

# fn make\_row\_by\_row

Michael Shoemate

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

Proves soundness of `make_row_by_row` in `mod.rs` at commit `f5bb719` (outdated<sup>1</sup>). This constructor is a special case of `make_row_by_row_fallible`. The proof for this constructor appeals to the proof for `make_row_by_row_fallible`.

`make_row_by_row` returns a Transformation that applies a user-specified function to each record in the input dataset.

## Vetting History

- [Pull Request #562](#)

## 1 Hoare Triple

### Precondition

- DI (input domain) is a type with trait `RowByRowDomain<D0>`. This trait provides a way to apply a map function to each record in the input dataset to retrieve a dataset that is a member of the output domain, of type D0. The trait further implies that `DatasetDomain` is also implemented for DI.
- D0 (output domain) is a type with trait `DatasetDomain`. `DatasetDomain` is used to define the type of the row domain.
- M (metric) is a type with trait `DatasetMetric`. `DatasetMetric` is used to restrict the set of valid metrics to those which measure distances between datasets.
- `MetricSpace` is implemented for (DI, M). Therefore M is a valid metric on DI.
- `MetricSpace` is implemented for (D0, M).
- `row_function` has no side-effects.
- If the input to `row_function` is a member of `input_domain`’s row domain, then the output is a member of `output_row_domain`.

### Pseudocode

```
1 def make_row_by_row_fallible(  
2     input_domain: DI,  
3     input_metric: M,  
4     output_row_domain: D0,  
5     # a function from input domain's row type to output domain's row type
```

<sup>1</sup>See new changes with `git diff f5bb719..84c4c47 rust/src/transformations/manipulation/mod.rs`

```

6   row_function: Callable([[DI::RowDomain::Carrier], D0::RowDomain::Carrier])
7 ) -> Transformation:
8   make_row_by_row_fallible(input_domain, input_metric, output_row_domain, row_function)

```

## Postcondition

For every setting of the input parameters (`input_domain`, `input_metric`, `output_domain`, `row_function`, `DI`, `D0`, `M`) to `make_row_by_row` such that the given preconditions hold, `make_row_by_row` raises an exception (at compile time or run time) or returns a valid transformation. A valid transformation has the following properties:

1. (Appropriate output domain). For every element  $x$  in `input_domain`, `function(x)` is in `output_domain` or raises a data-independent runtime exception.
2. (Stability guarantee). For every pair of elements  $x, x'$  in `input_domain` and for every pair  $(d_{in}, d_{out})$ , where  $d_{in}$  has the associated type for `input_metric` and  $d_{out}$  has the associated type for `output_metric`, if  $x, x'$  are  $d_{in}$ -close under `input_metric`, `stability_map(d_in)` does not raise an exception, and `stability_map(d_in)  $\leq$  d_out`, then `function(x), function(x')` are  $d_{out}$ -close under `output_metric`.

## 2 Proofs

*Proof. (Part 1 – appropriate output domain).* Since the preconditions for this constructor are a superset of the preconditions on `make_row_by_row_fallible`, the proof of `make_row_by_row_fallible` applies. Thus, by the output domain proof on `make_row_by_row_fallible`, for all settings of input arguments, the function returns a dataset in the output domain.  $\square$

*Proof. (Part 2 – stability map).* The proof of `make_row_by_row_fallible` similarly applies. Thus, by the stability map proof on `make_row_by_row_fallible`, for all settings of input arguments, where  $u, v$  are  $d_{in}$ -close under `input_metric`, `function(u), function(v)` are  $d_{out}$ -close under `output_metric`.  $\square$