

fn make_geometric

Michael Shoemate

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

Proves soundness of the implementation of `make_geometric` in `mod.rs` at commit `f5bb719` (outdated¹).

The implementation of this function constructs a random variable denoting the noise distribution to add, and then dispatches to the `MakeNoise<DI, MI, MO>` trait which constructs the core mechanism and wraps it in pre-processing transformations and post-processors to match the desired parameterization.

1 Hoare Triple

Precondition

Compiler-Verified

- generic `DI` implements trait `Domain`
- generic `MI` implements trait `Metric`
- generic `MO` implements trait `Measure`
- type `DiscreteLaplace` implements trait `MakeNoise<DI, MI, MO>`
- type `ConstantTimeGeometric` implements trait `MakeNoise<DI, MI, MO>` These traits constrain the choice of input domain, input metric and output measure to those that can form valid measurements when adding noise from these distributions.
- type `(DI, MI)` implements trait `MetricSpace`

User-Verified

None

Pseudocode

```
1 def make_geometric(
2     input_domain: DI,
3     input_metric: MI,
4     scale: f64,
5     bounds: Option[tuple[DI_Atom, DI_Atom]],
6 ) -> Measurement[DI, DI_Carrier, MI, MO]:
7     input_space = input_domain, input_metric
8     if bounds is None:
9         return DiscreteLaplace(scale, k=None).make_noise(input_space)
10    else:
11        return ConstantTimeGeometric(scale, bounds).make_noise(input_space)
```

¹See new changes with git diff `f5bb719..517e09e rust/src/measurements/noise/distribution/geometric/mod.rs`

Postcondition

Theorem 1.1. For every setting of the input parameters (`input_domain`, `input_metric`, `scale`, `bounds`, `DI`, `MI`, `M0`) to `make_geometric` such that the given preconditions hold, `make_geometric` raises an error (at compile time or run time) or returns a valid measurement. A valid measurement has the following properties:

1. (Data-independent runtime errors). For every pair of members x and x' in `input_domain`, `invoke(x)` and `invoke(x')` either both return the same error or neither return an error.
2. (Privacy guarantee). For every pair of members x and 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_measure`, if x, x' are d_{in} -close under `input_metric`, `privacy_map(d_in)` does not raise an error, and $\text{privacy_map}(d_{in}) = d_{out}$, then `function(x), function(x')` are d_{out} -close under `output_measure`.

Proof. If bounds are supplied, this constructor builds a specialized mechanism that adds noise to the input data from the `ConstantTimeGeometric` random variable. Otherwise noise is added from the `DiscreteLaplace` random variable, which uses a logarithmic-time discrete laplace sampling algorithm.

Since `MakeNoise.make_noise` has no preconditions, the postcondition follows, which matches the post-condition for this function. \square