Minard Database Dimensionality Report

Generated: 2026-02-03 **Status**: active **Category**: research

Executive Summary

Analysis of 1,381 modules across 159 packages reveals key structural patterns that should inform visualization design decisions.

Key Findings

Scale

Metric	Count
Total packages	159
Total modules	1,381
Total declarations (est.)	~15,000
Total function calls	19,210
Total child declarations	~2,000

Modules per Package

Statistic	Value
Min	1
Max	197 (web-html)
Median	3
Mean	~8

Distribution:

- 43 packages (27%) have exactly 1 module
- 59 packages (37%) have 2-5 modules
- 33 packages (21%) have 6-10 modules
- 14 packages (9%) have 11-20 modules
- 10 packages (6%) have 21+ modules

Implication: Most packages are small. A few large packages (web-html, hylograph-selection, prelude) dominate. Visualizations should handle 1-10 modules gracefully while scaling to 200.

Declarations per Module

Statistic	Value

Statistic	Value
Min	1
Max	265
Median	6
Mean	11

Distribution:

- 629 modules (46%) have 1-5 declarations
- 285 modules (21%) have 6-10 declarations
- 234 modules (17%) have 11-20 declarations
- 198 modules (14%) have 21-50 declarations
- 35 modules (3%) have 51+ declarations

Implication: Most modules are small (median 6 declarations). Circle packing works well. Large modules (50+ decls) need different treatment - perhaps aggregation or hierarchical nesting.

Declarations by Kind

From sampled data (50 modules):

Kind	Count	%
value	394	77%
type_synonym	76	15%
data	39	8%
newtype	rare	<1%
type_class	rare	<1%
foreign	rare	<1%

Implication: Values (functions) dominate. Type definitions are ~20% of declarations. The current color scheme appropriately emphasizes the distinction but values will visually dominate.

Children per Declaration (Constructors/Methods)

Statistic	Value
Declarations with children	39 (of ~500 sampled) = ~8%
Total children	202
Average children (when present)	5
Max children	54 (Action sum type)

Distribution (of declarations that have children):

- 16 decls have 1-2 children
- 14 decls have 3-5 children
- 6 decls have 6-10 children
- 3 decls have 11+ children

Implication: Most data types have few constructors (1-5). Large sum types (20+ constructors) are rare but exist. Nested circle packing could work - parent circle contains child circles.

LOC per Module

Statistic	Value
Min	5
Max	5,740
Median	66
Mean	139

Distribution:

- 515 modules (37%) have 1-50 LOC
- 314 modules (23%) have 51-100 LOC
- 277 modules (20%) have 101-200 LOC
- 215 modules (16%) have 201-500 LOC
- 60 modules (4%) have 500+ LOC

Implication: LOC and declaration count are correlated but not perfectly. A module with 10 declarations might have 50 LOC or 500 LOC depending on complexity. LOC remains a good sizing metric for treemaps.

Module Naming Hierarchy

Depth distribution:

- Depth 1 (e.g., "Prelude"): 33 modules (2%)
- Depth 2 (e.g., "Data.Array"): 348 modules (25%)
- Depth 3 (e.g., "Data.Array.ST"): 647 modules (47%)
- Depth 4 (e.g., "Web.HTML.HTMLElement.Attributes"): 295 modules (21%)
- Depth 5+: 58 modules (4%)

Top namespaces:

1. Web: 294 modules
2. Data: 275 modules
3. Hylograph: 119 modules (workspace)
4. Test: 101 modules
5. Control: 73 modules

Implication: Module hierarchy is meaningful and could be visualized independently of package structure. Data.* and Control.* span multiple packages. A namespace-based treemap would show different patterns than package-based.

Function Calls (Dependencies)

Statistic	Value
Modules with outgoing calls	330 (24%)
Total call edges	19,210
Average calls per module	13
Max calls from one module	353

Implication: The call graph is sparse - only 24% of modules have recorded outgoing calls (likely limited to workspace packages). When present, dependency links are meaningful but not overwhelming (~13 per module average).

Visualization Design Implications

What's Typical vs. Exceptional

Aspect	Typical	Exceptional
Package size	1-10 modules	50+ modules
Module size	5-20 declarations	100+ declarations
Data type size	1-5 constructors	20+ constructors
Module LOC	50-200	1000+
Module depth	2-3 levels	5+ levels

Recommended Visual Encodings

1. **Package overview (current)**: Treemap works well. Median package (3 modules) and max (197) both render acceptably.
2. **Module detail (current)**: Circle packing for declarations works. Median (6) and max (265) both manageable.
3. **Constructor nesting (proposed)**: Feasible. Most data types have 1-5 constructors. Nested circles or petals would work.
4. **Namespace view (proposed)**: Worth building. Data.* (275 modules) and Control.* (73 modules) would show cross-package patterns.
5. **Function signature visualization**: Lower priority - values dominate but signatures vary widely in complexity.

Views to Consider

1. **Namespace treemap**: Group by Data., *Control.*, etc. instead of package
2. **Declaration graph**: Focus on a single module, show its declarations and their call relationships
3. **Type family view**: Show data types with their constructors as nested structure

4. **Complexity outliers**: Highlight modules/declarations that exceed typical dimensions
5. **Cross-package namespace**: Show how Data.Array in prelude relates to Data.Array.ST in arrays package

Raw Data

The following data was used for this analysis:

- `/api/v2/modules` - 1,381 modules with LOC, declaration counts
- `/api/v2/module-declarations/:id` - Declaration details with children
- `/api/v2/all-calls` - 19,210 function call relationships

Queries executed 2026-02-03 against the minard database via API.