Effectively Combining the Event and Aggregate Summary Reports from the Privacy Sandbox Attribution Reporting API for Improving Ad-Measurement Fidelity

Tong Geng, Matt Dawson, Harikesh Nair

Glossary

This glossary describes terminology that will be used throughout this post.

- Ad-events or ad-interactions: a user action on an advertisement that was shown to them, typically,
    - Ad-click: when the user clicks on the ad shown
    - Ad-view: when the user views the ad shown (sometimes for a required duration)
- Ad-group: A grouping of ads that share similar targeting settings
- Ad-tech: A technology company which serves online ads and which helps advertisers to plan, buy, measure and/or optimize their ad-campaigns
- Aggregate Summary Reports: One part of the Attribution Reporting API which provides aggregated summary reports for clicks and views in a privacy-preserving way, without third-party cookies
- Application Programming Interface (or API): A set of software-defined protocols
- Attribution Reporting API (or ARA): A pair of new API that are proposed as part of the Privacy Sandbox which facilitates ad measurement in a privacy-preserving way, without third-party cookies
- Campaign: A set of ad-groups (ads, keywords, and bids) that share a budget, location targeting, and other settings. Campaigns are often used to organize categories of products or services that an advertiser offers
- Click-through conversions: conversions which follow an ad-click over a specific duration
- Conversion attribution: the act of assigning conversion activity to the appropriate prior ad-interaction(s)
- Conversion Type: a description of the conversion, such as a purchase, page view, subscription, etc.
- Conversion Value: The value to the advertiser of a conversion, typically expressed in currency units
- Conversion: a user action which advertisers care about, such as a visit or purchase on a website, which ad-techs report and optimize for on behalf of their advertisers
- Event-Level Reports: One part of ARA which provides event-level reports for clicks and views in a privacy-preserving way, without third-party cookies
- Metadata: additional information related to ad-interactions or conversions
- Privacy-enhancing technologies (or PETs): A set of technologies that facilitate digital advertising in a privacy preserving way, of which the ARA is an example
- Third-party cookies (or 3PCs): A set of digital identifiers that maintain a persistent identity state for a browser across visited sites
- View-through conversions: conversions which follow an ad-view over a specific duration

In addition to the terms above, some of the descriptions in this post reference technical topics from statistics:

- Aggregation: The process of combining granular pieces of data into larger pieces, typically through the use of an aggregating function such as a “sum”
- Differential Privacy (or DP): a framework for extracting useful information from a dataset, while providing protection against leakage of information corresponding to individuals in the dataset; often this is achieved by perturbing the information being extracted. When the perturbation is applied to statistics involving the entire dataset, this is also referred to as Central-DP.
- Expected value: A concept from statistics which describes the “average” outcome of a random quantity
- Laplace probability distribution: A description of a particular type of random outcome in which the most likely outcomes lie near the expected value and the likelihood decreases exponentially away from the expected value
- Local Differential Privacy (or Local-DP): A special case of “Differential Privacy” above, where noise is added to each individual data point to protect data privacy
- Privacy Budget or “ϵ” (see “sequential composition”): [not to be confused with the unrelated Privacy Sandbox Technology of the same name] A parameter which controls the degree of privacy a differentially private mechanism can guarantee. For the Laplace mechanism (which is used for Aggregate Summary Reports in the ARA), if multiple queries are used on the same dataset, this privacy budget must be divided across each of them.
- Randomized response: A statistical mechanism by which observations are randomly perturbed in order to satisfy a privacy requirement
- Slice: A particular instance of aggregation such as at the ad-campaign or ad-campaign-country level
- Truncation: The act of transforming a random variable so that its realizations below (analogously above) a predefined threshold are retained, and those above (analogously below) are dropped. When applied to conversions, truncation implies information loss because conversions following an ad-interaction that are beyond a predefined threshold are not observed
- Variance: A concept from statistics describing the deviation of a random quantity from its expected value

Summary

This post describes a set of methodologies developed by Google Ads for how ad-techs can combine data from the Event-level Reports and the Aggregate Summary Reports from the Attribution Reporting API (ARA) to enhance their utility for advertising use-cases.

Who this post is for and technical depth needed

This post describes details of the ARA API and sketches new statistical methodologies for how to merge data from both API reports. The post describes the ideas at a high level and does not get into specificities of the methodologies in detail. Some level of statistical knowledge and intuition on the part of the reader is presumed.

If you work in digital advertising and are interested in learning how to blend and merge the data from the Event-level and Aggregate Summary Reports for ads measurement, you will learn key ideas about how the API works in tandem and how the data from one can augment the other.

Introduction

As the advertising ecosystem makes a significant pivot towards improved ways of protecting user privacy, the use of privacy-enhancing technologies (PETs), such as the ARA API proposed by the Privacy Sandbox team, will increasingly become relevant for ads measurement. Measurement-focused PETs facilitate ads measurement while protecting user's cross-site and cross-app identities from being revealed to ad-techs, advertisers, publishers and other entities (henceforth, collectively referred to under the umbrella-term "ad-techs"). Though the specificities of implementation differ, a common feature of PETs is to limit the information content of campaign performance data, by leveraging some combination of anonymization, aggregation, information truncation and noise infusion to the data before it is released to ad-techs. By this process, these technologies seek to allow measurement with privacy guarantees while continuing to support advertising use-cases in a sustainable way.

For ad-techs who consume these data and utilize them for various ads use-cases, this new environment presents several new data and modeling-related issues that were previously not present with third-party cookies (3PCs). The new issues arise from the changes induced by the API data for privacy. Due to anonymization, aggregation, information truncation and addition of noise, the data reported to ad-techs by the API will deviate from the conversion data measured by 3rd-party cookies today (henceforth 3PC conversions). For ARA, the event-level reports that ad-techs will obtain from the Event-level Reports will have statistical noise added; with attributed conversions that are truncated within pre-specified limits; reported only with limited conversion metadata over one, two or three time window(s). The summary reports that ad-techs will obtain from the Aggregate Summary Reports will be reported only at the aggregate slice level, will have statistical noise added, and will have conversions that are truncated within pre-specified limits. See this explainer from Chrome for more details. As a consequence, all ad-techs consuming data from the API will need to develop ways to work with such new data structures for their advertising use-cases.

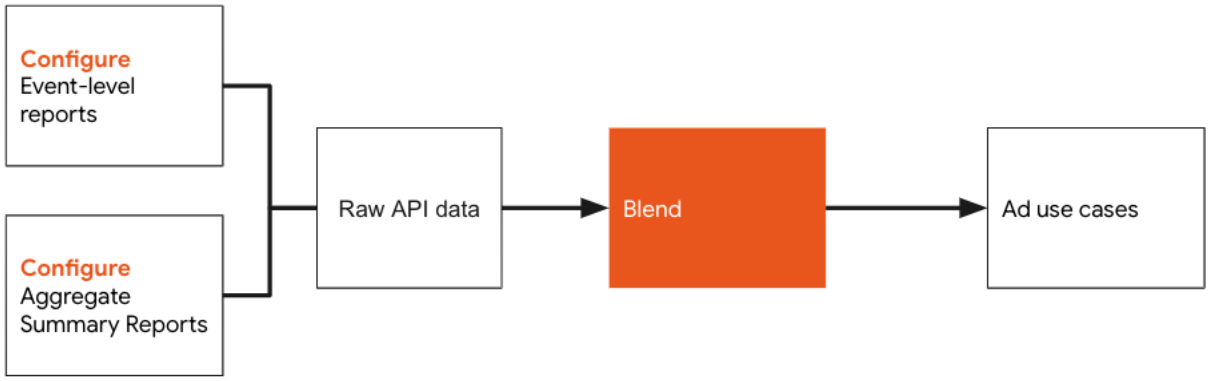
This post outlines a set of methodologies we developed in Google Ads for this purpose. Specifically, we outline ways by which the data from the Event-level and Aggregate Summary Reports can be merged together so as to facilitate ad-measurement with higher utility.

There are a variety of ways of combining and merging the data sets and different ad-techs will likely take different paths to achieve this. This post reflects the way Google Ads has approached the problem and outlines some of our thinking and some progress we have made. As part of our preparation for 3PC deprecation in 2024, we are incorporating these methodologies into our advertising stack with the API so as to provide high quality ad-tech services such as reporting and bidding for our advertisers. While comprehensive evaluations of these methodologies is out of scope for this post, we hope that by describing these methodologies and outlining our investments in this area, we:

- Provide better insights to our partners and to the broader ecosystem as to how the API can be utilized for improving advertising measurement post 3PC deprecation, and,
- Spur further innovation in the ecosystem on how the API can be leveraged to implement privacy preserving advertising measurement with higher utility

The Essence of Our Approach

As an ad-tech, Google Ads seeks to develop ways to effectively use the ARA API for ads-measurement. This requires reasoning about how we can obtain better quality data from the API and how the quality of these data can be improved once they arrive at our servers. The essence of our approach can be summarized in the below picture:



That is, at a high-level, we follow a two step approach:

1. **Configure** the API intelligently to get two separate raw API data feeds - one corresponding to the Event-level Reports, and one corresponding to the Aggregate Summary Reports
2. **Blend** the raw data from step one intelligently to obtain a combined dataset, which is then used for downstream ads-use-cases

Both steps are important. A key message here is that all ad-techs working with the API will benefit from paying special attention to both configuring and blending the API data. Implementing configuration and blending at scale requires both methodological and infrastructure innovation.

The API provides ad-techs a significant amount of flexibility in configuring the API and ad-techs may leverage this feature to obtain the right data according to their business needs. The API allows the ad-tech to decide on several parameters (henceforth “configs”) indexing how the API collects and processes events and conversions on their behalf. In turn, the choice of these configs affect both the nature and the quality of the data reported by the API back to the ad-tech. Examples of such configs for the Event-level Reports include the duration of time-windows over which conversions will be aggregated and the type of conversions that will be tracked by the API. Examples of such configs for the Aggregate Summary Reports include deciding the limits at which conversions reported in the API will be truncated, and deciding the allocation of privacy budgets across reports pulled from the API. While configurations can be picked naively (and defaults are available), we have found that ad-techs will benefit from deciding them in an intelligent and data-driven manner that makes sense for their book of business and advertiser profile. The details of intelligent and data-driven configuration are beyond the scope of this post and hence we abstract away from it in what follows below. The reason we highlight it explicitly here is so the reader can note the interaction between steps 1 and 2: how the API is configured determines the nature and quality of the raw data the ad-tech obtains from the API, which in turn affects its ability to blend them in step 2.

Step 2, which is the main subject of this post, involves blending the two data streams from step 1. Although the figure above may suggest that ad-techs always have to use the two reports in API in conjunction, this is not the only way. It is possible to use one report for some use-cases (e.g., bidding) while using the other report for other use-cases (e.g., reporting). Nevertheless, our view is that using them in combination and creating a unified view of the data for all use-cases has several advantages. There are at least two benefits of this approach:

1. [*Complementarity*] First, the two reports represent two different lossy views of the same underlying data generating process. So, using them in combination can complement each other, allowing one to address the loss in information in the other.
2. [*Consistency*] Second, a unified view ensures a consistent data input for all ads use-cases that rely on the API, thereby minimizing possible misalignment amongst them (for example, suppose the Event-level Reports alone are used for the automated bidding use-case, and the Aggregate Summary Reports alone are used for the

reporting use-case; it is possible that bidding performance and campaign reporting become misaligned).

Once we decide to use the reports in combination, the next task is to develop a principled methodology for how to effectively implement the blending. There are many ways in which the API data can be blended, and the details of *how* these reports are blended may vary from ad-tech to ad-tech.

Google Ad's approach to merging is to develop an **eventified and modeled data log** which reflects the combined information content of both reports. By that, we mean we develop a log whose rows represent ad-events and columns represent the outcomes (conversion counts and conversion values) attributed to these ad-events. To do this, we take the data from the Event-level Reports, and leverage custom post-processing techniques and aggregate information from the Aggregate Summary Reports to obtain an event-level view which is, as a consequence, more accurate than the original Event-level Reports. **It is important to note that, because of the privacy protection nature of the API, we can never recover the exact same event-level 3PC conversion data, but this approach may produce a more useful event-level data log for Ad use-cases**. We call this process "**eventification**".

Eventification has several advantages. First, though utilizing a different measurement technology altogether, the eventified log is similar in structure to how event-level data would look with third party cookies. Therefore, once the log is created, it can be inserted into existing data pipelines and other modeling infrastructure built for 3PC data with little changes. This reduces technical debt and makes it easier to transition to a world with the ARA. Second, the same eventified log can be used for many use-cases, including the two dominant use-cases of reporting and bidding. For reporting, the eventified log can be aggregated as appropriate to the slice for which reporting is required, say at the campaign level. For bidding, what is key is the ability to train machine learning models with conversions as the label and with ad-event characteristics as features. This requires as input a training data-set whose units are ad-events and attributed conversion combinations. This is exactly what the eventified log provides. Building both the reporting and the bidding off the same log thus reduces complexity, and automatically guarantees consistency across use-cases.

With this high level context in place, the rest of the post proceeds as follows. In part 1, we explain how the API works and describe the data structures they induce. Understanding these data structures is important for reasoning about how these data can be blended. Readers unfamiliar with the API should invest in part 1 so as to develop a clear understanding of the API data structures before moving to part 2. Readers familiar with the API can skip or simply skim part 1 and go directly to part 2 which describes the methodologies. Part 3 of the post provides a simple illustrative example that explicates the methodologies and unpack how they work. Part 4 concludes with some discussion of some open questions and future work.

Part 1: How the APIs Work and their Data-Structures

It is important to understand that the Event-level and Aggregate Summary Reports represent two different views of the same underlying data. The nature of the data generated by both is a function of how each transforms the same underlying data to preserve privacy. Hence, the first task is to obtain a clear understanding of these transformations. As the full details of the API are complex, our approach will be to explain them one step at a time. Our pedagogical approach in this section will be to first describe the data structure which ad-techs can expect to receive from each API, while temporarily ignoring the details of how the data are generated. Then, we consider two crucial aspects of working with the API data: conversion truncation and noise considerations, and how these aspects differ for each of the API. Finally, we dive into the full details of the API. Our hope is to leave the reader with a thorough understanding of the API by the end of the section, but outlined via a series of graded explanations. Interested ad-techs may also use Noise Lab and Attribution Reporting Simulation Library provided by Chrome to experiment with the Aggregate Summary Reports and understand the impact of the API.

As we do this, we will also comment on how their strengths complement each other. This sets up Part 2, which will describe how we can leverage these strengths so that the combined use of the reports provides better measurement fidelity than using either report in isolation.

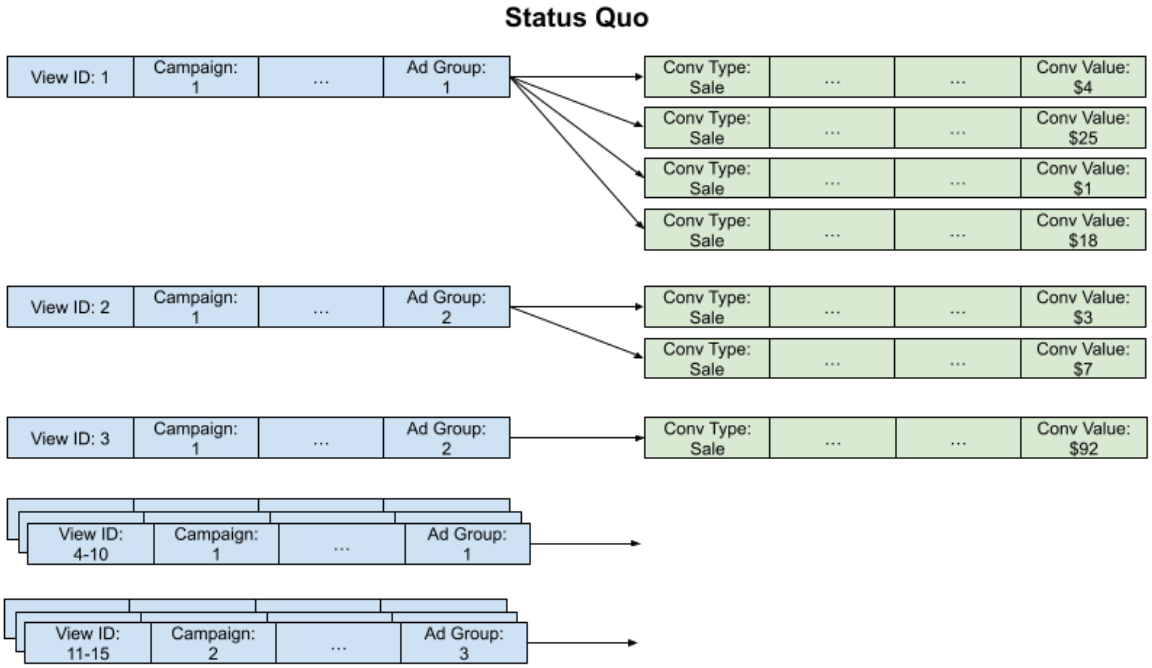
Event-level Reports

The term “Event” in Event-level Reports corresponds to ad-events. That is, the granularity of an Event-level Report is an ad-interaction, namely a click or a view. Along with the ad-interaction, the API returns some information about any conversions that may (or may not) have happened within a predefined duration after the ad-interaction.

A canonical use-case for the Event-level Reports is model training; this type of information is particularly useful when the goal is to predict conversions, conversion rates or conversion values, conditional on the features of an ad-interaction. Predicting these is a key input to automated bidding models, because it serves as a (stochastic) signal to the bidding algorithm which events are likely to lead to conversion that the advertiser cares about. The bidding algorithm decides on its bid algorithmically by taking the signal as an input, so the quality of the model prediction in turn affects the quality of the bidding optimization directly. To protect user privacy, the API does not return the event-level data with full fidelity. Rather, a small proportion of ad-interactions are randomly chosen by the browser/platform to be assigned random conversion metadata. In addition, there are limits on how much metadata can be extracted from the conversions. Details on these topics follow.

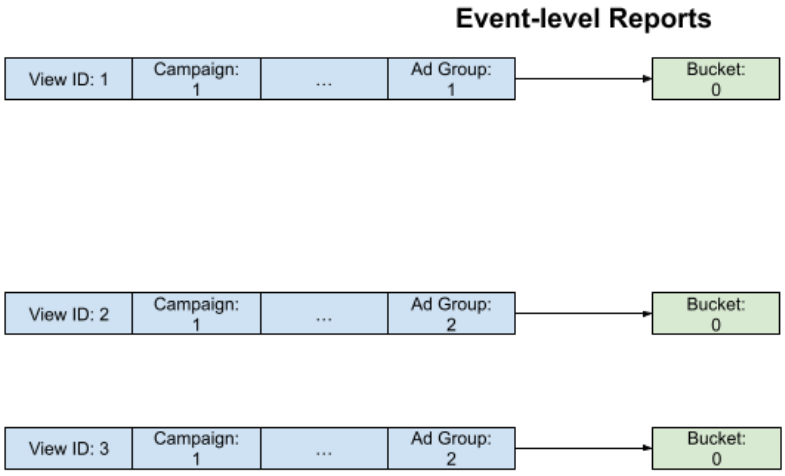
Data Format

Data coming from the Event-level Reports is “event-level” in the sense that we receive a record for each ad-interaction, paired with metadata corresponding to the subsequent conversion(s). Before we outline the details of the Event-level Reports, let us first explore the status quo, or 3PC conversion data that forms the input into the API.



In this figure, the blue boxes on the left represent ad-interactions (in this case, ad-views), while the green boxes on the right represent conversions that may or may not occur following the ad-view. There are 15 ad-events depicted (view IDs: 1-15). They are associated with several ad-event features such as which campaign and ad-group they correspond to. Looking at the blue boxes, there are 2 campaigns and 3 ad-groups. Conversions have their own metadata such as Conversion type (e.g., sale or purchase) and Conversion value (in $). If conversions occur, they are attributed to the corresponding ad-view. Each of the views and conversions has complete metadata, in the sense that all information is known about each piece. For example, the second view came from Campaign 1, ad-group 2, and resulted in 2 conversions, each of which were sale (purchase) events and totaling for $10 in value. There are also several views which are not associated with any conversions, as indicated by the lack of green boxes for views 4-15.

The Event-level Reports take as input all of this information, but only reports a transformed version of this data back to the ad-tech:



In this figure, we see that some conversions in the status quo have been truncated, and that the metadata is curtailed on the conversion side. For example, view ID: 1 truly had 4 attributed conversions, but the Event-level Reports report only 1. Also, looking at the green boxes, we do not have all the information about the conversions, but only that the conversion fell into "bucket 0." (the definition of a "bucket" is under the control of the ad-tech.) On the other hand, unlike conversions, on which metadata is curtailed, we see full metadata is retained on views. Notice also that, in this example, the API is correctly telling us where conversions did not happen - namely on views 4-15. This is a useful characteristic of the Event-level Reports, which we will leverage in some methodologies that we describe later in the post.

Conversion Truncation

Under the current specification of the Event-level Reports (see the API parameters section in Experiment with Attribution Reporting: Handbook), at most 3 conversions may be registered against an ad-click; and at most 1 conversion may be registered against an ad-view. In the figure above, this can be seen in "View ID: 1" where only 1 conversion was registered while in reality there were 4 conversions. Later, we will see that the Aggregate Summary reports are less susceptible to truncation, suggesting that blending the two reports can be a promising approach to "filling-in" such truncation gaps.

Noise Considerations

The data from Event-level Reports is granular. This is a useful property for accurate conversion attribution. But, to preserve privacy, the granular attribution is subject to noising. Statistically, this noising is achieved via a randomized response mechanism (a well-known statistical technique for this purpose). The randomized response mechanism implemented in the Event-level Reports involves a binary selection process with two distinct possibilities: (1) factual conversion activity is reported, or (2) random conversion activity is reported. For each ad-interaction, the API randomly selects which of these two options to use with some

probability. We will denote (1) and (2) above as the “true branch” and the “noised branch” respectively. The probability of any given ad-interaction landing on the noised branch is small.

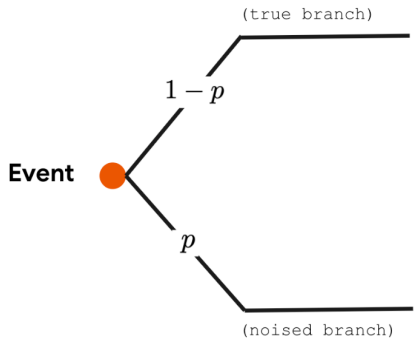
In More Detail

We now explain the specific implementation of some ideas outlined above in the Event-level Reports, paying special attention to how that induces a particular data-structure to the reports from the Event-level Reports. As mentioned above, the Event-level Reports use a randomized response model. Given a fixed value p, the Event-level Reports implement randomized response by randomly selecting each registered ad-event to be on the true or the noised branch with probability p (note: p is known to ad-techs, but whether an ad-event in the Event-level Reports is on the true branch or not is not known to ad-techs).

When the ad-event is on the true branch, the number of conversions reported by the Event-level Reports are truncated to the first *c* conversions and the metadata of such conversions are also limited – only some bits of metadata can be encoded and revealed to the ad-tech. Also, the exact conversion time is not shown and is bucketed into time-windows. The API will then send conversion reports containing such information to ad-techs.

When an ad-event is on the noised branch, the information of any true conversions or lack of conversions is suppressed and instead the API will randomly draw from a pool of possible configurations (a configuration is a specific realization of a conversion outcome) and produce conversion reports that are similar to those on the true branch. As a result, ad-techs cannot tell whether conversion reports sent from the API are on the true or the noised branch.

The figure below shows the mechanism pictorially.



Among the event’s true conversions, up to **c** are reported (truncation)

	Clicks	Views
p	0.24263%	0.00025%
c	3	1

A noised conversion config is drawn uniformly from a pool of possible configs (noising)

- Time Aggregation: Conversions will be reported across time windows (3 for clicks: 2/7/30 days; 1 for views: up to 30 days)
- Conversion Metadata: Conversion metadata will be restricted (clicks: 3 bits; views: 1 bit)

Aggregate Summary Reports

In contrast to the event-level data provided by the Event-level Reports, the Aggregate Summary Reports provide...aggregates. By “aggregates”, we mean a grouping of event-attributed conversion data that is aggregated to a slice level. The Aggregate Summary Reports allows the ad-tech to pre-define the slices over which they would like to learn about

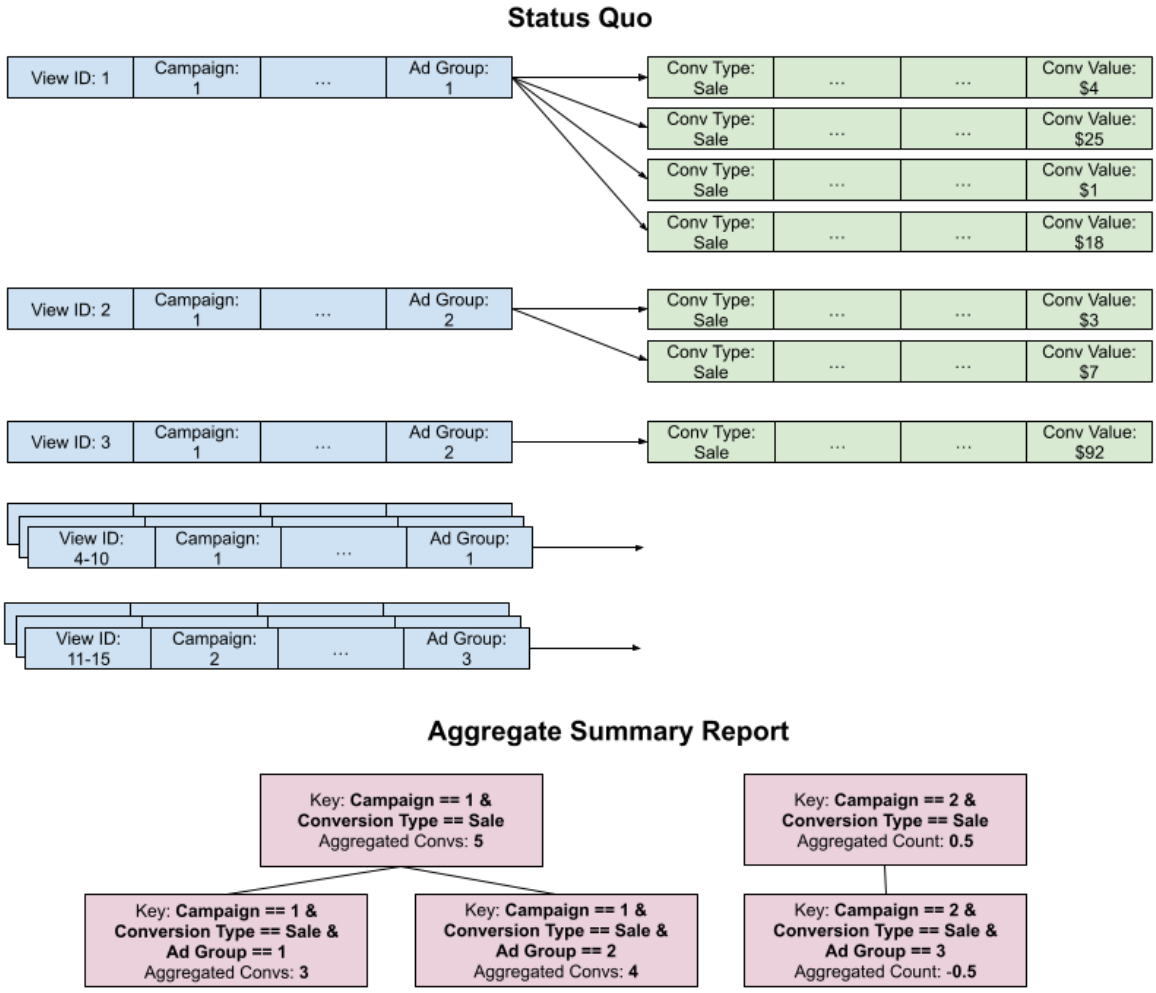
conversion activity. In plain language, these reports help answer questions like “How many conversions were there in a particular country?” or “What was the sum total of purchase values yesterday?”

This type of information is especially useful for reporting use-cases, by which advertisers can gain insights about their ad-campaigns and their business as a whole, as opposed to a click-by-click (or view-by-view) basis. But, using the Aggregate Summary Reports only for this is limiting. Later, we will also see that Aggregate Summary Report data can be used to enhance the data from the Event-level Reports, since they are ultimately two independent views of the same underlying data.

In the remainder of this section, we will follow the same expositional approach we used for the Event-level Reports, where we start with issues related to data formats. After that we describe the truncation and noise incorporated into the Aggregate Summary Reports for differential privacy.

Data Format

In contrast to the Event-level Reports, the Aggregate Summary Reports provide aggregated data to the ad-tech. Analogous to the description of the Event-level Reports, the figure below depicts an example of how 3PC conversion data would be represented by the Aggregate Summary Reports.



In this figure, the blue and green boxes at the top represent the same status quo data as before. The data generated by the Aggregate Summary Reports is shown in the pink boxes. We see while both reports provide a report generated from the same underlying data, the data structures are very different. In this case, the aggregate reports show only aggregate information split by Campaign and Ad-group. For example, the top pink box provides the aggregated count of conversions for Campaign 1 and of conversion type “Sale”. And the box below it provides a slightly more granular count, i.e., the total number of conversions for Campaign 1, of conversion type “Sale” and for ad-group 1. In what follows, we refer to the variables on the basis of which data can be aggregated as “keys” (e.g., Campaign ID, Ad Group, Conversion Type), a specific instantiation of those keys as “key-values” (Campaign ID == 1 , Ad Group == 1, Conversion Type == “Sale”) and the aggregate data corresponding a particular key-value as an “aggregate.” The data reported in each pink box in the above picture is an “aggregate” representing a specific combination of the event and conversion characteristics. The Aggregate Summary Reports provide datasets whose elemental granularity is at the level of aggregates.

There are three things to note about this data-structure. First, the aggregates reported by the Aggregate-API are not perfectly accurate. For instance, we see from the blue and green boxes that there are truly 7 conversions of type “sale” corresponding to Campaign 1, but the Aggregatable Summary Report shows 5 conversions in the top pink box. This is because what is reported in the pink boxes by the Aggregate Summary Reports, include statistical noise that is added to the counts for privacy reasons. More details on this follow below. Second, we see the data has been aggregated in a specific way: first, by Campaign, then by Conversion Type; and then by Ad-group. The exact specifics of the aggregates and how the aggregates should be organized is under the control of the ad-tech - the ad-tech can choose to aggregate the data in a way that is best for its business. More details on this are also described later. Finally, note the Aggregate Summary Reports can report a result even in cases where there were no factual conversions. The two pink boxes on the right report some conversions for Campaign 2, when in reality there were none. Again, this is due to the noising mechanism (described in more detail below).

Conversion Truncation

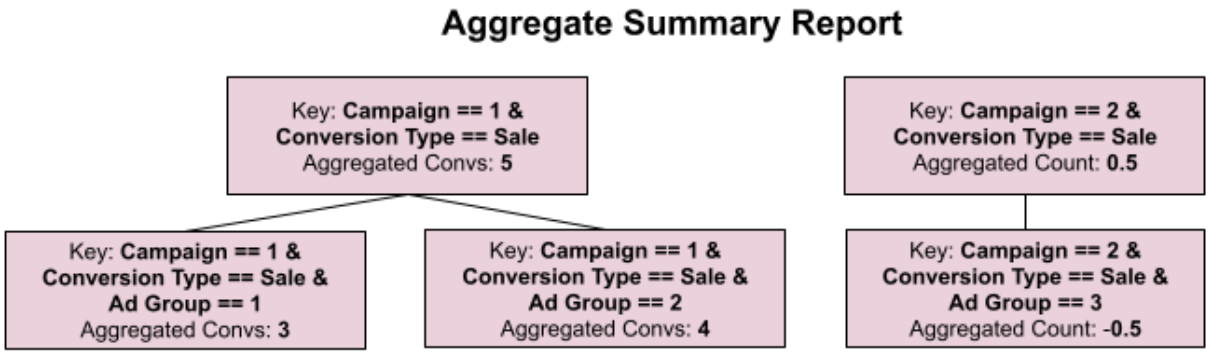
As is the case with the Event-level Reports, there are limits on how many contributions can be registered against a given ad-interaction for representation in the Aggregate Summary Reports. Specifically, the conversions following an ad-event will be truncated by these limits, and aggregates will be based on these truncated conversions. Therefore, the limits ultimately manifest in undersized aggregates if the conversion activity exceeds them. However, unlike the Event-level Reports which have fixed conversion limits, the limits in the Aggregate Summary Reports can be configured by the ad-tech by distributing the per-interaction L1 sensitivity parameter across potentially several conversions. In this way, the Aggregate Summary Reports offer a type of flexibility which can be used to capture the attribution picture better. This is another motivation for using both APIs in conjunction.

Noise Considerations

Noise is also added to the aggregate data, but the nature of the noise is different than in the Event-level Reports. Whereas the Event-level Reports use a local-DP model for adding noise, the differential privacy mechanism in the Aggregate Summary Reports is a central-DP model. Local-DP means that each ad-interaction is subject to noise; central-DP means that noise is only added after some aggregation has occurred. Also, unlike the noising mechanism in the Event-level Reports (randomized response), the Aggregate Summary Reports use a Laplace noising mechanism. This means that a random variate from a Laplace distribution is added to the aggregates. Specifically, the noising works as follows: attributed conversions following ad-events are truncated by the contribution bound; the truncated conversions are aggregated to a slice-level; and Laplace noise is added to that aggregate slice and reported. Specifics are described below.

In More Detail

The Aggregate Summary Reports require no specific structure nor a predefined set of keys over which to aggregate; those decisions are left up to the ad-tech. At Google Ads, we have found that a hierarchical structure for the aggregates allows for efficient and flexible use of the Aggregate Summary Reports. By "hierarchical", we mean that we arrange our aggregates in a tree-like structure, where parent "leaves" are split into children "leaves" with each additional key.



In the simple depiction above, we show a noisy realization of what Aggregate Summary Report data might look like in a hierarchical set up, splitting by two keys: Campaign ID and ad-group ID. A hierarchical structure allows for some flexibility to the ad-tech. When the number of conversions is low, the noise added by the Aggregate Summary Reports may overwhelm the true conversion count in its aggregates, so it may make sense for the ad-tech to collect aggregates at a coarser level, at the top of the hierarchy. On the other hand, when the number of conversions is high, the impact of noise may be lower relative to the true conversion count in the aggregates, and it makes more sense to collect aggregates at a more granular level at the lower part of the hierarchy. Thus, having a hierarchy provides the flexibility to the ad-tech to finetune the tradeoff between information and accuracy in its aggregates depending on its particular situation. We have also found that this particular structure can be helpful to efficiently combine information at multiple levels to improve the quality of aggregates across the tree as a whole. We discuss this more in the section on aggregate post-processing below.

In the following, we refer to a "slice" as a particular node, or combination of key-values, on our aggregate tree. In addition, we focus on the problem of obtaining accurate conversion counts, although similar techniques can be used when considering other types of aggregations (such as total conversion value).

The magnitude of the noise added to the aggregates is technically fixed, drawn from a Laplace distribution with mean 0 and standard deviation $\sqrt{2}L1/\epsilon$, where $L1$ and ϵ are respectively the sensitivity of the data and the privacy budget, decided by Chrome. In practice though, this

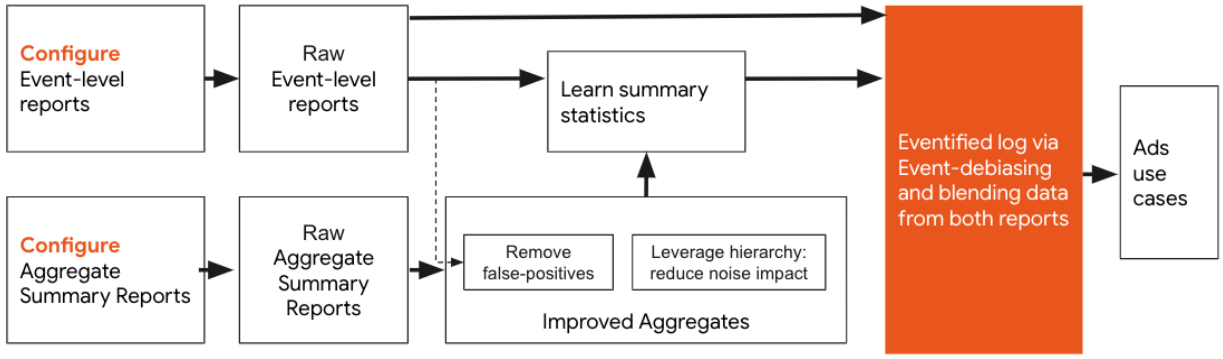
topic has many nuances and complexities and ad-techs may desire to optimize their privacy budget allocation (e.g., splitting of ϵ) across several queries.

Now that we understand the basic structure, strengths, and limitations of each of the API, we are ready to discuss our strategy for utilizing them. The guiding principle is that it will be important to use the API in **conjunction**. Details for how we achieve this follow.

Part 2: Combining Event-Level and Aggregate Summary Report Data

As mentioned in the introduction, our strategy for combining the Event-level Report and Aggregate Summary Report data involves eventification. Eventification is comprised of two steps:

1. Post-processing of Aggregate Summary Report data to obtain improved aggregates
2. Debiasing the Event-level Report data and blending it with the improved aggregates



This process is shown in the picture above. The ad-tech first chooses configs for both the API, and obtains the raw Event-level Report data and raw Aggregate Summary Report data from the API. The raw Event-level Report data is used to reduce the impact of noise in the aggregates. This process is shown in the lower part of the picture ad producing “Improved Aggregates”. This completes Step 1. The improved aggregates represent post-processed Aggregate Summary Report data. The post-processed Aggregate Summary Report data is used to learn a set of summary statistics, which represent key summaries of the underlying data generating process. These are used along with a “debiasing” procedure to account for the impact of noise and truncation in the Event-level Report data. This process generates an eventified log that blends the Event-level Report and Aggregate Summary Report data as shown in the top right part of the figure. This completes Step 2. Post-processing improves the fidelity of the data for ads-uses cases. **Note: post-processing and blending does not**

worsen the privacy guarantees from the API in the same way that any kind of post-processed differentially private data remains differentially private.

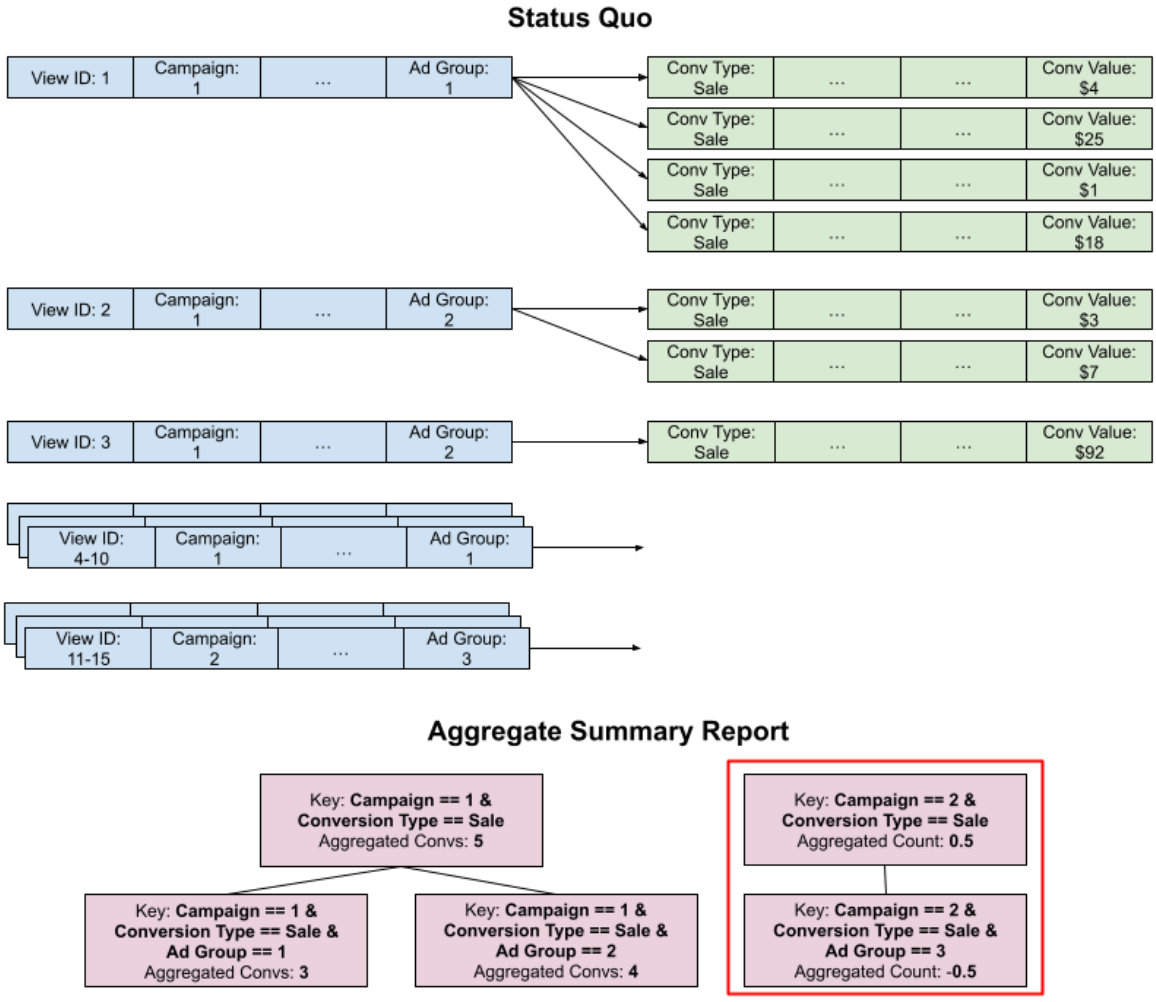
We briefly describe each of these steps below while more details and examples follow in Part 3.

Post-Processing of Aggregate Summary Reports

The first step to merging the Event-Level and Aggregate Summary Reports is to estimate better aggregates from the noisy data that the Aggregate Summary Reports sends the ad-tech. This step is essentially about reducing the impact of noise and is implemented by "post-processing" the Aggregate Summary Report data. Post-processing is motivated by the following fact: If we want to use aggregates as a form of calibration of the Event-level Report data, it is important to ensure that those aggregates are as accurate as possible. For obtaining improved aggregates from post-processing, we highlight two important components: dealing with "false positives" and leveraging the hierarchical nature of the aggregates.

"False Positives" in the Aggregate Summary Reports

One major way in which statistical noise in the Aggregate Summary Reports affects the quality of the data is via "false positives." These are aggregate slices where there are no conversions in reality, but for which the Aggregate Summary Reports add Laplace noise to 0 in its results. This results in slices that appear to have conversions when in actuality they do not.



False positives are a consequence of the structure of Aggregate Summary Reports, wherein the keys on the basis of which aggregation has to be done by the ARA are required to be pre-registered by the ad-tech *before* the data have been aggregated. Under this setup, ad-techs must register all aggregate keys which may have conversions, but are not guaranteed to, leading to a large number of key-values for which there may be false positives. In the above example, the two pink boxes on the right highlighted with a red box report some conversions for Campaign 2, when in reality there were none. These represent false positives. In these slices, there are 0 factual conversions, but the API returns $0 + L$, where L is a draw from the Laplace distribution above.

False positives represent error-filled aggregates and the first task is to identify them to the extent possible and drop them from the data. There are a variety of techniques possible to filter out false positives, but in this article we point out how the Event-level Reports can be leveraged for this task.

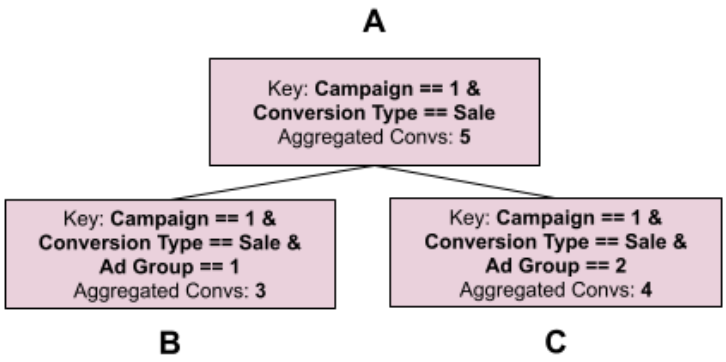
The intuition here is that if the Event-level Reports report no conversions for an ad-interaction, it is very likely that is indeed the case. To make this clear, we consider the CTC (click-through conversion) setting where each click can register 3 conversions, each with 3 bits of metadata, over 3 time windows. Now consider a click which in actuality resulted in a conversion, but was reported by the Event-level Reports as having no conversions. For this to happen, the click would need to be on the noised branch. After landing on the noised branch, the click would have to be randomly attributed to the single bucket with no conversions. Let num_states denote the number of conversion states in the noising mechanism. By working through the technical details of the specific noising mechanism (see here), we can show that num_states = 2925. Further, the probability the Event-level Reports would report a click as having no conversions, when in actuality it was associated with some conversions is p*1/num_states = 0.0024263 * 1/2925 = $8.3*10^{-7}$. VTCs (view-through conversions) have a similar property. This means that the probability the Event-level Reports would report an ad-interaction as having no conversions, when in actuality it was associated with some conversions, is tiny.

We can use this property – that the Event-level Reports give us a good signal of where conversions **did not happen** – to our advantage. To achieve this, we simply remove aggregate slices which the Event-level Reports suggest should not have conversions. This addresses the false positive problem in the Aggregate Summary Reports significantly, and is simple to implement. This is a specific example of how using the API in combination helps improve data quality.

Taking Advantage of the Hierarchy

The second post-processing procedure we discuss takes advantage of the hierarchical structure. It also leverages the fact that noise is added to all slices independently by the Aggregate Summary Reports. For example, suppose that our aggregate structure is as in our example where the top-level aggregates are Campaign ID-level and the bottom are ad-group-level. After noise has been applied, we essentially have multiple independent realizations of the campaign-level aggregate by taking a particular campaign and comparing it with the sum of conversions across all ad-groups in that campaign.

For example, consider the figure below. The Aggregate Summary Reports will report a noisy aggregate in the top box named “A” representing the number of conversions of Conversion type == “Sale” for Campaign == 1. The Aggregate Summary Reports will also report noisy aggregates for the two ad-groups (1 and 2) for Campaign == 1 and Conversion type == “Sale” in the bottom two boxes named “B” and “C”. We now note that both “A” and “B” + “C” are essentially measuring the same quantity. Thus we can combine the aggregates in “A” with that in “B” + “C” to improve the estimate of this quantity. Since the Laplace noise is added additively and is centered at 0, a simple strategy would be to take linear weighted averages of “A” and “B” + “C”, which would represent an unbiased estimate of the number of conversions of Conversion type == “Sale” for Campaign == 1. The figure shows one such weighting scheme.



Note that A and B+C are measuring the same quantity. So a revised estimate of the conversion count on A is given by:

$$0.5A + 0.5(B+C) = \mathbf{6}$$

The intuition is that because we have multiple realizations of the same random quantity, we should be able to improve our conversion count estimates by combining these realizations in some way. A more sophisticated approach than naive weighting would be to take the optimal weighted average of parents with their children, where "optimal" is defined as the particular weighted average which minimizes the final variance of the parent node estimate. This idea is discussed in Cormode et al; in the diagram above, we take the simple average between the parent slice and the sum of its children. The result is an improved estimate of the parent's conversion count.

Note that the above approach cannot be used to improve slices on the bottom layer of the aggregate hierarchy because it is a "bottom-up" approach. However, a second "top-down" pass can help to pass information down the tree, ultimately benefiting the most granular aggregates. The top-down approach looks at the difference between a slice and its children and then distributes this difference across all children. This process ensures "consistency" in that each slice's conversion count is exactly equal to the sum of its children; the raw API outputs do not share this property.

Post-Processing of Event-level Report Data and Blending with Aggregate Summary Reports

Once we have improved aggregates by post-processing the Aggregate Summary Report data as above, we can use it to post-process the Event-level Report data and create a unified, more accurate eventified log for various use-cases. We start with the event-level data provided by the Event-level Report data. The next step, which we refer to as **event-debiasing** involves addressing the noising and truncation of conversions on these events. To do this, we take each event in the Event-level Report data and transform the conversions reported by the Event-level Reports for that event by implementing a debiasing transformation. The debiasing transformation implements a correction both for the randomized response noise in the Event-level Reports and the truncation it imposes on attributed conversions. This transformation is data-driven and takes as input a set of summary statistics that index key aspects of the underlying data generating process. We estimate these summary statistics from the improved aggregates we obtain from the Aggregate Summary Reports after

post-processing.

At the end of this process, we will have an improved event-level log that reports conversions attributed to each event that have been partially corrected for the noise and truncation of the Event-level Reports. Further, since the correction leverages information from the Aggregate Summary Reports this corrected log merges information from both API taking advantage of the information content of both. Further, the improved log will have the attractive feature that the log, when aggregated, will be consistent with the post-processed aggregates from the Aggregate Summary Reports (which was not guaranteed to be the case with the raw Event-level Report and Aggregate Summary Reports).

To visualize how this would look, let us index an ad-event in the Event-level Reports by i and denote the conversion count reported by the Event-level Reports as y_i, and by n_i the true total conversion count. n_i is latent and unobserved in the data. Given that n_i is unknown, the goal of event-debiasing is to obtain an estimate the expected total conversion count $E[n_i]$ and use that as an estimate of n_i. The figure below show a simplified version of how the event-level conversion log will look after event-debiasing:

	Conversions	
Event	Registered	Debiased
.	.	.
.	.	.
i	y_i	$E[n_i]$
.	.	.

Implementing Event-Debiasing

Implementing this procedure requires us to address two issues.

1. First, develop a *debiasing transformation* (essentially a function) that effectively leverages the information from the API and produces an unbiased estimate of the true conversion count $E[n_i]$
2. Second, develop a way to obtain data-based "summary statistics" that form an input to the debiasing transformation.

The following debiasing transformation we developed, constructs an estimator $\hat{n}_i$ for the conversions associated with each ad-event i in the Event-level Reports as follows:

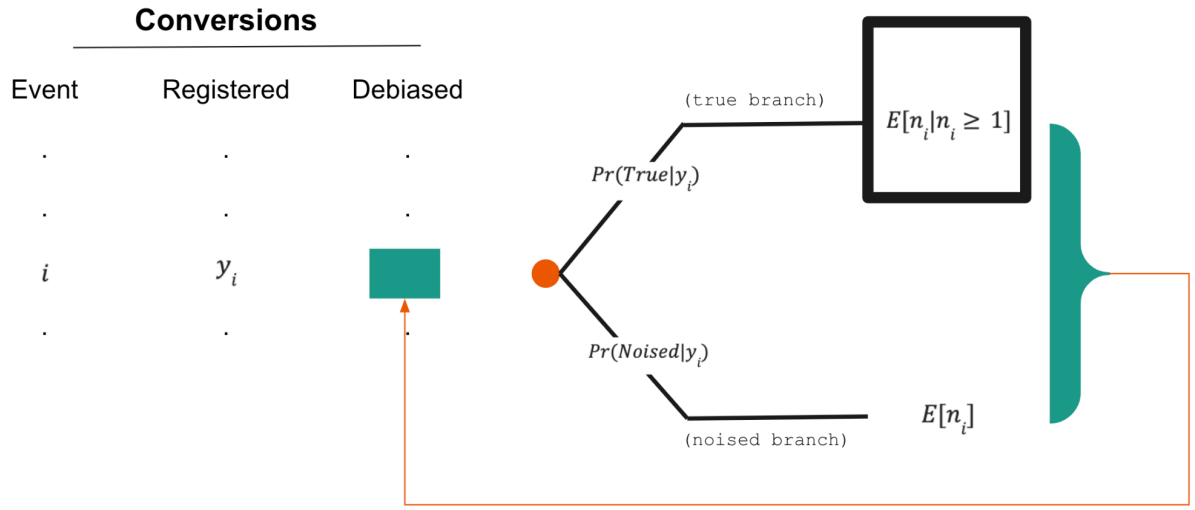
$$\hat{n}_i = E[n_i|y_i] = Pr(True|y_i) \times E[n_i|y_i, True] + Pr(Noised|y_i) \times E[n_i|y_i, Noised]$$

Here, the terms “noised branch” and “true branch” refer to whether the event to which conversions are attributed to by the API are on the noised or true branches. P(Noised branch|.) refers to a probability an event is on the noised branch; equivalently for the true branch. To write the expectation this way, all we have used is that the API leverages binary randomized response - allocating each event into a noised and a true branch with some probability.

This transformation has two attractive properties:

1. First, as a conditional mean, it is an unbiased estimator of n_i because by applying the law of iterated expectations, we can see that $\hat{n}_i = E[E(n_i|y_i)] = E[n_i]$. This guarantees the conversions coming out of this estimator are unbiased for every event.
2. Second, from well known results, the conditional mean achieves the [minimum mean squared error] within the set of estimators that estimate n_i via among all possible functions of y_i. This means this estimator achieves the highest precision among all possible ways of using the information supplied by the API via y_i. This is one of the reasons why the conditional mean is ubiquitously attractive as an estimator.

To illustrate how the formula works, we take ad-views as an example. Consider the following visualization of the probability tree implied by the Event-level Reports for an event i:



The key to notice here is that at the time of event-debiasing, we observe y_i, conversions reported by Event-level Reports. Therefore, when we try to estimate the true conversions for an event i, we can condition on the realization of y_i to do inference on both the probability that an event is on the noised or true branches of the API, and what its true conversions will be.

To understand the picture, first, we note that because of the binary randomized response of the Event-level Reports, each ad-event is either on the true or the noised branch, but the exact state is unknown to ad-techs. This is represented by the orange circle in the picture. As a result, the probability each ad-event is on the true and noised branch - $Pr(True|y_i)$ and $Pr(Noised|y_i)$ - have to be estimated given y_i (these are the conditional probabilities on the branches following the orange circle). Also, trivially, estimating one conditional probability gives the other as $Pr(True|y_i) + Pr(Noised|y_i) = 1$.

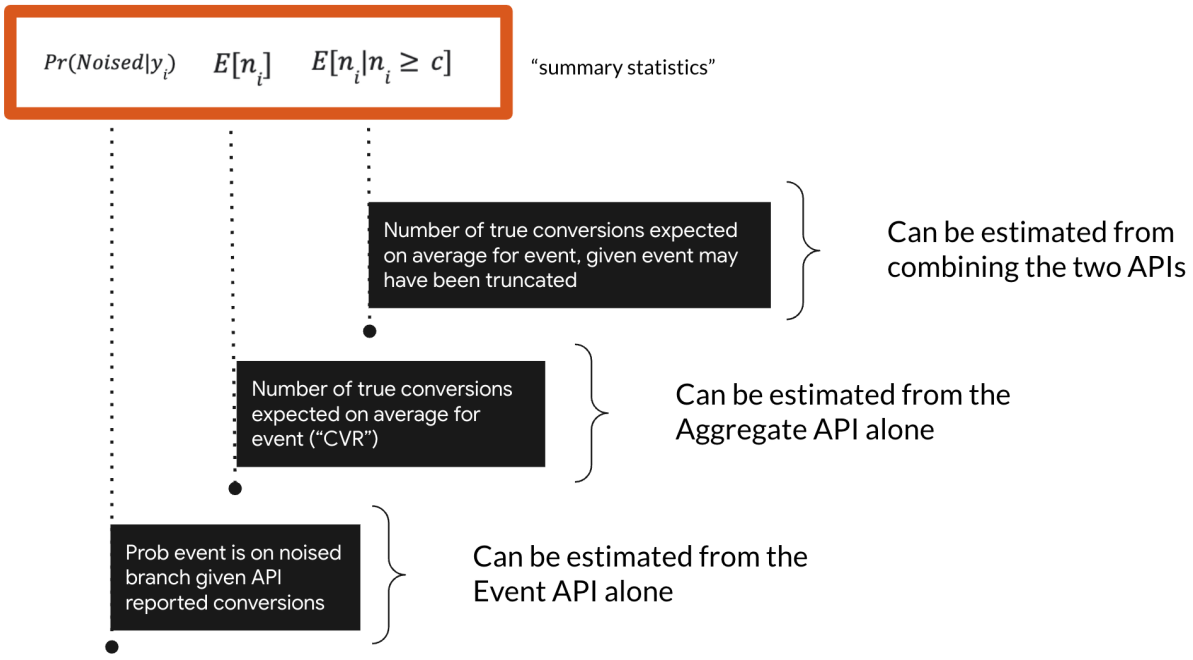
Next we come to our estimates of conversions on each branch. If the ad-event is on the noised branch, there is no relevant information from the API and our best estimate of the number of conversions given the event is on the noised branch is just $E[n_i]$, the average conversion rate (again to be estimated). If the ad-event is on the true branch, we recognize the conversions from event i may have been truncated given that the Event-level Reports report at most 1 conversion following a view. Therefore, given an event is on the true branch, $E[n_i|y_i] = E[n_i|n_i \geq 1]$.

Putting these pieces together gives the debiasing transformation formula for ad-views.

The case for ad-clicks is the same except for the term $E[n_i|y_i]$. Recall, at most 3 conversions can be reported for clicks: therefore, if we observe $y_i < 3$ from the Event-level Reports, we know conversions for event i have not been truncated, and we use $E[n_i|y_i] = y_i$ in the true branch formula. If we observe $y_i \geq 3$ in the Event-level Reports, truncation of conversions for event i may have occurred, so we use $E[n_i|y_i] = E[n_i|n_i \geq 3]$ in the true branch formula.

Parameter Learning and Blending with Aggregate Summary Reports Data

To implement the debiasing formula, we need to estimate key summary statistics from the data. Looking at the debiasing formula, we see the key summary statistics are $Pr(Noised|y_i)$, $E[n_i]$, and $E[n_i|n_i \geq c]$ for different values of c. We now show that all of these summary statistics can be learned from a combination of the raw Event-level Report log and the post-processed aggregates. The figure below illustrates, and we discuss them in sequence right after.



- $Pr(Noised|y_i)$: this is the probability an event is on the noised branch given that y_i conversions are reported by the Event-level Reports for that event. We can learn these objects directly from the Event-level Reports by leveraging our knowledge of the API mechanism. As we know both the noising probability and the way the noised branch configurations are drawn, we can estimate how likely each configuration would occur on the noised branch. We can then compare that with how likely each configuration actually occurs in the ad-tech's Event-level Report data. Intuitively, if a configuration occurs much more often in the Event-level Report data than what we expect only from the noised branch, $Pr(Noised|y_i)$ is small.
- $E[n_i]$: this is the expected conversion rate. We can estimate it by dividing the total conversion count obtained after post-processing the aggregates from the Aggregate Summary Reports, by the count of ad-events registered with the Event-level Reports.
- $E[n_i|n_i \geq c]$: this requires a combination of both API and is more involved to estimate. To understand our approach, imagine the Event-level Reports hypothetically only have a true branch and implements conversion truncation. Also, imagine hypothetically that the conversion count provided by the Aggregate Summary Reports is accurate. In this case, we can exactly calculate the count of conversions being truncated by the Event-level Reports by comparing the difference in the reported conversion count between the two API. We can then calculate on average how many conversions are being truncated for each truncated ad-event. This estimates the object $E[n_i|n_i \geq c]$. To translate this intuition to practice, we need to (1) configure the Aggregate Summary Reports appropriately so that it provides a reasonable overall count; and (2) account for

the noised branch in the Event-level Reports to account for the fact that the conversion count it reports is not fully accurate.

Takeaways

This concludes the description of how we leverage the debiasing transformation and estimates of key summary statistics from the post-processed aggregates from the Aggregate Summary Reports to implement event-debiasing and develop an improved (albeit not perfectly accurate) event-level log.

The log leverages detailed knowledge of both API and combines the informational strengths of both. This provides a unified view of the Event and Aggregate Summary Reports at the event-level. The results are also more accurate as we correct y_i reported by the Event-level Reports for each event to $\hat{n}_i$, which an unbiased estimator of n_i. An auxiliary advantage is consistency between the events and aggregates. Because we use the Aggregate Summary Reports to inform $E[n_i]$ and $E[n_i|n_i \geq c]$, the denoised event-level conversion counts from the eventified log add up to the total count reported by the post-processed aggregates in the Aggregate Summary Reports.

Now that we have described our approach to utilizing the Event-level and Aggregate Summary Reports, we turn to a simple example in Part 3 to illustrate how our methodology works.

Part 3: An Illustrative Example

For illustrative purposes, we focus on a simplified setting with only 1 conversion metadata bucket and only 1 conversion window. Our goal here is to outline the simplest possible example which provides the intuition for the methodology in the clearest way. Suppose we use the ARA API to measure conversions where the 3PC conversion event-level data is as follows:

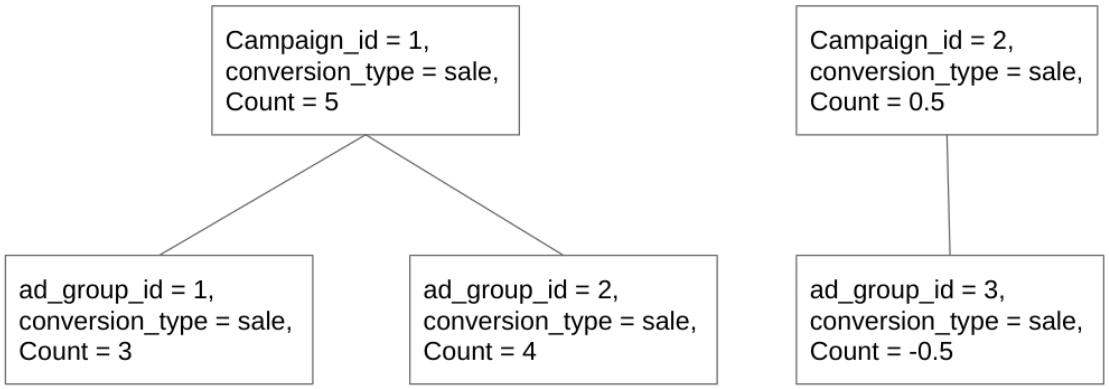
3PC Conversion Data

view_id	campaign_id	ad_group_id	conversion_type	True Conversion Count
1	1	1	Sale	4
2	1	2	Sale	2
3	1	2	Sale	1
4-10	1	1	Sale	0
11-15	2	3	Sale	0

In this example, we assume that the ad impressions are **views** so that the (1, 1, 1) Event-level Report config applies (1 conversion, 1 conversion metadata bit, 1 time window). Of the 15 views that occurred, only views 1, 2, and 3 had conversions.

This data is no longer available after 3PC deprecation; in its place, we will receive Aggregate Summary and Event-level reports from the Attribution Reporting API. To develop an eventified log, we start with the Aggregate Summary Report data. Under a 2-level hierarchical setup, the aggregate information will look as below,

Aggregate Summary Report Data

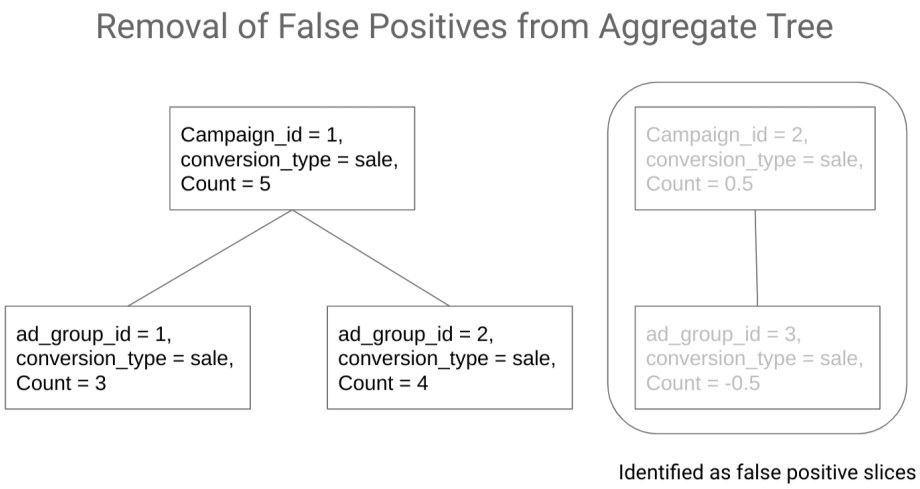


Let's start with the issue of false positives. The right portion of this tree contains the false positive aggregates, since there actually are no conversions corresponding to campaign_id 2.

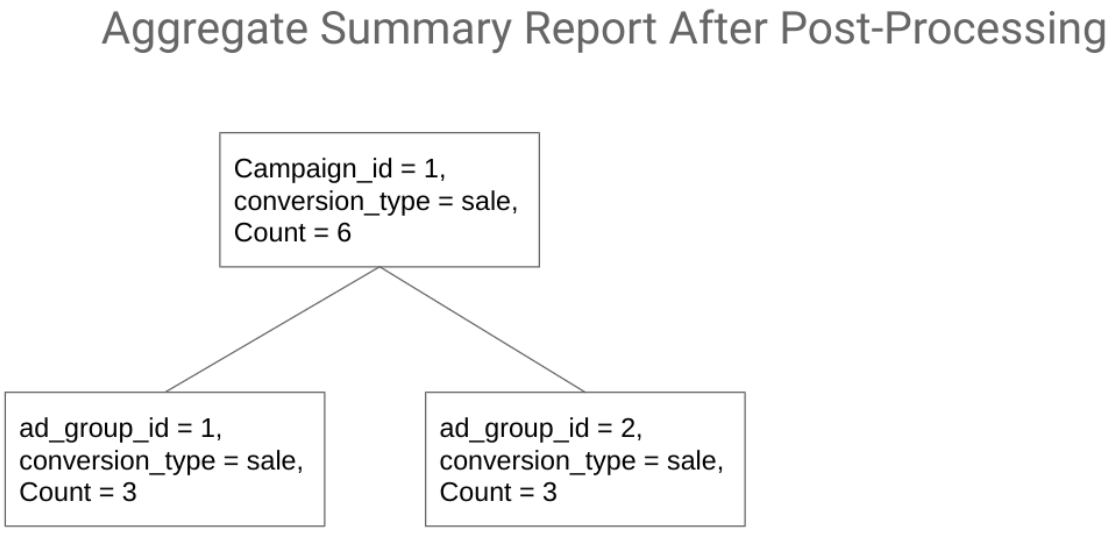
Event-level Report Data

view_id	campaign_id	ad_group_id	conversion_type	API Reported Count
1	1	1	Sale	1
2	1	2	Sale	1
3	1	2	Sale	1

To remove the false positives, we examine the Event-level Report data. The Event-level Report data shows only 3 views have conversions, and notably, we see none of those views come from campaign_id 2. We can conclude that it is very unlikely that any conversions occurred that should be attributed to campaign_id 2: so, we remove these from the aggregates we work with. This leaves an aggregate tree with only the left branch remaining, as depicted below,



Now, we note that the conversion counts on the remaining portion of the aggregate data do not align well. Specifically, the sum of the “children” conversion counts for the tree do not sum to the “parent” conversion count. Applying the post-processing methodology described above address this problem, after which we obtain a tree with aggregates that are better aligned,



Finally, we use the prost-processed Aggregate Summary Report data to create an eventified log. This involves event-debiasing of the Event-level Report data using summary statistics estimated from the raw event-level data and the post-processed aggregates:

Eventified Log After Event-Debiasing

view_id	campaign_id	ad_group_id	conversion_type	API Reported Count	Denoised Count
1	1	1	Sale	1	2
2	1	2	Sale	1	2
3	1	2	Sale	1	2

To see how the event-debiasing works here, recall we use the formula,

$$\hat{n}_i = Pr(True|y_i) \times E[n_i|y_i] + Pr(Noised|y_i) \times E[n_i]$$

to produce the debiased event-level conversion count.

Because the noising probability for views is low, $Pr(True|y_i) \approx 1$ and $Pr(Noised|y_i) \approx 0$, so we can consider $Pr(Noised|y_i) \times E[n_i] \approx 0$.

Also at most 1 conversion is reported per view and all views reported by the API are subject to truncation. Therefore, $E[n_i|y_i] = E[n_i|n_i >= 1]$ and we need to figure out the average number of truncated conversions. Using the campaign-level slice as an example, we can see that the Aggregate API shows there are 6 conversions in campaign 1 and the Event API shows there are 3 conversions in campaign 1 (by adding the “API Reported Count” column for campaign 1). Therefore, 6 (from the Aggregate Summary Reports) - 3 (from the Event-level Reports) = 3 conversions are being truncated in total; 3 views are reported with conversions and thus, on average, 1 conversion is being truncated per view, which leads to an estimate of $E[n_i|n_i >= 1] = 2$. Plugging these summary statistics into the debiasing formula yields $\hat{n}_i = 2$, which is a better estimate. This is shown in the figure under the “debiased count” column.

A similar procedure can be performed with the ad-group-level slices rather than the campaign-level slices. Which slice produces the best performance is an empirical question depending on many factors (e.g., the shape of the Aggregate tree and how the API’s privacy budget is allocated). This can vary from ad-tech to ad-tech and across situations, and needs to be tested and determined systematically for good performance.

As a final takeaway, note that at the end of this procedure, the total conversion counts of the post-processed Aggregate Summary Report and Event-level Report outputs are aligned (6 total). However, the per-event conversion counts will not be 100% accurate (view 1 has too few conversions and view 3 has one too many). They are, however, more accurate than the raw Event-level Report output.

Advanced Topics and Concluding Remarks

This post has sketched some of Google Ad's approaches to post-processing the API data to make it more useful for our ad use-cases. Our approach is to post-process and blend the two API and create a unified and eventified data for all of our use-cases. We believe this is a preferred approach for ad-techs because it takes advantage of the information from both API and ensures consistency across various use-cases. This post is not meant to be comprehensive in describing our approach but rather to lay out some building blocks of our approach without touching on several potential complications. We hope this post inspires interested ad-techs to build upon these building blocks and develop their own approaches for leveraging the API.

As noted in the introduction, there is another aspect to acquiring useful data from the ARA API which pertains to the fact that the API is highly configurable. Each ad-tech needs to configure the API intelligently to produce useful data to fit their ad use-cases. Both configuring and post-processing the API data are critical to obtaining useful data from the API.

Although configuring the API is beyond the scope of this post, we want to briefly discuss some considerations in configuring the API. For instance, on the Event-level Reports, ad-techs can determine what information to encode in the 1 or 3 bits of metadata and ad-techs may want to encode conversion information most important to them (conversion type, conversion value, etc.). Ad-techs can also determine the last conversion window used to capture conversions. A longer window may capture more conversions yet at the expense of a delay in the conversion report. Ad-techs may want to examine the delay profile of their conversions and determine the optimal conversion window. Also the API allows setting priority for different conversions and ad-techs may also want to prioritize more important conversions in attribution.

There are even more configurable levers in the Aggregate Summary Reports. One is free to choose the set of aggregate variables included in the report (campaign, ad-group, conversion type, etc.), the depth and width of the aggregation tree (if you decide to use a tree structure and leverage the post-processing technique in the post), the amount of privacy budget allocated to different levels, and the number of conversions to measure for each ad-interaction. There is no gold standard for these decisions and, similar to the Event-level Reports, ad-techs need to look into what conversion information is critical for their use-cases, include it and spend budget on it to obtain more accurate information on it from the Aggregate Summary Reports. To effectively unify the Event and Aggregate Summary Report data, ad-techs may also want to include some common metadata in both APIs (e.g. conversion types).