Yammer Case Study

Yammer is a messenger app. This Case Study has three parts:
1. Engagement - investigating a sudden drop in user engagement
2. Search - comparing the performance of three search features (autocomplete, run, and clicks)
3. A/B Test - evaluating the results of an experiment which showed increase user posting

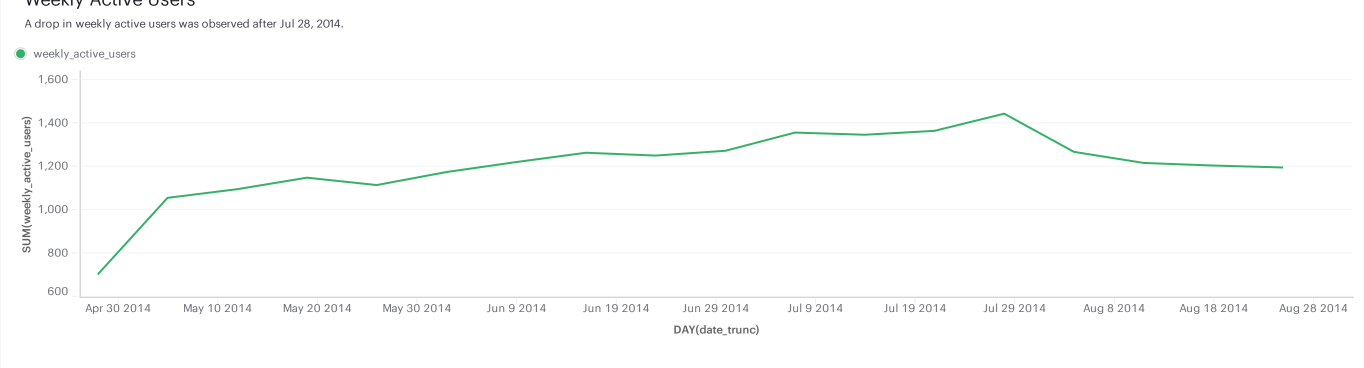
This analysis was prepared with Mode Analytics using a combination of SQL and Python. It is based on the Mode analytics case: https://mode.com/sql-tutorial/sql-business-analytics-training

1. Engagement

Addressing the sudden drop in user engagement is critical for understanding the health and effectiveness of Yammer's platform. The investigation aimed to pinpoint potential causes and develop strategies to mitigate such occurrences in the future.

Weekly Active Users

A drop in weekly active users was observed after Jul 28, 2014.

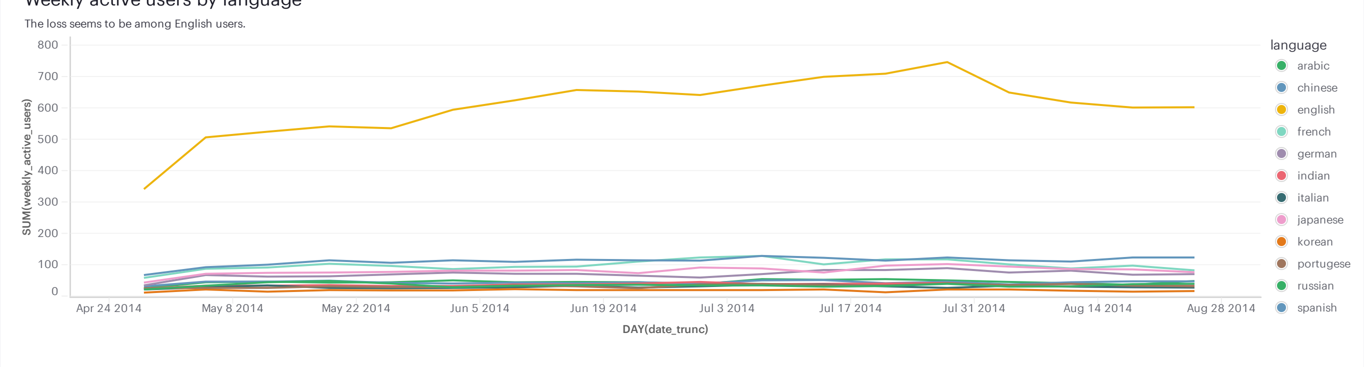


I) HYPOTHESIS DEVELOPMENT:

- Potential Causes: A range of hypotheses were considered, including holidays, broken features, tracking code issues, traffic anomalies, and marketing events. Prioritizing these based on likelihood and data availability guided the investigation.

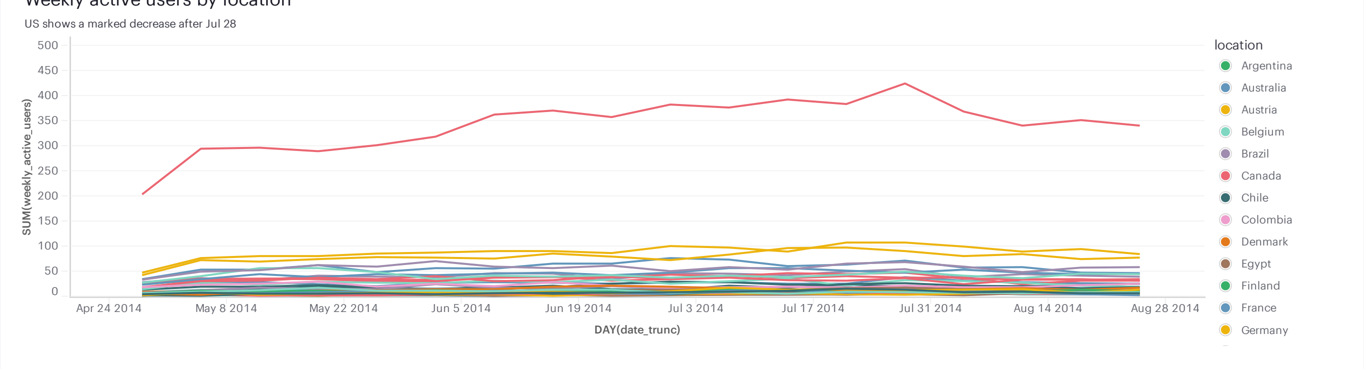
Weekly active users by language

The loss seems to be among English users.



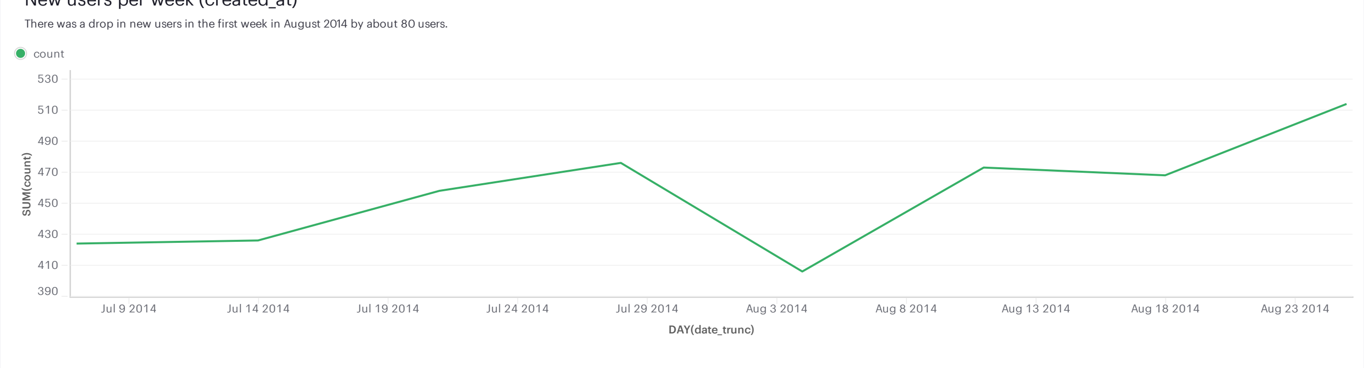
Weekly active users by location

US shows a marked decrease after Jul 28



New users per week (created_at)

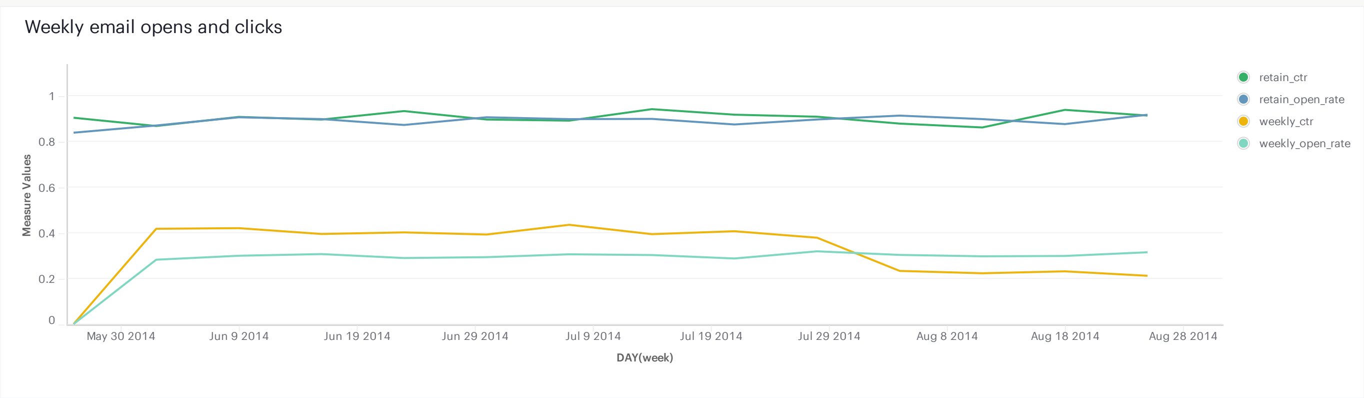
There was a drop in new users in the first week in August 2014 by about 80 users.



II) EMAIL ENGAGEMENT:

- *Digest Emails*: The investigation into digest emails showed a significant decrease in clickthrough rates, indicating a problem area related to user re-engagement.

Weekly email opens and clicks



RECOMMENDATIONS:

Further investigation into the digest emails, and if they are being sent out, and if the links are working.

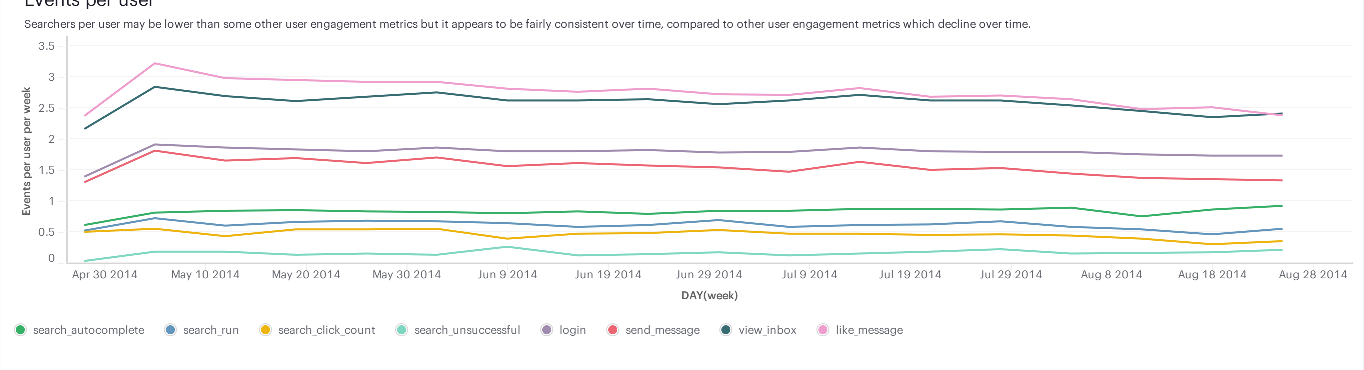
2. Search

In evaluating Yammer's search functionality, we aimed to determine if it effectively helps users find what they're looking for with ease and efficiency. The analyses focused on several key aspects of the search feature: usage frequency, the effectiveness of the autocomplete function, clickthroughs, and user engagement patterns.

Search autocomplete - search results displayed as a user is typing into the search bar
Search run - when a user finishes typing a query and runs search
Search clicks - when a user clicks on one of the search results presented after a search run

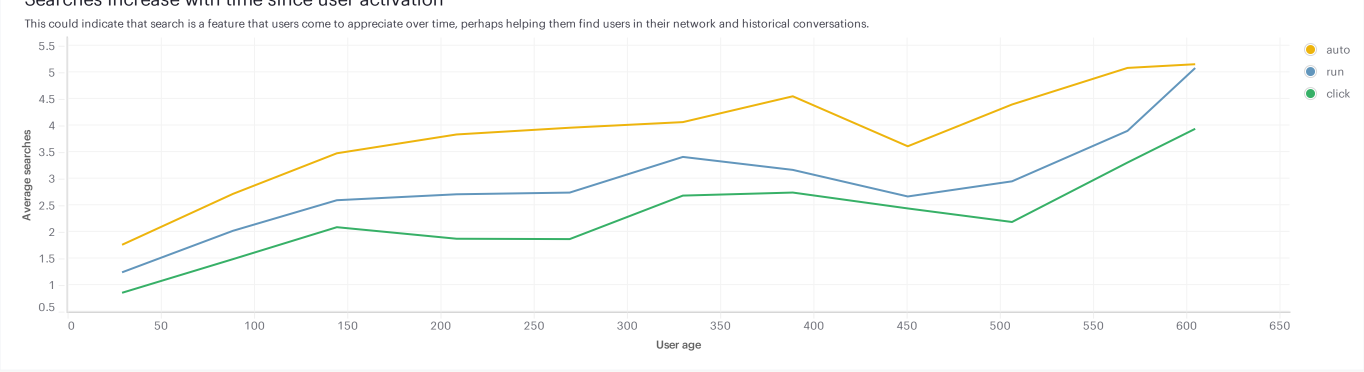
Events per user

Searchers per user may be lower than some other user engagement metrics but it appears to be fairly consistent over time, compared to other user engagement metrics which decline over time.



Searches increase with time since user activation

This could indicate that search is a feature that users come to appreciate over time, perhaps helping them find users in their network and historical conversations.

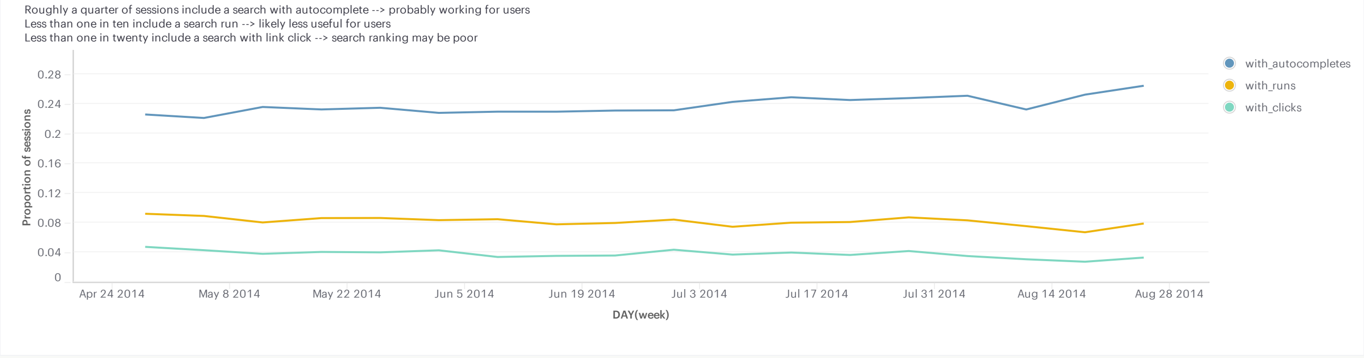


I) SEARCH USE AND FREQUENCY:

- *Usage Patterns*: Autocomplete is used in about 25% of sessions, indicating its value to the users. In contrast, search runs occur in only 8% of sessions, which might suggest lower value to users.
- *Observations*: The consistent use of autocomplete suggests that it is meeting user needs.

Proportion of sessions with searches

Roughly a quarter of sessions include a search with autocomplete --> probably working for users
Less than one in ten include a search run --> likely less useful for users
Less than one in twenty include a search with link click --> search ranking may be poor

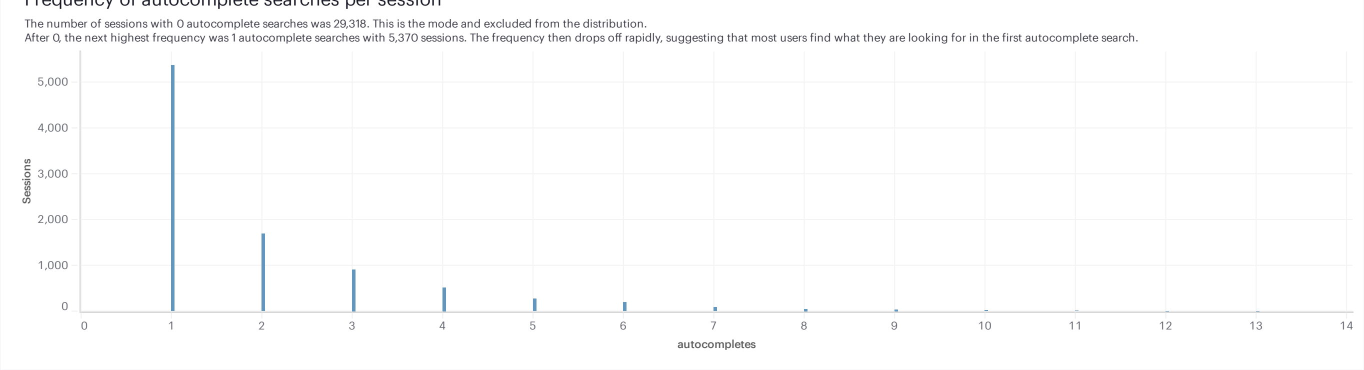


II) AUTOCOMPLETE VS. FULL SEARCH:

- *User Preference*: The data indicates a preference for the autocomplete feature over full search runs, hinting at possible shortcomings in the latter.
- *Search Runs and Clicks*: There is a persistence in the frequency of search runs and clicks per session, suggesting that users might not be finding what they are looking for in the initial search results.

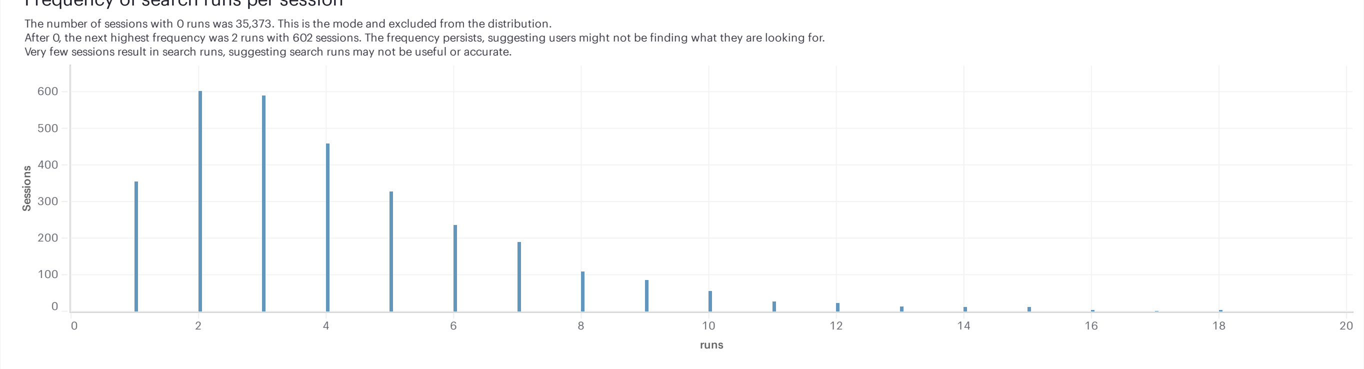
Frequency of autocomplete searches per session

The number of sessions with 0 autocomplete searches was 29,318. This is the mode and excluded from the distribution.
After 0, the next highest frequency was 1 autocomplete searches with 5,370 sessions. The frequency then drops off rapidly, suggesting that most users find what they are looking for in the first autocomplete search.



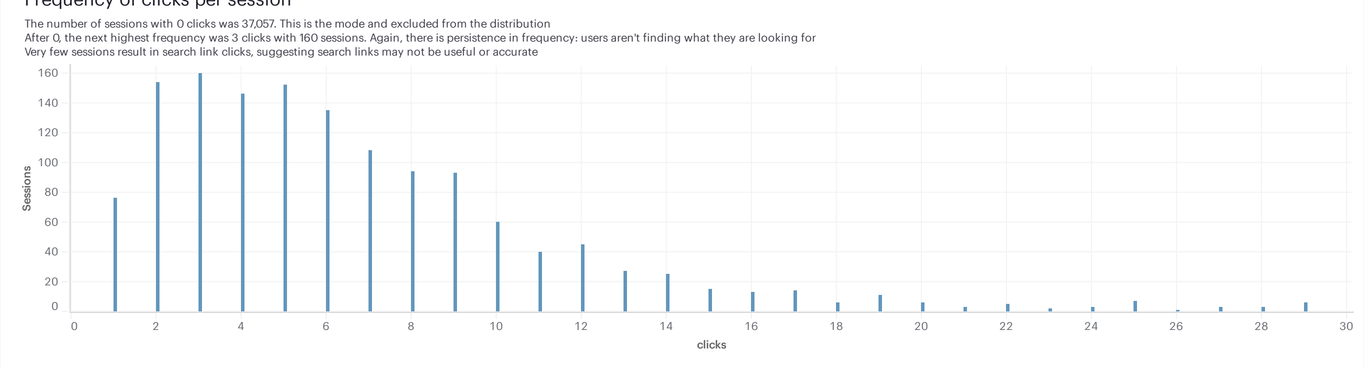
Frequency of search runs per session

The number of sessions with 0 runs was 35,373. This is the mode and excluded from the distribution.
After 0, the next highest frequency was 2 runs with 602 sessions. The frequency persists, suggesting users might not be finding what they are looking for.
Very few sessions result in search runs, suggesting search runs may not be useful or accurate.



Frequency of clicks per session

The number of sessions with 0 clicks was 37,057. This is the mode and excluded from the distribution
After 0, the next highest frequency was 3 clicks with 160 sessions. Again, there is persistence in frequency: users aren't finding what they are looking for
Very few sessions result in search link clicks, suggesting search links may not be useful or accurate



Recommendations:

- Investigate the accuracy of searches and the ranking of links
- Consider what is working about Autocomplete
- Consider whether the search run results should be similar to autocomplete (at least it's working) or different (offer users a different set of options)

3. A/B test

Yammer's A/B test on its publisher update aimed to improve user interaction. The initial results suggested a significant 50% increase in message posting for the treatment group. While promising, such a dramatic shift necessitates a thorough validation process to ensure the integrity and applicability of the results.

I) METHODOLOGICAL RIGOR:

- *Initial Observations*: The early analysis revealed a substantial rise in message posting within the treatment group. However, methodological nuances, such as the treatment of new versus existing users, raised concerns about potential biases in the data.

t-test initial results

After the experiment was run, these results were published indicating a 52% increase in posting in the test group.

experiment_gro...	total_treated_us...	users	treatment_perc...	rate_difference	rate_lift	average	stdev	t_stat	p_value
control_group	2595	1746	0.6728	0	0	2.669	3.5586	0	
test_group	2595	849	0.3272	1.4064	0.527	4.0754	4.7676	7.6245	

Page 1 of 1 — Showing rows 1-2 of 2

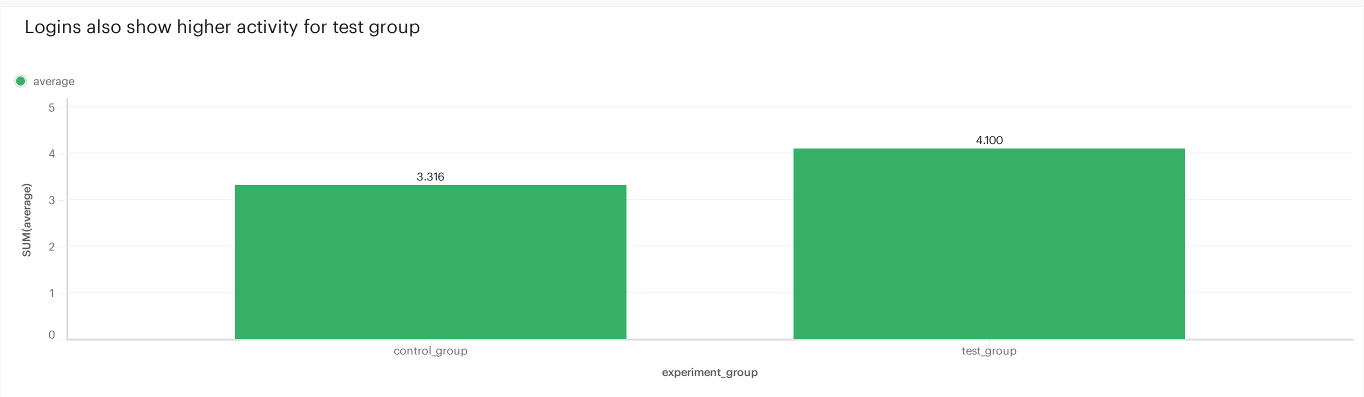
t-test repeated with Python (scipy) - results different but directionally similar

Exploring the differences between the SQL and Python t-tests:
1. Calculation Method: The SQL query calculates the t-statistic using aggregated data and considers variances, sample sizes, and group means. This differs from Python's `scipy.stats.ttest_ind` function, which uses individual data points and has an inherent approach for calculating variances and the t-statistic.
2. Aggregation Impact: The SQL approach involves data aggregation using `GROUP BY`, potentially leading to a loss of data granularity, unlike the Python method that utilizes raw data.
3. Variance and Standard Deviation: SQL explicitly computes variance and standard deviation per group, influencing the t-statistic. Python, however, internally computes these metrics from the data provided.
4. Data Consistency: Discrepancies in results may arise from differences in data filtering and selection between SQL and Python, such as varying time frames, user IDs, or treatment definitions.
5. Rounding Effects: SQL uses the `ROUND` function at several steps, which can slightly alter results. In contrast, Python maintains full data precision up to the final t-statistic calculation.
6. Handling Data Anomalies: SQL and Python may differ in how they handle ties, missing values, or other edge cases, impacting the final results.

II) COMPARATIVE METRICS EXAMINATION:

- *Complementary Metrics*: Other user engagement metrics like login frequency were also analyzed. Consistent improvements across these metrics would corroborate the initial findings.

Logins also show higher activity for test group

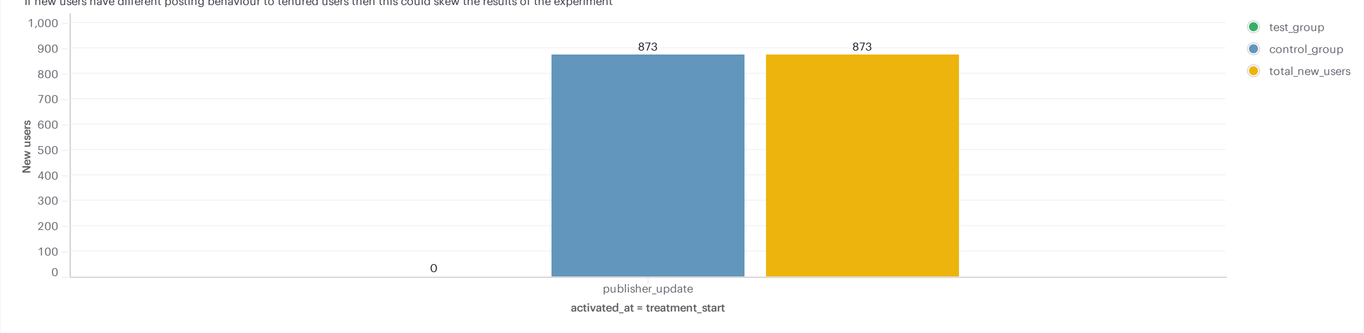


III) DATA INTEGRITY AND GROUP TREATMENT:

- *User Group Assignment*: A crucial discovery was the exclusive allocation of new users to the control group. This skewed the average posting rates, as new users inherently post less, given their shorter exposure to Yammer.

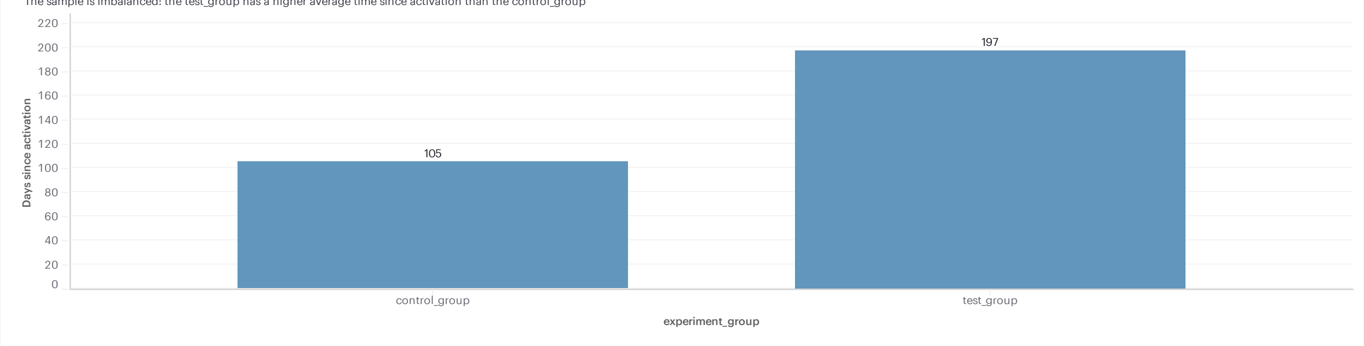
New users all allocated to the control_group

The test group has no users where activated_at = treatment_start
If new users have different posting behaviour to tenured users then this could skew the results of the experiment



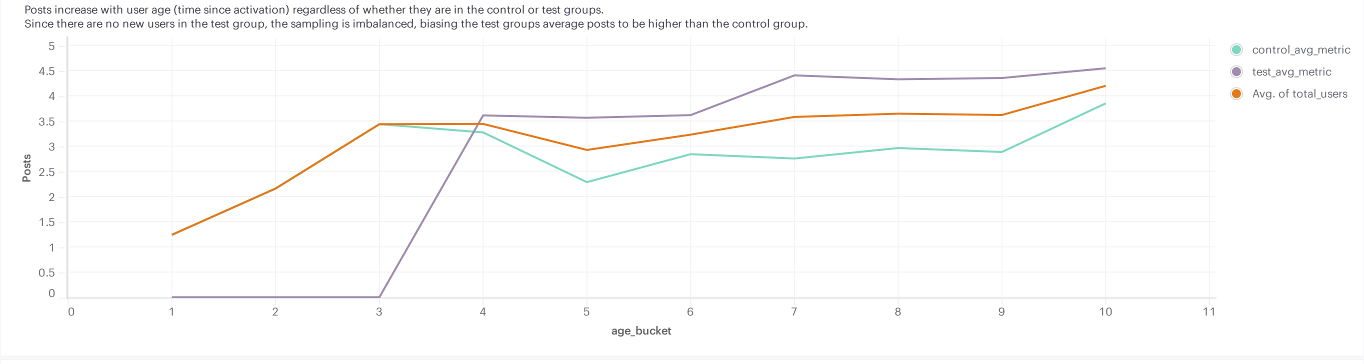
Resulting in higher average time since activation for the test_group

The sample is imbalanced: the test_group has a higher average time since activation than the control_group



New users post less on average

Posts increase with user age (time since activation) regardless of whether they are in the control or test groups.
Since there are no new users in the test group, the sampling is imbalanced, biasing the test groups average posts to be higher than the control group.



Posts by days since activation for the full sample

Notice that new users (close to 0 days on the x-axis) are all from the control group.
Let's ignore new users to create a more balanced sample and redo the initial analysis and t-test.

Posts by days since activation for the filtered sample (removing new users)

IV) REVISED RESULTS AND IMPLICATIONS:

- *Reworked Analysis*: After filtering out newer users, the rate lift was observed to be 41% instead of 52%, suggesting a positive impact, albeit lower than initially reported.

t-test results for filtered groups

When new users were removed, the test group still posted significantly more than the control group, however the rate_lift was lower: 41% vs 52% observed for the full sample.
This suggests that the experiment did have an affect, but potentially lower than the initial analysis indicated.
It would be worth retesting with a balanced sample including new users.

experiment_group	total_treated_users	users	treatment_percent	average	stdev	rate_lift	t_stat	p_value
control_group	1555	793	0.51	2.9231	3.7877	0	0	1
test_group	1555	762	0.49	4.1286	4.8779	0.4124	5.4285	0.00000005600

Page 1 of 1 — Showing rows 1-2 of 2

The t-test was recalculated with Python (scipy) producing the same results:

V) CONCLUSIONS AND RECOMMENDATIONS:

- The revised analysis, while still indicating a positive impact of the treatment, highlights the importance of a balanced and methodologically sound approach in A/B testing.
- Recommendations include re-testing with a more balanced sample that includes new users and ensuring rigorous methodological standards to avoid skewed results.
- Additionally, other sample biases should be considered to ensure groups are truly random: e.g. devices, locations, companys, etc.

Reflections on the Mode / SQL Yammer Tutorial

INTEGRATION OF SQL AND PYTHON IN MODE:

- **Strengths:** Mode's organization of SQL queries and its capability for quick graphical analysis that can be easily embedded into reports are standout features. The integration of a Python notebook with SQL queries is particularly convenient, simplifying complex statistical analyses (like t-tests in Python using scipy) that would be more cumbersome in SQL.
- **Challenges:** A notable limitation is the requirement to restart and rerun the Python notebook each time an SQL query is modified. This aspect introduces some inefficiency, particularly when alternating frequently between SQL and Python during analysis. The Report Builder seemed to slow down and crash once I had multiple SQL and Python analyses in it.

SQL SKILLS DEVELOPMENT:

- **Aggregating User Engagement Data:** The tutorial was instrumental in demonstrating how to aggregate user engagement data into sessions, a method that proved critical for unlocking numerous insights.
- **Complexity and Reusability:** While nested SQL queries required for session aggregation can be complex, they are easier to understand and build incrementally. Starting from simpler inner queries and expanding to more complex outer layers also creates reusable components for future analyses.
- **Documentation:** It's easy to populate a workbook with many SQL queries. If they aren't well labelled and ordered it quickly becomes a mess. Queries can only be ordered alphabetically. Mode would do well to improve query navigation.