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Proves soundness of `make_row_by_row` in `mod.rs` at commit f5bb719 (outdated[1]). 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<DO>`. 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 `DO`. The trait further implies that `DatasetDomain` is also implemented for `DI`.
- `DO` (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 `(DO, 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

```
def make_row_by_row(
    input_domain: DI,
    input_metric: M,
    output_row_domain: DO,
    # a function from input domain's row type to output domain's row type
```

[1] See new changes with `git diff f5bb719..d5d3e63 rust/src/transformations/manipulation/mod.rs`

```
    row_function: Callable[[DI_RowDomain_Carrier], DO_RowDomain_Carrier])
) -> Transformation:

    return make_row_by_row_fallible(
        input_domain, input_metric, output_row_domain, row_function
    )
```

Postcondition

Theorem 1.1. For every setting of the input parameters `(input_domain, input_metric, output_domain, row_function, DI, DO, 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. □

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`. □