough. Considering all calculation, I would communicate to the Product Team that the results are strong, however they should take care of the proper assignment of new users into both groups of the test and not specifically in one of them.

It is possible that the improvements in logins and message postings are related to some other factors – e.g. devices used for Yammer. As it was seen in problems earlier, there were some difficulties using Yammer from the portable devices, however if they affected experiment groups, I would rather see less posting activities and not more in the “test group”. But it can happen that the cohorts were formed not fully correct: e.g., one cohort is primarily out of portable device users, another one is from computer users, then if one had problems with logging in and posting, it will mean that it will have less logins and posts, while another more. This will have nothing to do with the new feature itself, but rather with the device issues treating new features wrongly. Let's check how the users of various devices were split into two groups and how many events happened for each possible device type users.

Query:

```
SELECT DATE_TRUNC('month', us.activated_at) AS account_activation_month,
ev.device,
COUNT(CASE WHEN ex.experiment_group = 'control_group' THEN 1 ELSE NULL END) AS control_group_users,
COUNT(CASE WHEN ex.experiment_group = 'test_group' THEN 1 ELSE NULL END) AS test_group_users
FROM tutorial.yammer_experiments ex
JOIN tutorial.yammer_users us
ON ex.user_id = us.user_id
JOIN tutorial.yammer_events ev
ON ex.user_id = ev.user_id
```

AND ev.occurred_at >= '2014-06-01' AND ev.occurred_at < '2014-07-01'

GROUP BY 1,2

ORDER BY 1,2

The obtained results are shown in Figure 26.

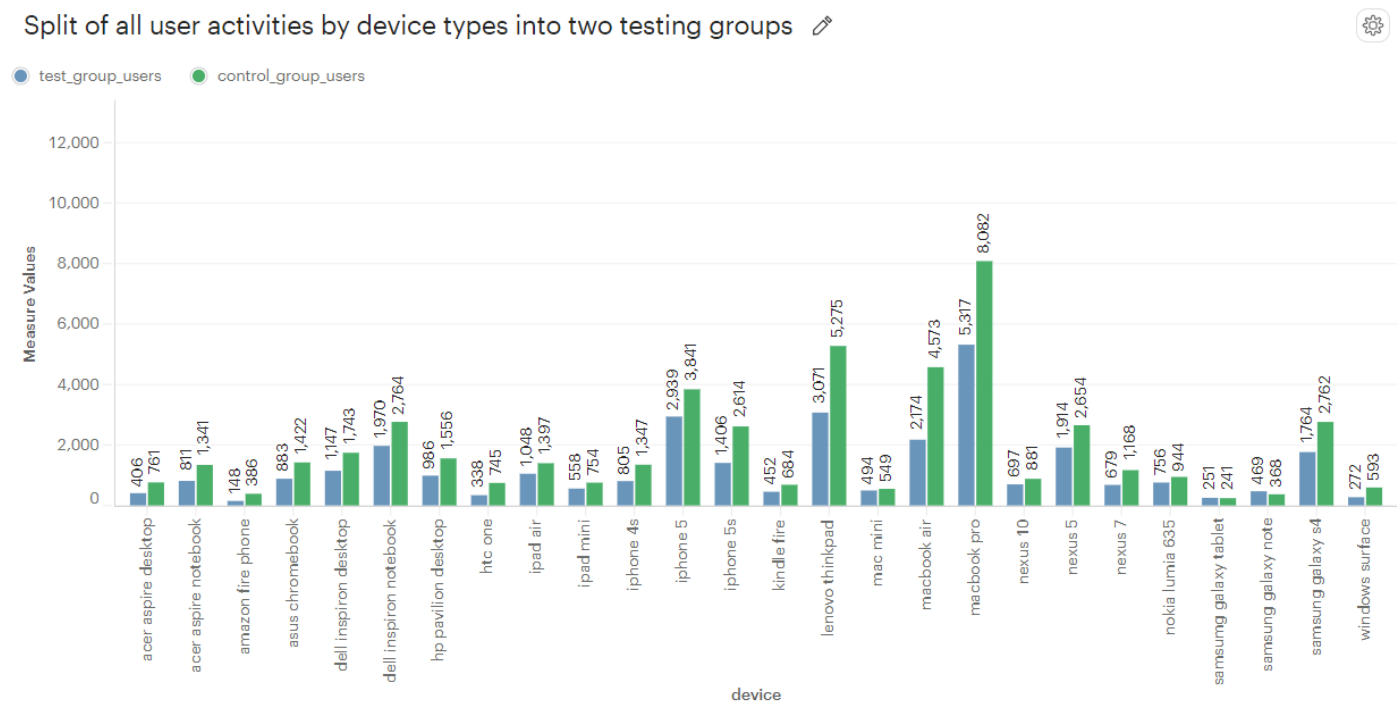


Figure 26. Split of all user activities by device types into two testing groups.

No any major peculiarities are observed within the split of users by device type into two groups. For the major users (iPhone, Lenovo, MacBook, Dell laptops) the split is close to 50/50 in most cases. One user may have multiple devices and use one of them often and another one rarely, thus contributing differently to each device type. However, seeing no any specific pattern here can mean that there are no major problems present.

Conclusions for Problem 3:

Digging deeper into results of the A/B test it was seen that there was an approach error – all new users who joined during June 2014 were assigned to the “control group”, what distorted the results to some extent. Fixing this issue by limiting all users only to those ones, who registered before June 2014, the narrower results were obtained that showed that the new feature caused more activity among the users of the “test group” in comparison to the ones who did not have this feature in the “control group”. Checking other metric – number of logins did not bring much impact, it only proved more that the results of the initial A/B test (after fixing) are strong enough to state the improvement thanks to a new feature. My suggestions would be to change the approach to conducting A/B tests by only treating people who were already active users before the test starts. Moreover, another round of the test with better assignments of people into both groups will improve the quality and trust in the test results.