

# fn truncate\_id\_bound

Michael Shoemate

September 17, 2025

This proof resides in “**contrib**” because it has not completed the vetting process.

Proves soundness of `truncate_id_bound` in `mod.rs` at commit `f5bb719` (outdated<sup>1</sup>). `truncate_id_bound` returns an upper bound on dataset distances in terms of the symmetric distance metric.

## 1 Hoare Triple

### Precondition

### Caller Verified

- `id_bound.by = truncate.by`

### Function

```
1 def truncate_id_bound(  
2     id_bound: Bound,  
3     truncation: Bound,  
4     total_ids: Optional[int],  
5 ) -> Bound:  
6     # Once truncated, max contributions when grouped by "over" are bounded  
7     row_bound = Bound.by(truncation.by)  
8  
9     # In each group, the worst-case row contributions is the  
10    # the number of ids contributed (known from id_bound)  
11    # times the number of rows contributed under each id (known from truncation),  
12    num_ids, num_rows = id_bound.per_group, truncation.per_group  
13    if num_ids is not None and num_rows is not None:  
14        row_bound = row_bound.with_per_group(num_ids.inf_mul(num_rows)) #  
15  
16    # Worst case number of groups contributed is the  
17    # total number of ids contributed (total_ids)  
18    # times the number of groups contributed under each id (known from truncation).  
19    num_groups_via_truncation = None #  
20    if total_ids is not None and truncation.num_groups is not None:  
21        num_groups_via_truncation = total_ids.inf_mul(truncation.num_groups)  
22  
23    # Alternatively, the number of groups contributed may be known outright from id_bound.  
24    # Use the smaller of the two if both are known.  
25    num_groups = option_min(num_groups_via_truncation, id_bound.num_groups)  
26    if num_groups is not None:  
27        row_bound = row_bound.with_num_groups(num_groups) #  
28  
29    return row_bound
```

<sup>1</sup>See new changes with `git diff f5bb719..e4fb7f4 rust/src/transformations/make_stable_lazyframe/truncate/mod.rs`

## Postcondition

**Theorem 1.1** (Postcondition). Let  $g$  vary over groups when partitioned by `id_bound.by`. If for any two datasets  $x, x'$  we have that

$$\begin{aligned} \|d_{\text{SymId}}(\text{function}(x)_g, \text{function}(x')_g)\|_\infty &\leq \text{id\_bound.per\_group}, \\ \|d_{\text{SymId}}(\text{function}(x)_g, \text{function}(x')_g)\|_0 &\leq \text{id\_bound.num\_groups}, \\ d_{\text{SymId}}(\text{function}(x), \text{function}(x')) &\leq \text{total\_ids}, \end{aligned}$$

and `function` truncates a dataset such that,

$$\begin{aligned} \max_{id} \|d_{\text{Sym}}(\text{function}(x)_{id,g}, \text{function}(x')_{id,g})\|_\infty &\leq \text{truncation.per\_group}, \\ \max_{id} \|d_{\text{Sym}}(\text{function}(x)_{id,g}, \text{function}(x')_{id,g})\|_0 &\leq \text{truncation.num\_groups}, \end{aligned}$$

then we have that

$$\begin{aligned} \|d_{\text{Sym}}(\text{function}(x)_g, \text{function}(x')_g)\|_\infty &\leq \text{row\_bound.per\_group}, \\ \|d_{\text{Sym}}(\text{function}(x)_g, \text{function}(x')_g)\|_0 &\leq \text{row\_bound.num\_groups}, \end{aligned}$$

where `row_bound` denotes the return value.

*Proof.* Assume the preconditions are met, as well as the conditions of the postcondition.

We first prove the per-group bound.

$$\begin{aligned} &\|d_{\text{Sym}}(\text{function}(x)_g, \text{function}(x')_g)\|_\infty \\ &\leq \|d_{\text{SymId}}(\text{function}(x)_g, \text{function}(x')_g)\|_\infty \cdot \max_{id} \|d_{\text{Sym}}(\text{function}(x)_{id,g}, \text{function}(x')_{id,g})\|_\infty \\ &\leq \text{id\_bound.per\_group} \cdot \text{truncation.per\_group} \\ &= \text{row\_bound.per\_group} \quad \text{by line 14} \end{aligned}$$

We now prove the number of groups bound. There are two ways to bound the number of contributed groups. We first reason by the total number of identifiers.

$$\begin{aligned} &\|d_{\text{Sym}}(\text{function}(x)_g, \text{function}(x')_g)\|_0 \\ &\leq d_{\text{SymId}}(\text{function}(x), \text{function}(x')) \cdot \max_{id} \|d_{\text{Sym}}(\text{function}(x)_{id,g}, \text{function}(x')_{id,g})\|_0 \\ &\leq \text{total\_ids} \cdot \text{truncation.num\_groups} \\ &= \text{num\_groups\_via\_truncation} \quad \text{by line 19} \end{aligned}$$

We can also directly bound the number of contributed groups by `id_bound.num_groups`:

$$\begin{aligned} &\|d_{\text{Sym}}(\text{function}(x)_g, \text{function}(x')_g)\|_0 \\ &= \|d_{\text{SymId}}(\text{function}(x)_g, \text{function}(x')_g)\|_0 \\ &\leq \text{id\_bound.num\_groups} \end{aligned}$$

This is a valid upper bound on the number of contributed groups, because truncation is applied independently to each group.

Therefore, we take the minimum of the two upper bounds.

$$\begin{aligned} &\|d_{\text{Sym}}(\text{function}(x)_g, \text{function}(x')_g)\|_0 \\ &\min(\text{num\_groups\_via\_truncation}, \text{id\_bound.num\_groups}) \\ &= \text{row\_bound.num\_groups} \quad \text{by line 27} \end{aligned}$$

In the implementation, any of the input bounds could be missing. When a bound is missing, any output bounds that require it are not claimed.  $\square$