The Select Reference Manual

DSL for array slices, version 1.0

Steve Nunez

Table of Contents

1 Introduction

`Select` is a library for taking slices from array-like objects. The most frequently used form is:

```
(select object selection1 selection2 ...)
```

where each *selection* specifies a set of subscripts along the corresponding axis. The selection specifications are found below.

1.1 Selection Specifiers

Selecting Single Values

A non-negative integer selects the corresponding index, while a negative integer selects an index counting backwards from the last index. For example:

```
(select #(0 1 2 3) 1)                  ; => 1
(select #(0 1 2 3) -2)                 ; => 2
```

These are called *singleton* slices. Each singleton slice drops the dimension: vectors become atoms, matrices become vectors, *etc.*.

Selecting Ranges

`(range start end)` selects subscripts i where start <= i < end. When end is nil, the last index is included (cf. subseq). Each boundary is resolved according to the other rules if applicable, so you can use negative integers:

```
(select #(0 1 2 3) (range 1 3))         ; => #(1 2)
(select #(0 1 2 3) (range 1 -1))        ; => #(1 2)
```

Selecting All Subscripts of a Dimension

`t` selects all subscripts:

```
(select #2A((0 1 2)
            (3 4 5))
         t 1)                           ; => #(1 4)
```

Selecting with a Sequence

Sequences can be used to make specific selections from the object. For example:

```
(select #(0 1 2 3 4 5 6 7 8 9)
(vector (range 1 3) 6 (range -2 -1))) ; => #(1 2 3 6 8 9)
(select #(0 1 2) '(2 2 1 0 0))                ; => #(2 2 1 0 0)
```

Using Bit Vectors as a Mask

Bit vectors can be used to select elements of arrays and sequences as well:

```
(select #(0 1 2 3 4) #*00110)          ; => #(2 3)
```

1.1.1 Extensions

Section 1.1 describes the core functionality. The semantics can be extended, as you will see in the next section. The extensions in this section are provided by the library and prove useful in practice. Their implementation provides good examples of extending the library.

`including` is convenient if you want the selection to include the end of the range:

```
(select #(0 1 2 3) (including 1 2))
```

```
                              ; => #(1 2), cf. (select ... (range 1 3))
```

`nodrop` is useful if you do not want to drop dimensions:

```
(select #(0 1 2 3) (nodrop 2))
                              ; => #(2), cf. (select ... (range 2 3))
```

`head` and `tail` do the obvious:

```
(select #(0 1 2 3) (head 2))            ; => #(0 1)
(select #(0 1 2 3) (tail 2))            ; => #(2 3)
```

All of these are trivial to implement. If there is something you are missing, you can easily extend `select`. Pull request are welcome.

`ref` is a version of `select` that always returns a single element, so it can only be used with singleton slices.

1.2 Select Semantics

Arguments of `select`, except the first one, are meant to be resolved using `canonical-representation`, in the `select-dev` package. If you want to extend `select`, you should define methods for `canonical-representation`. See the source code for the best examples. Below is a simple example that extends the semantics with ordinal numbers.

```
(defmacro define-ordinal-selection (number)
  (check-type number (integer 0))
  `(defmethod select-dev:canonical-representation
       ((axis integer)
(slice (eql ',(intern (format nil "~:@(~:r~)" number)))))
     (assert (< ,number axis))
     (select-dev:canonical-singleton ,number)))

(define-ordinal-selection 1)
(define-ordinal-selection 2)
(define-ordinal-selection 3)

(select #(0 1 2 3 4 5) (range 'first 'third)) ; => #(1 2)
```

Note the following:

- The value returned by `canonical-representation` needs to be constructed using `canonical-singleton`, `canonical-range`, or `canonical-sequence`. You should not use the internal representation directly as it is subject to change.
- You can assume that `axis` is an integer: this is the default. An object may define a more complex mapping (such as, for example, named rows & columns), but unless a method specialized to that is found, `canonical-representation` will just query its dimension (with `axis-dimension`) and try to find a method that works on integers.
- You need to make sure that the subscript is valid, hence the assertion.

2 Systems

The main system appears first, followed by any subsystem dependency.

2.1 `select`

Author Steve Nunez

Home Page
`https://symbolics.github.io/select/`

Source Control
`(:git "git://github.com/symbolics/select")`

Bug Tracker
`https://github.com/Symbolics/select/issues/`

License MS-PL

Description
DSL for array slices.

Long Description
Select is a facility for selecting portions of sequences or arrays. It provides:

An API for taking slices (elements selected by the Cartesian product of vectors of subscripts for each axis) of array-like objects. The most important function is 'select'. Unless you want to define additional methods for 'select', this is pretty much all you need from this library. See the documentation at https://symbolics.github.io/select/ for a tutorial.

An extensible DSL for selecting a subset of valid subscripts. This is useful if, for example, you want to resolve column names in a data frame in your implementation of slice.

A set of utility functions for traversing slices in array-like objects.

Version 1.0

Dependencies
- `alexandria`
- `anaphora`
- `let-plus`

Source [`select.asd`], page 5, (file)

Directory `s:/src/select/`

Components
- [`package.lisp`], page 5, (file)
- [`select-dev.lisp`], page 5, (file)
- [`select.lisp`], page 6, (file)

3 Files

Files are sorted by type and then listed depth-first from the systems components trees.

3.1 Lisp

3.1.1 `select.asd`

Location `select.asd`

Systems [`select`], page 3, (system)

3.1.2 `select/package.lisp`

Parent [`select`], page 3, (system)

Location `package.lisp`

Packages [`slct`], page 9,

3.1.3 `select/select-dev.lisp`

Dependency
[`package.lisp`], page 5, (file)

Parent [`select`], page 3, (system)

Location `select-dev.lisp`

Exported Definitions

- [`select-reserved-symbol?`], page 13, (function)
- [`singleton-representation?`], page 13, (function)

Internal Definitions

- [`all-singleton-representations?`], page 16, (function)
- [`axis-dimension`], page 20, (generic function)
- [`canonical-range`], page 16, (function)
- [`canonical-range`], page 21, (structure)
- [`canonical-range-end`], page 16, (function)
- [`(setf canonical-range-end)`], page 16, (function)
- [`canonical-range-p`], page 17, (function)
- [`canonical-range-start`], page 17, (function)
- [`(setf canonical-range-start)`], page 17, (function)
- [`canonical-representation`], page 20, (generic function)
- [`canonical-representation`], page 20, (method)
- [`canonical-representation`], page 20, (method)
- [`canonical-representation`], page 20, (method)
- [`canonical-representation`], page 20, (method)
- [`canonical-representation`], page 20, (method)
- [`canonical-representation`], page 20, (method)
- [`canonical-representation`], page 21, (method)
- [`canonical-representation`], page 21, (method)
- [`canonical-representations`], page 17, (function)

- [canonical-sequence], page 17, (function)
- [canonical-sequence], page 21, (structure)
- [canonical-sequence-p], page 17, (function)
- [canonical-sequence-vector], page 17, (function)
- [(setf canonical-sequence-vector)], page 17, (function)
- [canonical-singleton], page 17, (function)
- [column-major-setup], page 17, (function)
- [copy-canonical-range], page 17, (function)
- [copy-canonical-sequence], page 18, (function)
- [make-canonical-range], page 18, (function)
- [make-canonical-sequence], page 18, (function)
- [representation-dimension], page 19, (function)
- [representation-dimensions], page 19, (function)
- [representation-initial-value], page 19, (function)
- [representation-iterator], page 19, (function)
- [row-major-setup], page 19, (function)
- [traverse-representations], page 16, (macro)

3.1.4 select/select.lisp

Dependency

[select-dev.lisp], page 5, (file)

Parent [select], page 3, (system)

Location select.lisp

Exported Definitions

- [head], page 13, (function)
- [including], page 13, (function)
- [including], page 15, (structure)
- [mask], page 14, (generic function)
- [mask], page 14, (method)
- [nodrop], page 13, (function)
- [nodrop], page 15, (structure)
- [range], page 13, (function)
- [range], page 16, (structure)
- [ref], page 14, (generic function)
- [ref], page 14, (method)
- [(setf ref)], page 14, (method)
- [(setf ref)], page 14, (generic function)
- [select], page 14, (generic function)
- [select], page 14, (method)
- [select], page 14, (method)
- [(setf select)], page 15, (method)
- [(setf select)], page 14, (generic function)
- [tail], page 13, (function)

Internal Definitions

4 Packages

Packages are listed by definition order.

4.1 slct

SELECT is a facility for selecting portions of sequences or arrays.

Source [package.lisp], page 5, (file)

Nickname select

Use List

- let-plus
- anaphora
- alexandria.1.0.0
- common-lisp

Exported Definitions

- [head], page 13, (function)
- [including], page 13, (function)
- [including], page 15, (structure)
- [mask], page 14, (generic function)
- [mask], page 14, (method)
- [nodrop], page 13, (function)
- [nodrop], page 15, (structure)
- [range], page 13, (function)
- [range], page 16, (structure)
- [ref], page 14, (generic function)
- [ref], page 14, (method)
- [(setf ref)], page 14, (method)
- [(setf ref)], page 14, (generic function)
- [select], page 14, (generic function)
- [select], page 14, (method)
- [select], page 14, (method)
- [(setf select)], page 15, (method)
- [(setf select)], page 14, (generic function)
- [select-reserved-symbol?], page 13, (function)
- [singleton-representation?], page 13, (function)
- [tail], page 13, (function)
- [which], page 15, (generic function)
- [which], page 15, (method)

Internal Definitions

- [all-singleton-representations?], page 16, (function)
- [axis-dimension], page 20, (generic function)
- [canonical-range], page 16, (function)
- [canonical-range], page 21, (structure)

5 Definitions

Definitions are sorted by export status, category, package, and then by lexicographic order.

5.1 Exported definitions

5.1.1 Functions

`head` *COUNT* [Function]
First COUNT indexes.

Package [`slct`], page 9,

Source [`select.lisp`], page 6, (file)

`including` *START END* [Function]
Range, including both ends.

Package [`slct`], page 9,

Source [`select.lisp`], page 6, (file)

`nodrop` *INDEX* [Function]
Select a single index, but do not drop a dimension.

Package [`slct`], page 9,

Source [`select.lisp`], page 6, (file)

`range` *START END* [Function]
Range, including START, excluding END.

Package [`slct`], page 9,

Source [`select.lisp`], page 6, (file)

`select-reserved-symbol?` *SYMBOL* [Function]
Test if SYMBOL has special semantics for SELECTION.

Package [`slct`], page 9,

Source [`select-dev.lisp`], page 5, (file)

`singleton-representation?` *REPRESENTATION* [Function]
Test if a canonical REPRESENTATION is a singleton.

Package [`slct`], page 9,

Source [`select-dev.lisp`], page 5, (file)

`tail` *COUNT* [Function]
Last COUNT indexes.

Package [`slct`], page 9,

Source [`select.lisp`], page 6, (file)

5.1.2 Generic functions

`mask` *PREDICATE SEQUENCE* [Generic Function]
Map sequence into a simple-bit-vector, using 1 when PREDICATE yields true, 0 otherwise.

Package [`slct`], page 9,

Source [`select.lisp`], page 6, (file)

Methods

`mask` *PREDICATE* (*SEQUENCE* `sequence`) [Method]

`ref` *OBJECT* **&rest** *SUBSCRIPTS* [Generic Function]
Return the element of OBJECT specified by SUBSCRIPTS.

Package [`slct`], page 9,

Source [`select.lisp`], page 6, (file)

Writer [`(setf ref)`], page 14, (generic function)

Methods

`ref` (*ARRAY* `array`) **&rest** *SUBSCRIPTS* [Method]

`(setf ref)` *VALUE OBJECT* **&rest** *SUBSCRIPTS* [Generic Function]
Stores VALUE into the place specified by SUBSCRIPTS.

Package [`slct`], page 9,

Source [`select.lisp`], page 6, (file)

Reader [`ref`], page 14, (generic function)

Methods

`(setf ref)` *VALUE* (*ARRAY* `array`) **&rest** *SUBSCRIPTS* [Method]

`select` *OBJECT* **&rest** *SELECTIONS* [Generic Function]
Return the slices of OBJECT specified by SELECTIONS.

Package [`slct`], page 9,

Source [`select.lisp`], page 6, (file)

Writer [`(setf select)`], page 14, (generic function)

Methods

`select` (*LST* `list`) **&rest** *SELECTIONS* [Method]
Select from LST the subscripts or range specified in SELECTIONS. SELECTIONS must be a VECTOR, LIST or RANGE.

`select` (*ARRAY* `array`) **&rest** *SELECTIONS* [Method]
Return the SELECTIONS in the given ARRAY.

`(setf select)` *VALUE OBJECT* **&rest** *SELECTIONS* [Generic Function]
Stores VALUES into the locations given by SELECTIONS.

Package [`slct`], page 9,

Source [`select.lisp`], page 6, (file)

Reader [`select`], page 14, (generic function)

Methods

`(setf select)` *VALUE* (*ARRAY* `array`) **&rest** *SELECTIONS* [Method]

`which` *PREDICATE SEQUENCE* [Generic Function]
Return an index of the positions in SEQUENCE which satisfy PREDICATE.

Package [`slct`], page 9,

Source [`select.lisp`], page 6, (file)

Methods

`which` *PREDICATE* (*SEQUENCE* `sequence`) [Method]

5.1.3 Structures

`including` () [Structure]
Range, including both ends.

Package [`slct`], page 9,

Source [`select.lisp`], page 6, (file)

Direct superclasses
`structure-object` (structure)

Direct methods
[`canonical-representation`], page 20, (method)

Direct slots

`start` [Slot]
Readers [`including-start`], page 18, (function)
Writers [`(setf including-start)`], page 18, (function)

`end` [Slot]
Readers [`including-end`], page 18, (function)
Writers [`(setf including-end)`], page 18, (function)

`nodrop` () [Structure]
Select a single index, but don't drop a dimension.

Package [`slct`], page 9,

Source [`select.lisp`], page 6, (file)

Direct superclasses
`structure-object` (structure)

Direct methods
[`canonical-representation`], page 20, (method)

Direct slots

`index` [Slot]
Readers [`nodrop-index`], page 19, (function)
Writers [`(setf nodrop-index)`], page 19, (function)

`range` () [Structure]
Range, including start, excluding end.

Package [`slct`], page 9,

Source [`select.lisp`], page 6, (file)

Direct superclasses
`structure-object` (structure)

Direct methods
[`canonical-representation`], page 20, (method)

Direct slots

`start` [Slot]
Readers [`range-start`], page 19, (function)
Writers [`(setf range-start)`], page 19, (function)

`end` [Slot]
Readers [`range-end`], page 19, (function)
Writers [`(setf range-end)`], page 19, (function)

5.2 Internal definitions

5.2.1 Macros

`traverse-representations` (*SUBSCRIPTS REPRESENTATIONS* **&key** *INDEX SETUP*) **&body** *BODY* [Macro]
Loops over all possible subscripts in REPRESENTAITONS, making them available in SUBSCRIPTS during the execution of BODY. The iterator is constructed using the function SETUP (see for example ROW-MAJOR-SETUP). When INDEX is given, a variable with that name is provided, containing an index that counts iterations.

Package [`slct`], page 9,

Source [`select-dev.lisp`], page 5, (file)

5.2.2 Functions

`all-singleton-representations?` *REPRESENTATIONS* [Function]
Test if all canonical representations are singletons.

Package [`slct`], page 9,

Source [`select-dev.lisp`], page 5, (file)

`canonical-range` *START END* [Function]
Canonical representation of a contiguous set of array indices from START (inclusive) to END (exclusive).

Package [`slct`], page 9,

Source [`select-dev.lisp`], page 5, (file)

`canonical-range-end` *INSTANCE* [Function]
`(setf canonical-range-end)` *VALUE INSTANCE* [Function]

Package [`slct`], page 9,

Source [`select-dev.lisp`], page 5, (file)

`canonical-range-p` *OBJECT* [Function]

Package [slct], page 9,

Source [select-dev.lisp], page 5, (file)

`canonical-range-start` *INSTANCE* [Function]
`(setf canonical-range-start)` *VALUE INSTANCE* [Function]

Package [slct], page 9,

Source [select-dev.lisp], page 5, (file)

`canonical-representations` *AXES SELECTIONS* [Function]
Return the canonical representations of SELECTIONS given the corresponding AXES, checking for matching length.

Package [slct], page 9,

Source [select-dev.lisp], page 5, (file)

`canonical-sequence` *SEQUENCE* [Function]
Canonical representation of array indexes from canonical-sequence SEQUENCE.

May share structure. Vectors of the upgraded type of (SIMPLE-ARRAY ARRAY-INDEX (*)) are preferred for efficiency, otherwise they are coerced.

Package [slct], page 9,

Source [select-dev.lisp], page 5, (file)

`canonical-sequence-p` *OBJECT* [Function]

Package [slct], page 9,

Source [select-dev.lisp], page 5, (file)

`canonical-sequence-vector` *INSTANCE* [Function]
`(setf canonical-sequence-vector)` *VALUE INSTANCE* [Function]

Package [slct], page 9,

Source [select-dev.lisp], page 5, (file)

`canonical-singleton` *INDEX* [Function]
Canonical representation of a singleton index (a nonnegative integer, which is a valid array index).

Package [slct], page 9,

Source [select-dev.lisp], page 5, (file)

`column-major-setup` *REPRESENTATIONS TERMINATOR* [Function]
Return SUBSCRIPTS (a list) and ITERATOR (a closure, no arguments) that increments the contents of SUBSCRIPTS in column-major order. TERMINATOR is called when all subscripts have been visited.

Package [slct], page 9,

Source [select-dev.lisp], page 5, (file)

`copy-canonical-range` *INSTANCE* [Function]

Package [slct], page 9,

Source [select-dev.lisp], page 5, (file)

`copy-canonical-sequence` *INSTANCE* [Function]

Package [slct], page 9,

Source [select-dev.lisp], page 5, (file)

`copy-including` *INSTANCE* [Function]

Package [slct], page 9,

Source [select.lisp], page 6, (file)

`copy-nodrop` *INSTANCE* [Function]

Package [slct], page 9,

Source [select.lisp], page 6, (file)

`copy-range` *INSTANCE* [Function]

Package [slct], page 9,

Source [select.lisp], page 6, (file)

`including-end` *INSTANCE* [Function]
`(setf including-end)` *VALUE INSTANCE* [Function]

Package [slct], page 9,

Source [select.lisp], page 6, (file)

`including-p` *OBJECT* [Function]

Package [slct], page 9,

Source [select.lisp], page 6, (file)

`including-start` *INSTANCE* [Function]
`(setf including-start)` *VALUE INSTANCE* [Function]

Package [slct], page 9,

Source [select.lisp], page 6, (file)

`make-canonical-range` **&key** (*START* **START**) (*END* **END**) [Function]

Package [slct], page 9,

Source [select-dev.lisp], page 5, (file)

`make-canonical-sequence` **&key** (*VECTOR* **VECTOR**) [Function]

Package [slct], page 9,

Source [select-dev.lisp], page 5, (file)

`make-including` **&key** (*START* **START**) (*END* **END**) [Function]

Package [slct], page 9,

Source [select.lisp], page 6, (file)

`make-nodrop` **&key** (*INDEX* **INDEX**) [Function]

Package [slct], page 9,

Source [select.lisp], page 6, (file)

`make-range` **&key** (*START* **START**) (*END* **END**) [Function]

Package [slct], page 9,

Source [select.lisp], page 6, (file)

`nodrop-index` *INSTANCE* [Function]
`(setf nodrop-index)` *VALUE INSTANCE* [Function]

Package [slct], page 9,

Source [select.lisp], page 6, (file)

`nodrop-p` *OBJECT* [Function]

Package [slct], page 9,

Source [select.lisp], page 6, (file)

`range-end` *INSTANCE* [Function]
`(setf range-end)` *VALUE INSTANCE* [Function]

Package [slct], page 9,

Source [select.lisp], page 6, (file)

`range-p` *OBJECT* [Function]

Package [slct], page 9,

Source [select.lisp], page 6, (file)

`range-start` *INSTANCE* [Function]
`(setf range-start)` *VALUE INSTANCE* [Function]

Package [slct], page 9,

Source [select.lisp], page 6, (file)

`representation-dimension` *REPRESENTATION* [Function]
Return the dimension of a canonical-representation, or NIL for singleton selections (they are dropped).

Package [slct], page 9,

Source [select-dev.lisp], page 5, (file)

`representation-dimensions` *REPRESENTATIONS* [Function]
Return a list for the dimensions of canonical representations, dropping singletons.

Package [slct], page 9,

Source [select-dev.lisp], page 5, (file)

`representation-initial-value` *REPRESENTATION* [Function]
Initial value for iteration.

Package [slct], page 9,

Source [select-dev.lisp], page 5, (file)

`representation-iterator` *REPRESENTATION CARRY CONS* [Function]
Return a closure that sets the car of CONS to the next value each time it is called, resetting and calling CARRY when it reaches the end of its range.

Package [slct], page 9,

Source [select-dev.lisp], page 5, (file)

`row-major-setup` *REPRESENTATIONS TERMINATOR* [Function]
Return SUBSCRIPTS (a list) and ITERATOR (a closure, no arguments) that increments the contents of SUBSCRIPTS in row-major order. TERMINATOR is called when all subscripts have been visited.

Package [slct], page 9,

Source [select-dev.lisp], page 5, (file)

5.2.3 Generic functions

`axis-dimension` *AXIS* [Generic Function]
Return the dimension of axis. Needs to be defined for non-integer axes.

Package [`slct`], page 9,

Source [`select-dev.lisp`], page 5, (file)

`canonical-representation` *AXIS SELECTION* [Generic Function]
Canonical representation of SELECTION, given information in AXIS. The default methods use dimensions as AXIS.

Each selection needs to be resolved into a canonical representation, which is either a singleton, a range, or a sequence of subscripts. They should only be constructed with the corresponding CANONICAL-SINGLETION, CANONICAL-RANGE and CANONICAL-SEQUENCE functions.
@c(CANONICAL-REPRESENTATION) needs to ensure that the represented subscripts are valid for the axis.

Unless a specialized method is found, the dimension of the axis is queried with AXIS-DIMENSION and resolution is attempted using the latter. Methods that resolve symbols should test them with SELECT-RESERVED-SYMBOL? and use CALL-NEXT-METHOD.

Package [`slct`], page 9,

Source [`select-dev.lisp`], page 5, (file)

Methods

`canonical-representation` *AXIS* (*SELECTION* `nodrop`) [Method]
The canonical representation for NODROP.

Source [`select.lisp`], page 6, (file)

`canonical-representation` *AXIS* (*SELECTION* `range`) [Method]
The canonical representation for RANGE.

Source [`select.lisp`], page 6, (file)

`canonical-representation` *AXIS* (*SELECTION* `including`) [Method]
The canonical representation for INCLUDING.

Source [`select.lisp`], page 6, (file)

`canonical-representation` *AXIS SELECTION* [Method]

`canonical-representation` *AXIS* (*CANONICAL-RANGE* `canonical-range`) [Method]

`canonical-representation` *AXIS* (*CANONICAL-SEQUENCE* `canonical-sequence`) [Method]

`canonical-representation` (*AXIS* `integer`) (*SLICE* `null`) [Method]

`canonical-representation` (*AXIS* `integer`) (*SELECTION* `integer`) [Method]

`canonical-representation` *AXIS* (*SELECTION* `sequence`) [Method]

`canonical-representation` (*AXIS* `integer`) (*SELECTION* `(eql t)`) [Method]

`canonical-representation` *AXIS* (*SELECTION* `bit-vector`) [Method]

5.2.4 Structures

`canonical-range` () [Structure]

Canonical representation of a contiguous set of array indices from START (inclusive) to END (exclusive).

Package [`slct`], page 9,

Source [`select-dev.lisp`], page 5, (file)

Direct superclasses

`structure-object` (structure)

Direct methods

[`canonical-representation`], page 20, (method)

Direct slots

`start` [Slot]

Type `alexandria.1.0.0:array-index`

Readers [`canonical-range-start`], page 17, (function)

Writers [`(setf canonical-range-start)`], page 17, (function)

`end` [Slot]

Type `alexandria.1.0.0:array-index`

Readers [`canonical-range-end`], page 16, (function)

Writers [`(setf canonical-range-end)`], page 16, (function)

`canonical-sequence` () [Structure]

Canonical representation of a sequence of array indexes.

Package [`slct`], page 9,

Source [`select-dev.lisp`], page 5, (file)

Direct superclasses

`structure-object` (structure)

Direct methods

[`canonical-representation`], page 20, (method)

Direct slots

`vector` [Slot]

Type `(simple-array alexandria.1.0.0:array-index (*))`

Readers [`canonical-sequence-vector`], page 17, (function)

Writers [`(setf canonical-sequence-vector)`], page 17, (function)

6 Conclusion

`Select` was originally called slice (`https://github.com/tpapp/cl-slice`) and written by Tamas K. Papp. Since it was abandoned in 2017 (`https://tpapp.github.io/post/orphaned-lisp-libraries/`), I have taken it over to be part of a rebooted Common Lisp statistics library. Changes in this version include:

Documentation Improvements

- Move to HTML based documentation system
- Docs now on github.io

Test Improvements

- Ported to FiveAM and refactored
- Improved test coverage
- Added failure messages to aid debugging
- Added tests for selection iteration

Enhancements

- Renamed to 'cons' to 'range'
- Range now handles (range x x) => nil
- Selections work identically on sequences; previously differed between lists and vectors
- Selections may be specified using a list; previously could only be a vector
- Sequence selections now honor fill-pointer, if any

Bug Fixes

- Range now handles END = (length <sequence>)
- Selecting from a list no longer drops dimension

Appendix A Indexes

A.1 Concepts

F

L

S

A.2 Functions

(

A

C

F

G

H

I

M

N

R

S

T

W

A.3 Variables

A.4 Data types

C

I

N

P

R

S