

# fn counting\_query\_stability\_map

Michael Shoemate

September 30, 2025

This document proves that the implementation of `counting_query_stability_map` in `mod.rs` at commit `f5bb719` (outdated<sup>1</sup>) satisfies its proof definition.

## 1 Hoare Triple

### Preconditions

#### Compiler-verified

- Argument `public_info` must be `Keys`, `Lengths` or `None`
- Generic `M` must implement `UnboundedMetric`
- Const generic `P` must be of type `usize`

#### User-verified

None

### Pseudocode

```
1 def counting_query_stability_map(  
2     public_info: Literal["Keys"] | Literal["Lengths"] | None,  
3 ) -> StabilityMap[PartitionDistance[M], LpDistance[P, f64]]:  
4  
5     if public_info == "Lengths": #  
6         return StabilityMap.new(lambda _: 0.)  
7  
8     def norm_map(d_in: f64) -> f64: #  
9         if P == 1:  
10             return d_in  
11         if P == 2:  
12             return d_in.inf_sqrt()  
13         raise ValueError("unsupported Lp norm. Must be an L1 or L2 norm.")  
14  
15     def stability_map(d_in: tuple[u32, u32, u32]) -> f64:  
16         l0, l1, l_inf = d_in #  
17         l0_p = norm_map(f64.from_(l0)) #  
18         l1_p = f64.from_(l1)  
19         l_inf_p = f64.from_(l_inf)  
20         return l1_p.total_min(l0_p.inf_mul(l_inf_p)) #  
21  
22     return StabilityMap.new_fallible(stability_map) #
```

---

<sup>1</sup>See new changes with `git diff f5bb719..4850993 rust/src/transformations/make_stable_expr/expr_count/mod.rs`

## Postcondition

**Definition 1.1.** For any setting of the input parameters `public_info`, `M` and `P`, returns a `StabilityMap` where for any predicate  $p(\cdot)$  and  $f(x) = [\sum_i \mathbb{1}_{p(x_{1i})}, \sum_i \mathbb{1}_{p(x_{2i})}, \dots]$ , if  $d_{\text{PartitionDistance} < M >}(x, x') \leq d_{in}$ , then  $d_{LP}(f(x), f(x')) \leq d_{out}$ , where  $d_{out} = \text{StabilityMap.eval}(d_{in})$ .

*Proof.* Since the input metric is `PartitionDistance < M >`, and `M` is an unbounded dataset distance metric (with associated distance type `u32`), the distance type is a tuple of the  $L_0$ ,  $L_1$  and  $L_\infty$  distances between the per-partition distances with respect to the input metric `M`, as shown on 16.

$$d_{LP}(f(x), f(x')) \tag{1}$$

$$= d_{LP} \left( \left[ \sum_j^{\text{len}(x_1)} \mathbb{1}_{p(x_{1j})}, \sum_j^{\text{len}(x_2)} \mathbb{1}_{p(x_{2j})}, \dots \right], \left[ \sum_j^{\text{len}(x_1)} \mathbb{1}_{p(x'_{1j})}, \sum_j^{\text{len}(x_2)} \mathbb{1}_{p(x'_{2j})}, \dots \right] \right) \tag{2}$$

$$= \left( \sum_i^{\text{len}(x)} \left( \sum_j^{\text{len}(x_i)} \mathbb{1}_{p(x_{ij})} - \mathbb{1}_{p(x'_{ij})} \right)^P \right)^{1/P} \tag{3}$$

Consider two cases. First, when substituting the  $\Delta_0$  and  $\Delta_\infty$  bounds:

$$= \left( \sum_i^{\text{len}(x)} \left( \sum_j^{\text{len}(x_i)} \mathbb{1}_{p(x_{ij})} - \mathbb{1}_{p(x'_{ij})} \right)^P \right)^{1/P} \tag{4}$$

$$\leq \left( \sum_i^{\Delta_0} (\Delta_\infty)^P \right)^{1/P} \tag{5}$$

$$= \Delta_0^{1/P} \Delta_\infty \tag{6}$$

Alternatively, apply the  $\Delta_1$  bound, considering that distance is maximized when all  $\Delta_1$  contributions are made to the same partition:

$$= \left( \sum_i^{\text{len}(x)} \left( \sum_j^{\text{len}(x_i)} \mathbb{1}_{p(x_{ij})} - \mathbb{1}_{p(x'_{ij})} \right)^P \right)^{1/P} \tag{7}$$

$$\leq (\Delta_1^P)^{1/P} \tag{8}$$

$$= \Delta_1 \tag{9}$$

The sensitivity is no greater than the smaller of these two bounds:

$$d_{LP}(f(x), f(x')) \leq \min(\Delta_1, \Delta_0^{1/P} \Delta_\infty) \tag{10}$$

We now switch to the pseudocode. If partition length is invariant (via `public_info`), then  $\Delta_1 = \Delta_\infty = 0$ , making the query insensitive. This invariant is reflected in 5, where an insensitive stability map is returned, satisfying the postcondition.

`norm_map` on line 8 computes  $x^{1/P}$ , and is used on line 17 to compute  $\Delta^{1/P}$ . Line 20 then returns the bound from equation 10. Therefore, the stability map returned on line 22 also satisfies the postcondition.  $\square$