Mode Analytics Practice – Validating an A/B Test Result

The background and description of this practice can be found in the link:
https://community.modeanalytics.com/sql/tutorial/validating-ab-test-results/

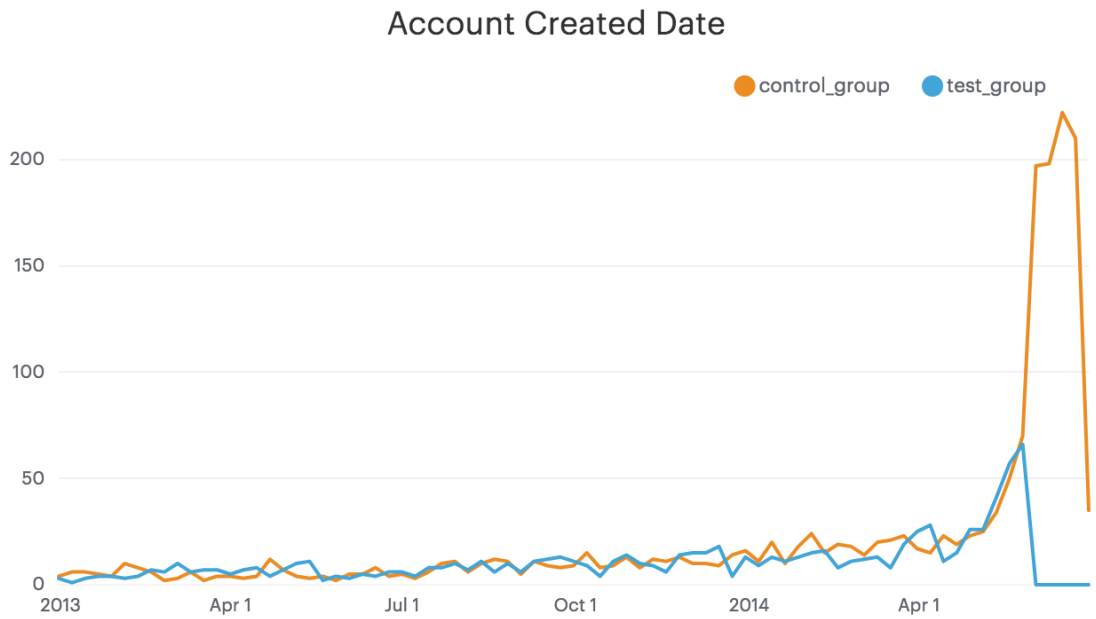
Orientation

To validate the result of an A/B test, it is worthwhile to verify the basic condition of A/B test has been met:

1. The user compositions under each control and test group are similar, i.e. there's no significant differences that may affect the result.
2. The difference of end result is significant. This will require us to investigate the deviation of the end result of each group to calculate the probability of type I or type 2 error.

Analysis

1. First we examine the composition of each group. We found that all the new users who joined after July 2nd have been included into the control group. This would lead potential error to the test result, since the new users might have different behavior that might affect the result, for example, the new users might naturally have lower engage rate. It will make sense to remove those users from the control group and verify the test result again.



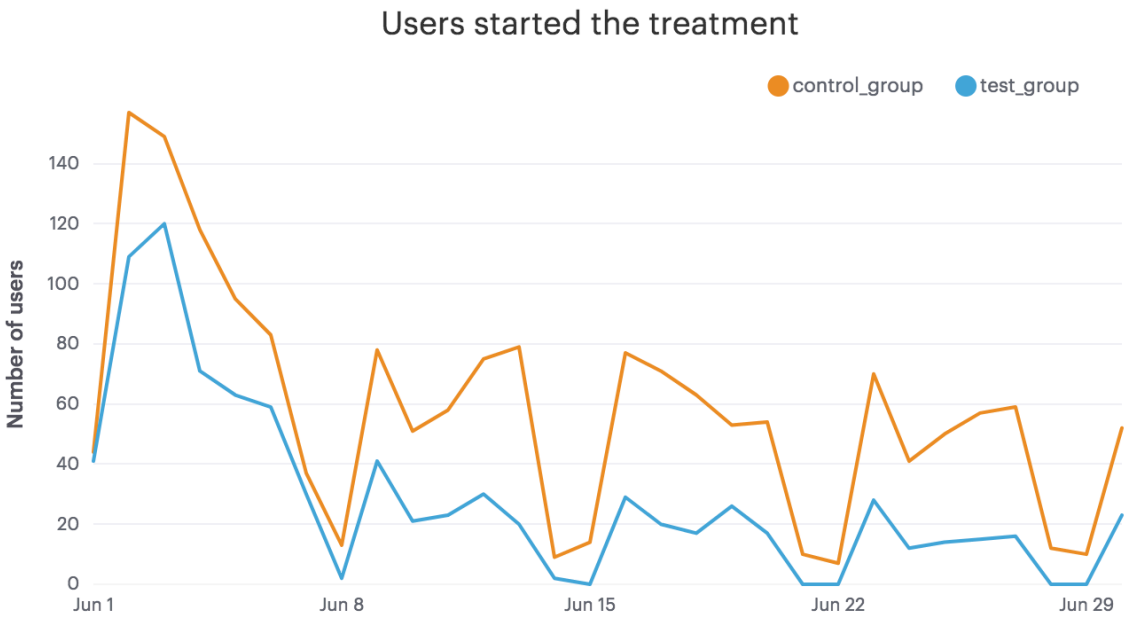
See Appendix 1 for the code of this query.

2. After eliminating the factors that may impact the result except the variable that we'd like to test, we'd like to examine if the result each group has significant difference to reject the alternative hypothesis, i.e. the new feature is effective.

   There are a few metrics that can help us understand the effectiveness of the new features:
   a. The rate of sending message
   b. The rate of login
   c. The rate of engagement

   Notice that we are using rate instead of the count of the metrics, since we are trying to verify each user's behavior during the period of the experiment. And this actually raised an interesting question: does every user was treated to the test in the same amount of time?

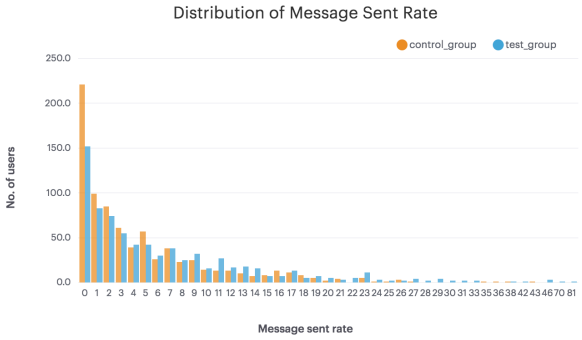
   We can actually query the database:

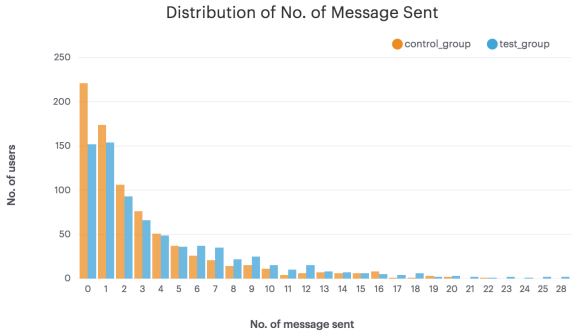


   See Appendix 2 for the code of this query.

   We can clearly see that the treatment of the test to each user was started in different time. This implies that some users were tested in longer period, which likely to engage more activity than those who were tested in shorter period.

To adjust this, we'll need to calculate the rate of each activity listed above by counting the number of each activity and divided by the period they were subjected to the test.

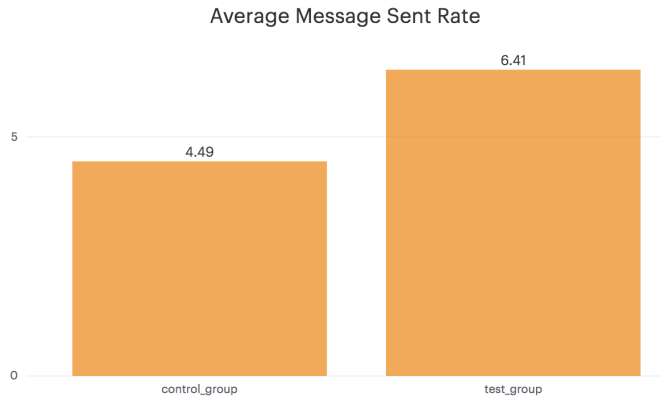




See Appendix 3 for the code of this query.

After the adjustment, we can calculate the mean, standard deviation and standard error of each metrics within each group in order to test the hypothesis.

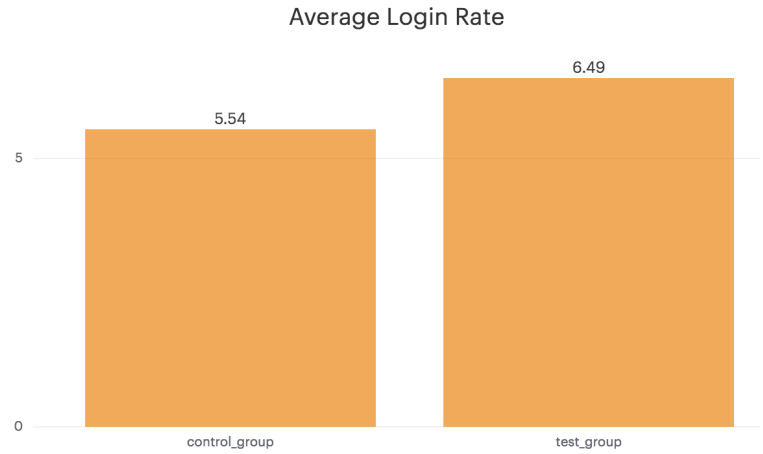
Message sent rate:



experiment_group	sample_size	mean	stdev	standard_err_deviation
control_group	797	4.49	5.68	0.2
test_group	760	6.41	7.98	0.29

See Appendix 4 for the code of this query

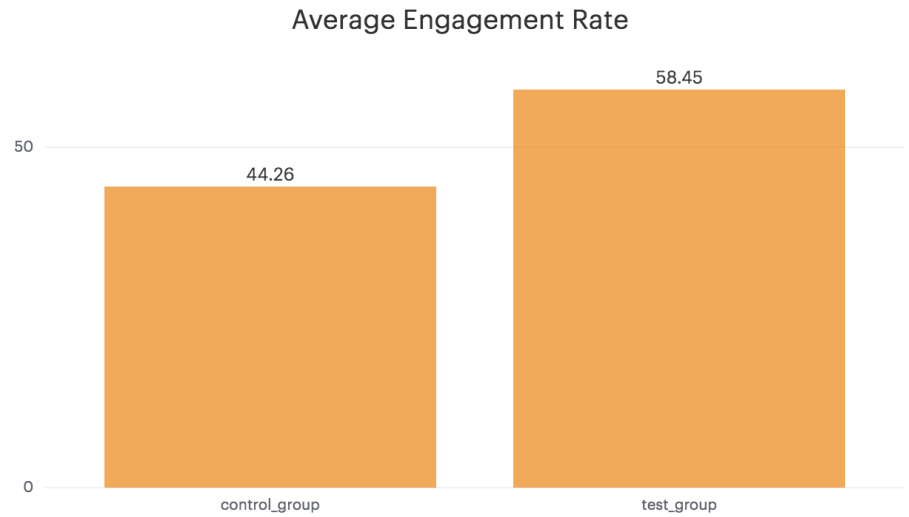
Login rate:



experiment_group	sample_size	mean	stdev	standard_err_deviation
control_group	797	5.54	3.92	0.14
test_group	760	6.49	5.07	0.18

See Appendix 5 for the code of this query

Engagement rate:



experiment_group	sample_size	mean	stdev	standard_err_deviation
control_group	797	44.26	44.78	1.59
test_group	760	58.45	60.7	2.2

See Appendix 6 for the code of this query

Conclusion:

Base on the query results above, we can confidently say that the new feature will increase the rate of sending message, login and engagement.

Appendix 1

```
SELECT DATE_TRUNC('week', u.created_at) AS day,
    COUNT(CASE WHEN experiment_group = 'control_group' THEN u.user_id ELSE NULL END) AS
"control_group",
    COUNT(CASE WHEN experiment_group = 'test_group' THEN u.user_id ELSE NULL END) AS "test_group"
  FROM tutorial.yammer_users u
  JOIN tutorial.yammer_experiments e
  ON  u.user_id = e.user_id
  GROUP BY 1
```

Appendix 2

```
SELECT DATE_TRUNC('day', e.occurred_at) AS day,
    COUNT(CASE WHEN experiment_group = 'control_group' THEN u.user_id ELSE NULL END) AS
"control_group",
    COUNT(CASE WHEN experiment_group = 'test_group' THEN u.user_id ELSE NULL END) AS "test_group"
  FROM tutorial.yammer_users u
  JOIN tutorial.yammer_experiments e
  ON  u.user_id = e.user_id
GROUP BY 1
```

Appendix 3

Query for No. of message sent

```
SELECT u.user_id, e.experiment_group,
  COUNT(CASE WHEN ev.event_name = 'send_message' THEN u.user_id ELSE NULL END) AS metrics
FROM(
  SELECT user_id
    FROM tutorial.yammer_users
    WHERE created_at <= '2014-05-26') u
JOIN tutorial.yammer_experiments e
ON u.user_id = e.user_id
JOIN tutorial.yammer_events ev
ON u.user_id = ev.user_id
WHERE ev.occurred_at > '2014-06-01'
  AND ev.occurred_at < '2014-06-30'
  AND e.experiment = 'publisher_update'
GROUP BY 1,2
ORDER BY metrics, experiment_group
```

Query for message sent rate

```
SELECT ROUND(CAST(metrics_freq AS NUMERIC),0) AS metrics_rate,
      experiment_group,
      COUNT(user_id)
FROM(
  SELECT user_id, experiment_group, metrics,
    metrics/((EXTRACT(days from ('2014-06-30' - occurred_at::timestamp))*24+
    EXTRACT(hours from ('2014-06-30' - occurred_at::timestamp)))/
    ((EXTRACT(days from ('2014-06-30' - '2014-05-31'::timestamp)))*24)) AS metrics_freq
  FROM(
    SELECT u.user_id, e.experiment_group, e.occurred_at,
      COUNT(CASE WHEN ev.event_name = 'send_message' THEN u.user_id ELSE NULL END) AS metrics
    FROM(
      SELECT user_id
        FROM tutorial.yammer_users
        WHERE created_at <= '2014-05-26') u
    JOIN tutorial.yammer_experiments e
    ON u.user_id = e.user_id
    JOIN tutorial.yammer_events ev
    ON u.user_id = ev.user_id
    WHERE ev.occurred_at > '2014-06-01'
      AND ev.occurred_at < '2014-06-30'
      AND e.experiment = 'publisher_update'
    GROUP BY 1,2,3
    ORDER BY metrics, experiment_group) c
  ORDER BY metrics_freq, experiment_group) d
  GROUP BY 1,2
  ORDER BY metrics_rate, experiment_group
```

Appendix 4

```
SELECT
  s.experiment_group,
  s.sample_size,
  ROUND(s.mean,2) AS mean,
  ROUND(s.stdev,2) AS stdev,
  ROUND(s.standard_err_deviation,2) AS standard_err_deviation

FROM(
  SELECT
    experiment_group,
    CAST(COUNT(user_id) AS NUMERIC) AS sample_size,
    CAST(SUM(metrics_freq)/COUNT(user_id) AS NUMERIC) AS mean,
    CAST(STDDEV(metrics_freq) AS NUMERIC) AS stdev,
    CAST(STDDEV(metrics_freq)/SQRT(COUNT(user_id)) AS NUMERIC) AS standard_err_deviation

  FROM(
    SELECT user_id, experiment_group,
      metrics/((EXTRACT(days from ('2014-06-30' - occurred_at::timestamp))*24+
      EXTRACT(hours from ('2014-06-30' - occurred_at::timestamp)))/
      ((EXTRACT(days from ('2014-06-30' - '2014-06-01'::timestamp)))*24)) AS metrics_freq
    FROM(
      SELECT u.user_id, e.experiment_group, e.occurred_at,
        COUNT(CASE WHEN ev.event_name = 'send_message' THEN u.user_id ELSE NULL END) AS metrics
      FROM(
        SELECT user_id
          FROM tutorial.yammer_users
          WHERE created_at <= '2014-05-26') u
      JOIN tutorial.yammer_experiments e
      ON u.user_id = e.user_id
      JOIN tutorial.yammer_events ev
      ON u.user_id = ev.user_id
      WHERE ev.occurred_at > '2014-06-01'
        AND ev.occurred_at < '2014-06-30'
        AND e.experiment = 'publisher_update'
      GROUP BY 1,2,3
      ORDER BY metrics, experiment_group) c
      ORDER BY metrics_freq, experiment_group) d
  GROUP BY 1
  ORDER BY 1) s
```

Appendix 5

```
SELECT
  s.experiment_group,
  s.sample_size,
  ROUND(s.mean,2) AS mean,
  ROUND(s.stdev,2) AS stdev,
  ROUND(s.standard_err_deviation,2) AS standard_err_deviation

FROM(
  SELECT
    experiment_group,
    CAST(COUNT(user_id) AS NUMERIC) AS sample_size,
    CAST(SUM(metrics_freq)/COUNT(user_id) AS NUMERIC) AS mean,
    CAST(STDDEV(metrics_freq) AS NUMERIC) AS stdev,
    CAST(STDDEV(metrics_freq)/SQRT(COUNT(user_id)) AS NUMERIC) AS standard_err_deviation

  FROM(
    SELECT user_id, experiment_group,
      metrics/((EXTRACT(days from ('2014-06-30' - occurred_at::timestamp))*24+
      EXTRACT(hours from ('2014-06-30' - occurred_at::timestamp)))/
      ((EXTRACT(days from ('2014-06-30' - '2014-06-01'::timestamp)))*24)) AS metrics_freq
    FROM(
      SELECT u.user_id, e.experiment_group, e.occurred_at,
        COUNT(CASE WHEN ev.event_name = 'login' THEN u.user_id ELSE NULL END) AS metrics
      FROM(
        SELECT user_id
          FROM tutorial.yammer_users
          WHERE created_at <= '2014-05-26') u
      JOIN tutorial.yammer_experiments e
      ON u.user_id = e.user_id
      JOIN tutorial.yammer_events ev
      ON u.user_id = ev.user_id
      WHERE ev.occurred_at > '2014-06-01'
        AND ev.occurred_at < '2014-06-30'
        AND e.experiment = 'publisher_update'
      GROUP BY 1,2,3
      ORDER BY metrics, experiment_group) c
      ORDER BY metrics_freq, experiment_group) d
  GROUP BY 1
  ORDER BY 1) s
```

Appendix 6

```
SELECT
  s.experiment_group,
  s.sample_size,
  ROUND(s.mean,2) AS mean,
  ROUND(s.stdev,2) AS stdev,
  ROUND(s.standard_err_deviation,2) AS standard_err_deviation

FROM(
  SELECT
    experiment_group,
    CAST(COUNT(user_id) AS NUMERIC) AS sample_size,
    CAST(SUM(metrics_freq)/COUNT(user_id) AS NUMERIC) AS mean,
    CAST(STDDEV(metrics_freq) AS NUMERIC) AS stdev,
    CAST(STDDEV(metrics_freq)/SQRT(COUNT(user_id)) AS NUMERIC) AS standard_err_deviation

  FROM(
    SELECT user_id, experiment_group,
      metrics/((EXTRACT(days from ('2014-06-30' - occurred_at::timestamp))*24+
      EXTRACT(hours from ('2014-06-30' - occurred_at::timestamp)))/
      ((EXTRACT(days from ('2014-06-30' - '2014-06-01'::timestamp)))*24)) AS metrics_freq
    FROM(
      SELECT u.user_id, e.experiment_group, e.occurred_at,
        COUNT(CASE WHEN ev.event_type = 'engagement' THEN u.user_id ELSE NULL END) AS metrics
      FROM(
        SELECT user_id
          FROM tutorial.yammer_users
          WHERE created_at <= '2014-05-26') u
      JOIN tutorial.yammer_experiments e
      ON u.user_id = e.user_id
      JOIN tutorial.yammer_events ev
      ON u.user_id = ev.user_id
      WHERE ev.occurred_at > '2014-06-01'
        AND ev.occurred_at < '2014-06-30'
        AND e.experiment = 'publisher_update'
      GROUP BY 1,2,3
      ORDER BY metrics, experiment_group) c
      ORDER BY metrics_freq, experiment_group) d
  GROUP BY 1
  ORDER BY 1) s
```