

# Privacy Proofs for OpenDP: Lipschitz Sized Variance for Proportion CI (for Partitioned Data)

Summer 2022

# Contents

<b>1</b>	<b>Algorithm Implementation</b>	<b>1</b>
1.1	Code in Rust . . . . .	1
1.2	Pseudo Code in Python . . . . .	1
<b>2</b>	<b>Proof</b>	<b>2</b>

## 1 Algorithm Implementation

## 1.1 Code in Rust

The current OpenDP library contains the `make_lipschitz_sized_proportion_ci_variance` function estimating the variance of the overall sample proportion for partitioned data. This is defined in the Git repository [https://github.com/opendp/opendp/blob/e9a8ce533a900a6561c0ea3be6berust/src/trans/proportion\\_ci/mod.rs#L184-L263](https://github.com/opendp/opendp/blob/e9a8ce533a900a6561c0ea3be6berust/src/trans/proportion_ci/mod.rs#L184-L263)

## 1.2 Pseudo Code in Python

## Preconditions

To ensure the correctness of the output, we require the following preconditions:

- User-specified types:
  - Variable `sample_sizes` must be of type `Vec<usize>`.
  - Variable `strat_sizes` must be of type `Vec<usize>`.
  - Variable `mean_scale` must be of type `TA`.
  - `TA`: must be of type `float`.

## Postconditions

- A `transformation` is returned (i.e., if a `transformation` cannot be returned successfully, then an error should be returned).

## Pseudo Code

```
1 def make_lipschitz_sized_proportion_ci_variance(sample_sizes, strat_sizes,
2   mean_scale):
3     '''
4     :param strat_sizes: the population size of each stratum
5     :param sample_sizes: sample sizes in each stratum
6     :param mean_scale: scale of Gaussian noise added to mean'''
7     input_domain = ProductDomain<AllDomain<TA>>
8     output_domain = AllDomain<TA>
9     strat_weights = strat_sizes / sum(strat_sizes)
10    def function(sample_sums:Vec<TA>) -> TA:
11        strat_means = sample_sums / sample_sizes
12        strat_var = (strat_sizes - sample_sizes) / strat_sizes *
13        (strat_means * (1 - strat_means)) / (sample_sizes - 1)
14        return sum(strat_weights ** 2 * strat_var) + mean_scale ** 2
15    input_metric = ProductMetric<AbsoluteDistance<TA>>
16    output_metric = AbsoluteDistance<TA>
17    def stability_map(d_in: AbsoluteDistance<TA>) -> TA:
18        sens = max(strat_weights ** 2 * (strat_sizes - sample_sizes) /
19        strat_sizes / (sample_sizes - 1) / sample_sizes
20        return d_in * sens
21    return Transformation(input_domain, output_domain, function,
22    input_metric, output_metric, stability_map)
```

## 2 Proof

**Theorem 2.1.** *For every setting of the input parameter `sample_sizes`, `strat_sizes`, `mean_scale` to `make_lipschitz_sized_proportion_ci_variance` such that the given pre-conditions hold, `make_lipschitz_sized_proportion_ci_variance` raises an exception (at compile time or runtime) or returns a valid transformation with the following properties:*

1. **(Appropriate output domain).** *For every vector  $v$  in the input domain, `function`( $v$ ) is in the output domain.*
2. **(Domain-metric compatibility).** *The domain `input_domain` matches one of the possible domains listed in the definition of `input_metric`, and likewise `output_domain` matches one of the possible domains listed in the definition of `output_metric`.*
3. **(Stability guarantee).** *For every pair of elements  $v, w$  in `input_domain` and for any  $d\_in$ , where  $d\_in$  has the associated type for `input_metric`, if  $v, w$  are  $d\_in$ -close under `input_metric`, then `function`( $v$ ), `function`( $w$ ) are `stability_map`( $d\_in$ )-close under `output_metric`.*

*Proof.*

1. **(Appropriate output domain).** Since `strat_sizes`, `sample_sizes` and `sample_sums` are all of type `Vec<TA>`, we know that `strat_weights`, `strat_means` and `strat_vars` are also of type `Vec<TA>` from the calculation on lines 7, 12 and 13. The function returns a sum of a vector of type `Vec<TA>`, then the sum will be of type `TA`. That is, the output is in the output domain `AllDomain<TA>`.

2. **(Domain-metric compatibility).** The input domain of `make_lipschitz_sized_proportion_ci_variance` is `ProductDomain` of `AllDomain<TA>` and the input metric is `ProductMetric` of `AbsoluteDistance<TA>`. Each component of the input is in `AllDomain<TA>`. Since `AllDomain<TA>` matches one of the possible domains listed in the definition of `AbsoluteDistance`, the input domain is compatible with the input metric.

Also, it follows directly that the output domain (`AllDomain<TA>`) is compatible with the output metric (`AbsoluteDistance<TA>`).

3. **(Stability guarantee.)** Let `Abs` stand for `AbsoluteDistance`. If  $v, w$  are `d_in`-close, then by the definition 1,

$$d_{\text{PM}, \text{Abs}}(v, w) = \sum_i d_{\text{Abs}}(v_i, w_i) \leq \text{d\_in}.$$

Let  $f$  denote the function in `make_lipschitz_sized_proportion_ci_variance`. For ease of notation, let  $N_i, c_i$  and  $n_i$  denote the  $i$ th element of `strat_sizes`, `strat_weights` and `sample_sizes`, respectively. Note that the sample sum should satisfy  $0 \leq v_i \leq n_i$  and  $0 \leq w_i \leq n_i$ , then

$$\left| 1 - \frac{v_i + w_i}{n_i} \right| \leq 1. \quad (1)$$

Then

$$\begin{aligned} d_{\text{Abs}}(f(v), f(w)) &= \left| \sum_i c_i^2 \cdot \frac{N_i - n_i}{N_i(n_i - 1)} \cdot \frac{v_i}{n_i} \left( 1 - \frac{v_i}{n_i} \right) - \sum_i c_i^2 \cdot \frac{N_i - n_i}{N_i(n_i - 1)} \cdot \frac{w_i}{n_i} \left( 1 - \frac{w_i}{n_i} \right) \right| \\ &= \sum_i c_i^2 \cdot \frac{N_i - n_i}{N_i(n_i - 1)} \cdot \left| \frac{v_i}{n_i} \left( 1 - \frac{v_i}{n_i} \right) - \frac{w_i}{n_i} \left( 1 - \frac{w_i}{n_i} \right) \right| \\ &= \sum_i c_i^2 \cdot \frac{N_i - n_i}{N_i(n_i - 1)} \cdot \left| \frac{v_i - w_i}{n_i} \left( 1 - \frac{v_i + w_i}{n_i} \right) \right| \\ &\stackrel{\text{Eq. (1)}}{\leq} \sum_i c_i^2 \cdot \frac{N_i - n_i}{N_i(n_i - 1)} \cdot \left| \frac{v_i - w_i}{n_i} \right| \\ &= \sum_i c_i^2 \cdot \frac{N_i - n_i}{N_i(n_i - 1)n_i} \cdot d_{\text{Abs}}(v_i, w_i) \\ &\leq \max_i \frac{c_i^2(N_i - n_i)}{N_i(n_i - 1)n_i} \cdot \sum_i d_{\text{Abs}}(v_i, w_i) \\ &\leq \max_i \frac{c_i^2(N_i - n_i)}{N_i(n_i - 1)n_i} \cdot \text{d\_in}. \end{aligned}$$

That is,  $f(v)$  and  $f(w)$  are `stability_map(d_in)`-close.

□

**Definition 1** (Distance under `ProductMetric`). Let  $d_{\text{PM}, M}$  denote the distance under `ProductMetric`( $M$ ) where  $M$  is a valid metric. Then  $d_{\text{PM}, M}$  is defined as the sum of distance under each  $M$ . Specifically, for any  $v, w$  in the input domain and  $v_i, w_i$  denote their  $i$ th entry, respectively,

(i) for input metric  $MI$ ,

$$d_{PM,MI}(v, w) = \sum_i d_{MI}(v_i, w_i).$$

(ii) for output metric  $MO$ ,

$$d_{PM,MO}(g(v), g(w)) = \sum_i d_{MO}(f_i(v_i), f_i(w_i)),$$

where  $g$  and  $f_i$  denote the **function** in their corresponding **Transformation**.