Written by Grace Tian

1 Is Equal

Lines 40-56 of https://github.com/opendp/opendp/blob/21-impute/rust/opendp/src/trans/cast.rs#L40-L56

1.1 Code

```rust
/// A [`Transformation`] that checks equality elementwise with `value`.
/// Maps a Vec<T> -> Vec<bool>
pub fn make_is_equal<M, T>(
    value: T
) -> Fallible<Transformation<VectorDomain<AllDomain<T>>, VectorDomain<AllDomain<bool>>, M, M>>
    where M: DatasetMetric,
          T: 'static + Eq,
          M::Distance: One + DistanceConstant {
    Ok(Transformation::new(
        VectorDomain::new_all(),
        VectorDomain::new_all(),
        Function::new(move |arg: &Vec<T>| arg.iter().map(|v| v == &value).collect()),
        M::default(),
        M::default(),
        StabilityRelation::new_from_constant(M::Distance::one())
    ))
}
```

1.2 Pseudo Code

```
def make_is_equal(value : T, metric):
    input_domain = (VectorDomain(AllDomain(T));
    output_domain = (VectorDomain(AllDomain(bool));
    input_metric = metric;
    let output_metric = metric
    assert is_instance(metric, SymmetricDistance);

    let function(data: Vec<T>) -> Vec<Bool>:
        return map(data == value);
    let stability_relation = lambda(d_in, d_out): (d_in <= d_out);

    return Transformation(input_domain, output_domain, function,
    input_metric, output_metric, stability_relation)

Is_Equal = make_is_equal(value, SymmetricDistance);
```

1.3 Conditions as Specified in Pseudo Code

- Input Domain: all vector domain of input value type `T` (same type as input `value`). T is a generic type that is static and can be evaluated with equality.
- Output Domain: all vector domain of type `bool`.

- Function: returns vector mapping of whether vector elements equal input `value`.
- Input Metric: Symmetric Distance
- Output Metric: same as Input Metric.
- Stability Relation: function that takes in 32 bit integers d_{in} and d_{out} and returns whether $d_{in} \leq d_{out}$.

(silvia) Do we need to include the stability parameter? And perhaps include an example calling `make_is_equal` with metrics?

(silvia) Shouldn't it be checked / imposed that the end user passes the same input and out metric somewhere? Otherwise have only one?

1.4 Proof

Theorem 1.1. *For every setting of the input parameters to* `make_is_equal`*, the transformation returned by* `make_is_equal` *has the following properties*

1. (Appropriate output domain). *If vector* `v` *of type* `T` *is in the input domain, then* `function(v)` *is in the output domain, otherwise the program will raise an* `Exception`*.*
2. (Stability Guarantee). *If two vector inputs* v, w *are "*d_{in} *- close" and if* `Relation(`d_{in}`,` d_{out}`) = True` *then the corresponding outputs* $function(v), function(w)$ *are "*d_{out} *- close".*

Proof. 1. (Appropriate output domain). Rust automatically enforces the type system to ensure it is in the correct output domain. The type signature of the function in the pseudo code and Rust code takes in a `Vec<T>` and returns `Vec<Bool>`. Since the rust code successfully compiles, by definition of the type signature the theorem must hold. Otherwise, the code will raise an exception for incorrect input type.

2. (Stability Guarantee). We can assume $d_{in} \leq d_{out}$ because Relation(d_{in}, d_{out}) = `True`. We consider the symmetric distance as input metric. Since vector inputs v, w are d_{in}-close, then by definition the symmetric distance of the multisets is bounded by d_{in}: $|MultiSet(v) \Delta MultiSet(w)| \leq d_{in}$

   We want to show that $|MultiSet(function(v)) \Delta MultiSet(function(w))| \leq d_{out}$. To do so, we consider the symmetric distance between the multiset transformations $function(v)$ and $function(w)$. We claim that for any pair of set elements $v^* \in Multiset(v)$ and $w^* \in Multiset(w)$

   $$|MultiSet(function(v)) \Delta MultiSet(function(w))| \leq |MultiSet(v) \Delta MultiSet(w)|$$

   We consider 4 cases.

   - WLOG v^* == `value`, $w^* \neq$ `value`
     When we apply the `function()` to the elements, we will still get different values (`T` and `F`). So the symmetric distance stays the same after we transform 2 sets with these elements separately.

- $v^* ==$ `value`, $w^* ==$ `value`
  When we apply the `function()` to the elements, we will still get same values (`T` and `T`). So the symmetric distance stays the same after we transform 2 sets with these elements separately.
- $v^* \neq$ `value`, $w^* \neq$ `value`
  When we apply the `function()` to the elements (which are not necessarily the same), we will get the same values (both `F`). So the symmetric distance must nonstrictly decrease after we transform 2 sets with these elements separately.

Since we considered all possible pairs of elements, the symmetric distance must nonstrictly decrease after we transform 2 multi sets v and w.

$$|MultiSet(function(v))\Delta MultiSet(function(w))| \leq |MultiSet(v)\Delta MultiSet(w)|$$

Since the right hand side $\leq d_{in} \leq d_{out}$, then the transformed outputs are d_{out}-close: $|MultiSet(function(v))\Delta MultiSet(function(w))| \leq d_{out}$.

□

(silvia) I think you are using the histogram notation implicitly because I don't think you can say something like $function(v) = T$ (for true) because it is a vector/set, and you mean to say one entry, right? (grace) Not super sure - it's just function applied to a singular element v but that's not a vector of size 1. (silvia) Ah then it is the map we discussed today