A/B testing at Uber
A BYOM (Bring Your Own Metrics) Platform

Milène Darnis
Product Manager, Experimentation
1. Overview of A/B testing at Uber
2. Decoupling experimentation events from business metrics
3. Extending the platform
4. Future work
5. Conclusion
Overview of A/B testing at Uber

Decoupling experimentation events from business metrics

Extending the platform

Future work

Conclusion
Uber’s mission is to ignite opportunity by setting the world in motion.
Uber’s mission is to ignite opportunity by setting the world in motion.

In order to do this, we run thousands of experiments every year.
Where we experiment
Where we experiment

- Backend
  - Python, Java, Go

- Mobile
  - iOS, Android
Where we experiment

Backend
Python, Java, Go

Mobile
iOS, Android
Where we experiment

Backend
Python, Java, Go

Mobile
iOS, Android
What we experiment on

User facing features
What we experiment on

User facing features
What we experiment on

User facing features

Bug fixes
Analyzing A/B tests, a seemingly simple problem...
...but not at Uber’s scale and pace
BEFORE We computed all the metrics for all experiments

8
Pipelines
of 3,000+ lines of SQL

1-2
Runs per day

60%
Unused metrics
BEFORE

To onboard new metrics
BEFORE  To onboard new metrics

For each new metric:

1. Metric request
2. Implementation in code
3. Experimentation engineers need to understand the metric
4. Validation of the metric between the two teams
5. Metric is used for experimentation
BEFORE  We had other problems too

People doing analysis themselves

- Duplicate efforts across teams
- Use of slightly different methodologies

- Waste of time and resources
- Incomprehension
BEFORE We had other problems too

People doing analysis themselves

- Duplicate efforts across teams
- Use of slightly different methodologies
- Our team productivity suffered

- Waste of time and resources
- Incomprehension
- Less time for more interesting problems
Overview of A/B testing at Uber

Decoupling experimentation events from business metrics

Extending the platform

Future work

Conclusion
Changing the paradigm

Experimentation team:
- Experimentation data
- Who is seeing which experiment when

Other teams:
- Team metrics
- Self-serve creation and edit of metrics
Experimentation events: the need for small data

Step 1
15B records

Step 2
5B records

Step 3
250M records
Logging of events happens automatically

Mobile or backend SDK

get_treatment_result
(experiment_name,
user_id=user_id)

control_group

Experimentation service
Logging of events happens automatically

Mobile or backend SDK

```python
get_treatment_result(experiment_name, user_id=user_id)
```

Experimentation service

control_group

Kafka

- timestamp
- experiment_name
- user_id
- treatment_group

→ Approx. 15 billion records emitted daily.
→ 1.5 TB of data.
Step 2: Deduplication using Spark

Kafka ➔ HDFS ➔ Hive

→ Down to approx. 5 billion records daily
Partitioned by date

30 minutes
Spark Jobs average runtime
Still a lot of records

<table>
<thead>
<tr>
<th>TIMESTAMP</th>
<th>USER_ID</th>
<th>EXPERIMENT_NAME</th>
<th>TREATMENT_GROUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018-09-12 10:12:45</td>
<td>42986</td>
<td>experiment_abc</td>
<td>treatment_1</td>
</tr>
<tr>
<td>2018-09-12 11:13:27</td>
<td>32989</td>
<td>experiment_abc</td>
<td>treatment_2</td>
</tr>
<tr>
<td>2018-09-12 11:17:45</td>
<td>98829</td>
<td>experiment_abc</td>
<td>control</td>
</tr>
<tr>
<td>2018-09-12 14:45:34</td>
<td>98397</td>
<td>experiment_abc</td>
<td>control</td>
</tr>
<tr>
<td>2018-09-12 14:38:28</td>
<td>42986</td>
<td>experiment_abc</td>
<td>treatment_1</td>
</tr>
</tbody>
</table>
Step 3: only keeping the relevant data

1st user

2nd user

Entered the experiment
### Step 3: only keeping the relevant data

Entered the experiment

<table>
<thead>
<tr>
<th>ENTRY_DATE</th>
<th>EXIT_DATE</th>
<th>USER_ID</th>
<th>EXPERIMENT_NAME*</th>
<th>TREATMENT_GROUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018/09/01</td>
<td>NULL</td>
<td>1</td>
<td>experiment_abc</td>
<td>treatment</td>
</tr>
<tr>
<td>2018/09/01</td>
<td>2018/09/30</td>
<td>2</td>
<td>experiment_abc</td>
<td>treatment</td>
</tr>
<tr>
<td>2018/09/30</td>
<td>NULL</td>
<td>2</td>
<td>experiment_abc</td>
<td>control</td>
</tr>
</tbody>
</table>
Step 3: only keeping the relevant data

<table>
<thead>
<tr>
<th>ENTRY_DATE</th>
<th>EXIT_DATE</th>
<th>USER_ID</th>
<th>EXPERIMENT_NAME*</th>
<th>TREATMENT_GROUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018/09/01</td>
<td>NULL</td>
<td>1</td>
<td>experiment_abc</td>
<td>treatment</td>
</tr>
<tr>
<td>2018/09/01</td>
<td>2018/09/30</td>
<td>2</td>
<td>experiment_abc</td>
<td>treatment</td>
</tr>
<tr>
<td>2018/09/30</td>
<td>NULL</td>
<td>2</td>
<td>experiment_abc</td>
<td>control</td>
</tr>
</tbody>
</table>

Now 250 million daily records
Step 3: only keeping the relevant data

Now **250 million** daily records

* PARTITION COLUMN
Takeaway #1

Whenever possible, present the data in a format that is easy to consume, not easy to compute.
Letting people define their own experimentation metrics

```
-- Start writing your metric SQL using the template included below

SELECT
  {{experiment_user_id}},
  {{experiment_treatment}},
  <your formula> AS 'metric_value'
FROM
  {{experimentCohort}}
LEFT JOIN
  <your dataset> a
ON
  AND {{experiment_user_id}} = a.user_id
-- TODO: Filter the metric table based on dates passed in from experimentation
AND a.datestr >= {{measureStart}}
AND a.datestr < {{measureEnd}}
GROUP BY
  {{experiment_user_id}}, {{experiment_treatment}}
```

Supported:

- Hive
- Presto
- Vertica
Tying it all together

Experiment
- my_experiment_L1

Morpheus Segment
- 1 item selected

Control Group
- control

Treatment Groups
- 1 treatment selected

Metrics
- 3 metrics selected
Tying it all together

3 metrics selected

**SQL | metric_1**

```sql
SELECT Treatment_group_key, user_id, denominator_value, numerator_value FROM (SELECT exp.user_id, exp.treatment_group_key AS treatment_group_key, begin_effective_timestamp FROM
JOIN exp.user_id = e.driver_id AND e.start_timestamp_utc > NOW() AND e.end_timestamp_utc < NOW() AND a.datestr = '1970-01-01' GROUP BY exp.user_id, exp.treatment_group_key) AS a
```
Tying it all together

3 metrics selected

**Experiment**
my_experiment_1

**Morpheus Segment**
1 item selected

**Control Group**
control

**Treatment Groups**
1 treatment selected

**Metrics**
3 metrics selected

---

### SQL | metric_1

```sql
SELECT Treatment_group_key, user_uuid, numerator_value, denominator_value FROM (SELECT exp.user_uuid, exp.treatment_group_key, exp.start_timestamp_utc, exp.end_timestamp_utc FROM (SELECT user_uuid, treatment_group_key, start_timestamp_utc, end_timestamp_utc FROM experiment) AS exp) AS all_treatment
JOIN (SELECT user_uuid, treatment_group_key FROM (SELECT user_uuid, treatment_group_key FROM experiment) AS all_treatment) AS treatment
ON exp.user_uuid = e.user_uuid
AND exp.start_timestamp_utc >= treatment_group_key
GROUP BY exp.user_uuid, exp.treatment_group_key
```

---

### metric_1

<table>
<thead>
<tr>
<th>Treatment</th>
<th>LRT (Mean)</th>
</tr>
</thead>
<tbody>
<tr>
<td>control</td>
<td>0.587</td>
</tr>
<tr>
<td>enabled</td>
<td>-8.07%</td>
</tr>
</tbody>
</table>

---

### metric_2

<table>
<thead>
<tr>
<th>Treatment</th>
<th>LRT (Mean)</th>
</tr>
</thead>
<tbody>
<tr>
<td>control</td>
<td>0.159</td>
</tr>
<tr>
<td>enabled</td>
<td>-23.48%</td>
</tr>
</tbody>
</table>

---

### metric_3

<table>
<thead>
<tr>
<th>Treatment</th>
<th>LRT (Mean)</th>
</tr>
</thead>
<tbody>
<tr>
<td>control</td>
<td>25.27</td>
</tr>
<tr>
<td>enabled</td>
<td>+17.92%</td>
</tr>
</tbody>
</table>
Everything happens asynchronously

- **2.9 minutes**
  Average runtime of a metric

- **5.75 metrics**
  Average number of metrics per report
## Before vs. After

<table>
<thead>
<tr>
<th></th>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freshness</td>
<td>24 hours</td>
<td>4 hours</td>
</tr>
<tr>
<td>Metrics selection</td>
<td>Static set</td>
<td>BYOM</td>
</tr>
<tr>
<td>Metrics onboarding</td>
<td>Engineering work required</td>
<td>No engineering work needed</td>
</tr>
<tr>
<td>Resources</td>
<td>Computation of irrelevant metrics</td>
<td>Computation of what’s needed only</td>
</tr>
<tr>
<td>Speed</td>
<td>Faster</td>
<td>Slower</td>
</tr>
</tbody>
</table>
User delight, making the wait pleasant
User delight, making the wait pleasant
Takeaway #2

Balance speed versus flexibility based on user needs.
Overview of A/B testing at Uber

Decoupling experimentation events from business metrics

Extending the platform

Future work

Conclusion
It’s now easy to add new features

Data Engineering

Data Science
Data engineering: slicing and dicing of results

<table>
<thead>
<tr>
<th>ENTRY_DATE</th>
<th>EXIT_DATE</th>
<th>USER_ID</th>
<th>EXPERIMENT_NAME</th>
<th>TREATMENT_GROUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018/09/01</td>
<td>NULL</td>
<td>1</td>
<td>experiment_abc</td>
<td>treatment_1</td>
</tr>
<tr>
<td>2018/09/01</td>
<td>2018/09/30</td>
<td>2</td>
<td>experiment_abc</td>
<td>treatment_2</td>
</tr>
</tbody>
</table>
Data engineering: slicing and dicing of results

<table>
<thead>
<tr>
<th>ENTRY_DATE</th>
<th>EXIT_DATE</th>
<th>USER_ID</th>
<th>EXPERIMENT_NAME</th>
<th>TREATMENT_GROUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018/09/01</td>
<td>NULL</td>
<td>1</td>
<td>experiment_abc</td>
<td>treatment_1</td>
</tr>
<tr>
<td>2018/09/01</td>
<td>2018/09/30</td>
<td>2</td>
<td>experiment_abc</td>
<td>treatment_2</td>
</tr>
</tbody>
</table>

join on user_id = user_id and to_date(entry_date) = date

<table>
<thead>
<tr>
<th>DATE</th>
<th>USER_ID</th>
<th>CITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018/09/01</td>
<td>1</td>
<td>Veracruz</td>
</tr>
<tr>
<td>2018/09/01</td>
<td>2</td>
<td>Medellin</td>
</tr>
</tbody>
</table>

= Enriched experimentation logs
Segmented results for a fraction of time

<table>
<thead>
<tr>
<th>City</th>
<th>Value</th>
<th>Change (%)</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leon</td>
<td>964</td>
<td>-3.843%</td>
<td>Not Significant</td>
</tr>
<tr>
<td>Chihuahua</td>
<td>756</td>
<td>+0.706%</td>
<td>Not Significant</td>
</tr>
<tr>
<td>Medellin</td>
<td>657</td>
<td>+35.97%</td>
<td>Not Significant</td>
</tr>
<tr>
<td>Cali</td>
<td>561</td>
<td>+127.90%</td>
<td>Significant</td>
</tr>
<tr>
<td>Toluca</td>
<td>450</td>
<td>-4.786%</td>
<td>Not Significant</td>
</tr>
<tr>
<td>Veracruz</td>
<td>314</td>
<td>+21.80%</td>
<td>Not Significant</td>
</tr>
</tbody>
</table>

Additional run time: **+35%** on average
Data Science innovation

Outlier removal
Removes irregularities in the data, enables more robust results
Data Science innovation

Outlier removal
Removes irregularities in the data, enables more robust results

Pre-existing bias detection and correction
Using CUPED method to adjust results and increase statistical power

Warning: Pre-existing Bias
We detected significant pre-existing bias in at least one metric.
Learn more about these checks →
Takeaway #3

Instead of trying to do it all, do what you are great at and build an infrastructure that lets others add the missing pieces.
1. Overview of A/B testing at Uber
2. Decoupling experimentation events from business metrics
3. Extending the platform
4. Future work
5. Conclusion
What looking for the right metric looks like

Metrics governance
Metrics governance

What looking for the right metric looks like
A vast but organized catalog of metrics

Filter metrics by:

IMPORTANCE
High

TEAM
Uber for Business

TYPE

- DECISION METRICS
  Used to conclude an experiment

- GUARDRAIL METRICS
  Making sure experiments don't introduce regressions
Self-serve metrics != metrics as an afterthought

### Hypothesis

Instructions: Add one row per metric that you will monitor as a part of this experiment, as well as the amount by which you expect to move each metric (no change is okay). This should include all of your KPIs associated with your experiment design. Note: Estimated impact is a requirement, even though you will probably not hit your hypothesized impact perfectly.

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Relative Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Growth</td>
<td>+5%</td>
</tr>
<tr>
<td>Retention</td>
<td>+2%</td>
</tr>
</tbody>
</table>
Self-serve metrics != metrics as an afterthought

**Hypothesis**
Instructions: Add one row per metric that you will monitor as a part of this experiment, as well as the amount by which you expect to move each metric ("no change" is OK). This should include all of your KPIs associated with your experiment design. Note: Estimated impact is a requirement, even though you will probably not hit your hypothesized impact perfectly.

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Relative Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Growth</td>
<td>+5%</td>
</tr>
<tr>
<td>Retention</td>
<td>+2%</td>
</tr>
</tbody>
</table>

Run experiment → Find a metric that moves in the right direction → Launch
Upcoming work

Outline Your Hypothesis

Objective*
What will be changing

Expected Start and End Date
Start Date    End Date

Primary Metrics*
Metric    MDE    Time frame

Secondary Metrics*
Metric    MDE    Time frame

+Add Metric
1. Overview of A/B testing at Uber
2. Decoupling experimentation events from business metrics
3. Extending the platform
4. Future work
5. Conclusion
We’re hiring!

- Data Scientists
  All levels

- Software Engineers
  All levels

>> Check uber.com/careers
Acknowledgments

Colin Reid
David Schnurr
Egor Gryaznov
Spencer Lin
Suman Bhattacharya
Tianxia Zhou
If you only remember three things
If you only remember three things

01. Easy to consume data > Easy to compute data
If you only remember three things

01. Easy to consume data > Easy to compute data

02. Speed ↔ Flexibility tradeoff
If you only remember three things

01. Easy to consume data > Easy to compute data

02. Speed ↔ Flexibility tradeoff

03. Leverage your strengths, build products that users can contribute to
Thank you!

Milène Darnis <milene@uber.com>