

# fn make\_vec

Michael Shoemate

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

Proves soundness of the implementation of `make_vec` in `mod.rs` at commit `f5bb719` (outdated<sup>1</sup>).

This transformation simply wraps an input scalar in a singleton vec. The output metric then becomes an Lp distance.

## 1 Hoare Triple

### Precondition

#### Compiler-Verified

- Generic T implements trait `Number`
- Generic Q implements trait `Number`

#### User-Verified

None

### Pseudocode

```
1 def make_vec(  
2     input_space: tuple[AtomDomain[T], AbsoluteDistance[Q]],  
3 ) -> Transformation[  
4     AtomDomain[T], VectorDomain[AtomDomain[T]], AbsoluteDistance[Q], LpDistance[P, Q]  
5 ]:  
6     input_domain, input_metric = input_space  
7     return Transformation.new(  
8         input_domain,  
9         VectorDomain.new(input_domain).with_size(1),  
10        lambda arg: [arg],  
11        input_metric,  
12        LpDistance.default(),  
13        lambda d_in: d_in,  
14    )
```

### Postcondition

#### Theorem 1.1.

**Theorem 1.2.** For every setting of the input parameters (`input_space`, T, Q) to `make_vec` such that the given preconditions hold, `make_vec` raises an error (at compile time or run time) or returns a valid transformation. A valid transformation has the following properties:

---

<sup>1</sup>See new changes with `git diff f5bb719..8b6d431 rust/src/transformations/scalar_to_vector/mod.rs`

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. (Appropriate output domain). For every member  $x$  in `input_domain`, `function(x)` is in `output_domain` or raises a data-independent runtime error.
3. (Stability guarantee). For every pair of members  $x$  and  $x'$  in `input_domain` and for every pair  $(\mathbf{d\_in}, \mathbf{d\_out})$ , where  $\mathbf{d\_in}$  has the associated type for `input_metric` and  $\mathbf{d\_out}$  has the associated type for `output_metric`, if  $x, x'$  are  $\mathbf{d\_in}$ -close under `input_metric`, `stability_map(d_in)` does not raise an error, and `stability_map(d_in) = d_out`, then `function(x), function(x')` are  $\mathbf{d\_out}$ -close under `output_metric`.

*Proof.* The function is infallible, and the output domain trivially follows, since all output vectors are length-one. For all  $x$  in the input domain, the output of `make_vec` is a vector of length 1, so the output domain is trivially valid. The function is 1-stable because:

$$\mathbf{d\_out} \tag{1}$$

$$= \max_{x \sim x'} d_{Lp}(f(x), f(x')) \tag{2}$$

$$= \max_{x \sim x'} \left( \sum_i (x_i - x'_i)^p \right)^{1/p} \tag{3}$$

$$= \max_{x \sim x'} ((x_1 - x'_1)^p)^{1/p} \tag{4}$$

$$= \max_{x \sim x'} |x_1 - x'_1| \tag{5}$$

$$= \max_{x \sim x'} d_{Abs}(x_1, x'_1) \tag{6}$$

$$= 1 \cdot \mathbf{d\_in} \tag{7}$$

$$\tag{8}$$

For every pair of elements  $x, x'$  in `input_domain` and for every pair  $(\mathbf{d\_in}, \mathbf{d\_out})$ , where  $\mathbf{d\_in}$  has the associated type for `input_metric` and  $\mathbf{d\_out}$  has the associated type for `output_metric`, if  $x, x'$  are  $\mathbf{d\_in}$ -close under `input_metric`, `stability_map(d_in)` does not raise an exception, and `stability_map(d_in) ≤ d_out`, then `function(x), function(x')` are  $\mathbf{d\_out}$ -close under `output_metric`.  $\square$