

# Package ‘plumbr’

October 14, 2022

**Version** 0.6.10

**Title** Mutable and Dynamic Data Models

**Author** Michael Lawrence, Hadley Wickham

**Depends** R (>= 2.10.0)

**Imports** utils, methods, objectSignals (>= 0.10.2)

**Suggests** plyr, testthat, MASS

**URL** <https://github.com/ggobi/plumbr/wiki>

**Maintainer** Michael Lawrence <michafla@gene.com>

**Description** The base R data.frame, like any vector, is copied upon modification. This behavior is at odds with that of GUIs and interactive graphics. To rectify this, plumbr provides a mutable, dynamic tabular data model. Models may be chained together to form the complex plumbing necessary for sophisticated graphical interfaces. Also included is a general framework for linking datasets; an typical use case would be a linked brush.

**License** GPL (>= 2)

**Collate** 'accessors.r' 'bindings.r' 'coercion.r' 'constructor.r'  
'dimensions.r' 'events.r' 'linking.r' 'mutalist.R' 'names.r'  
'print.r' 'proxy-filter.r' 'selection.r' 'utils.r' 's4.r'  
'import.r' 'globals.r'

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2022-05-06 18:30:02 UTC

## R topics documented:

add_listener . . . . .	2
as.data.frame.mutaframe . . . . .	3
as.list.mutaframe . . . . .	3

as.mutaframe . . . . .	4
changed . . . . .	4
combine_data_events . . . . .	5
DataSelection . . . . .	5
duplex_data_linker . . . . .	6
is.mutaframe . . . . .	6
is_paused . . . . .	7
ItemSelection-class . . . . .	7
match_any_linker . . . . .	9
mutaframe . . . . .	9
mutalist . . . . .	10
notify_listeners . . . . .	12
pause . . . . .	12
proxy_bindings . . . . .	13
raw_binding . . . . .	13
raw_bindings . . . . .	14
RegionSelection-class . . . . .	14
remove_listener . . . . .	15
Selection-class . . . . .	16
shape_changed . . . . .	17
unpause . . . . .	18
variable_names . . . . .	18
\$.mutaframe . . . . .	19

<b>Index</b>	<b>20</b>
--------------	-----------

---

add_listener	<i>Plumbr events</i>
--------------	----------------------

---

## Description

Plumbr data structures send only single event for data changes: `data_changed`. This has a two arguments, `i` and `j`. Either both are `NULL`, indicating a change in the shape of the underlying data, or they give the the locations of changed data values.

## Usage

```
add_listener(mf, callback)
```

## Arguments

<code>mf</code>	muta frame
<code>callback</code>	function with arguments <code>i</code> and <code>j</code>

---

as.data.frame.mutaframe  
*Coercion to data.frame*

---

**Description**

Coerces a mutafame to a data.frame

**Usage**

```
## S3 method for class 'mutafame'  
as.data.frame(x, row.names =  
  rownames(x), optional = FALSE, ...)
```

**Arguments**

x	a mutafame
row.names	character vector of rownames, defaults to rownames of x
optional	see <a href="#">as.data.frame</a>
...	see <a href="#">as.data.frame</a>

**Value**

a data.frame

---

as.list.mutaframe      *Coercion to list*

---

**Description**

Coerces a mutafame to a list

**Usage**

```
## S3 method for class 'mutafame'  
as.list(x, ...)
```

**Arguments**

x	a mutafame
...	ignored

**Value**

a list, with one element for each mutafame column

---

as.mutaframe	<i>Coercion to mutaframe</i>
--------------	------------------------------

---

### Description

Coerce an object to a mutaframe. Supported types include `data.frame`, or anything coercible to one.

### Usage

```
as.mutaframe(x, ...)
```

```
## S3 method for class 'mutaframe'
```

```
as.mutaframe(x, ...)
```

```
## S3 method for class 'data.frame'
```

```
as.mutaframe(x, ...)
```

```
## Default S3 method:
```

```
as.mutaframe(x, ...)
```

### Arguments

<code>x</code>	the object to coerce
<code>...</code>	arguments passed to methods

### Value

a mutaframe

---

changed	<i>Get the 'changed' signal</i>
---------	---------------------------------

---

### Description

Get the 'changed' signal

### Usage

```
changed(mf)
```

### Arguments

<code>mf</code>	a mutaframe
-----------------	-------------

---

combine\_data\_events     *Combine list of events into single event.*

---

**Description**

If any event is a shape\_changed event, return it. Otherwise, take the unique elements of the union of all element changes.

**Usage**

```
combine_data_events(events)
```

**Arguments**

events             a list of event parameters

**Value**

a unified event

---

DataSelection             *Selection in Data*

---

**Description**

Implement a selection model against a dataset/pipeline

**Usage**

```
DataSelection(data, column = 1L)
```

**Arguments**

data                [mutaframe](#) of the dataset/pipeline  
column              Column index of selection variable in data

**Value**

An [ItemSelection](#) reflecting the selection in the data

**Author(s)**

Michael Lawrence

---

`duplex_data_linker`      *Duplex linking*

---

### Description

A utility for creating linking functions that operate in both directions (full duplex).

### Usage

```
duplex_data_linker(delegate, from_data, to_data =
  from_data)
```

### Arguments

<code>delegate</code>	The linking function that performs the mapping, such as <a href="#">match_any_linker</a> .
<code>from_data</code>	A <code>data.frame</code> of keys
<code>to_data</code>	A <code>data.frame</code> of keys

### Details

The generated linker function takes two arguments: `from_selection` and `new_selection`. If `new_selection` is specified, `new_selection` is mapped from `to_data` to `from_data`. Otherwise, `from_selection` is mapped from `from_data` to `to_data`.

### Value

A two-way linking function as described in the details.

### Author(s)

Michael Lawrence

---

`is.mutaframe`      *Test for mutaframes*

---

### Description

Tests whether an object is a mutaframe

### Usage

```
is.mutaframe(x)
```

### Arguments

<code>x</code>	an object to check
----------------	--------------------

**Value**

TRUE if x is an instance of a class that inherits from `mutaframe`; otherwise, FALSE

---

is_paused	<i>Is a mutaframe currently paused?</i>
-----------	---

---

**Description**

Is a mutaframe currently paused?

**Usage**

```
is_paused(mf)
```

**Arguments**

mf                    a mutaframe

---

ItemSelection-class	<i>The ItemSelection class implements <a href="#">Selection</a> for the very common case of selecting items in a dataset, optionally with weights.</i>
---------------------	--

---

**Description**

The ItemSelection class implements [Selection](#) for the very common case of selecting items in a dataset, optionally with weights.

**Constructor**

- `ItemSelection(delegate = NULL)`: Constructs an ItemSelection object with the underlying selection provided by `delegate`, which may be a function or any other R object. If it is not a function, `delegate` must support the coercions described in the next section. A good example would be a logical vector. However, `delegate` is usually a function that is invoked whenever the selection is stored or retrieved. If the function is called with no arguments, it should return the selection. Otherwise, the argument is the new selection status, and the function should store it. This is the same semantic as [active bindings](#). This dynamic functionality allows proxying of other Selection objects or external sources, such as a selection model from a GUI toolkit.

## Interpreting the Selection

Any R object can represent the underlying selection, so for simplicity we recommend that the client interpret the selection through coercion. Each of these simply delegate to the underlying selection object, which will need to support all of them for consistency. The following coercions are supported, where `x` is a `ItemSelection` instance:

- `which(x)`: integer indices of the selected items.
- `as.logical(x)`: TRUE where selected.
- `as.integer(x)`: usually 0L (unselected) or 1L (selected), but in general it is a weighting of the selection.
- `as.numeric(x)`: similar to `as.integer`, except with real values.
- `as.factor(x)`: ordinarily this will have two levels, FALSE and TRUE, although it could have more, which confers support for multinary selections.

## Supported Selection Calculus

All operations mentioned in [Selection](#) are supported: add, subtract, toggle, intersect.

## Author(s)

Michael Lawrence

## See Also

[Selection](#) for the rest of the details.

## Examples

```
## Assume we have a dataset:
data(Cars93, package="MASS")
mf <- mtaframe(Cars93)
mf$.color <- "gray"
## First step is to create a base selection
sel <- ItemSelection()
## Now, link that selection to other cases in same dataset by some variable
linked_sel <- sel$link(match_any_linker(Cars93["Manufacturer"]))
## Finally, scale that linked selection to the data
linked_sel$scale(function(x, d) {
  d[as.logical(x), ".color"] <- "red"
}, mf)
## To test, select some cases
cases <- rep(FALSE, nrow(mf))
cases[seq(1, 10, 2)] <- TRUE
sel$replace(cases)
```



---

match_any_linker	<i>match_any_linker</i>
------------------	-------------------------

---

### Description

Linking functions return a logical vector, with the TRUE elements indicating rows in the data that are linked.

### Usage

```
match_any_linker(from_data, to_data = from_data)
```

### Arguments

from_data	A data.frame-like object containing the keys for linking the corresponding rows to rows in to_data
to_data	A data.frame-like object containing the keys that will be matched against the keys in from_data

### Details

The match\_any\_linker function links rows in from\_data to rows in to\_data that share the same key.

By convention, a key is defined as the combination of the values in every column of from\_data and to\_data. Thus, from\_data and to\_data should contain only the columns necessary for key generation. They should not be an entire dataset.

### Value

a logical vector, indicating which from\_data rows are linked

### Author(s)

Michael Lawrence

---

mutaframe	<i>Create a mutaframe, a mutable data.frame</i>
-----------	---

---

### Description

Create a mutaframe, a mutable data.frame

### Usage

```
mutaframe(..., row.names = NULL)
```

**Arguments**

... Objects to coerce to a mutafame and combine column-wise  
 row.names optional, the character vector of row names

**Value**

a mutafame

---

mutalist	<i>mutalist</i>
----------	-----------------

---

**Description**

The mutalist is a mutable list. Modifications to a mutalist occur by a reference semantic. Otherwise, it should act like an ordinary R list and provides a similar API. If anything is found missing, please inform the authors.

**Usage**

```
mutalist(...)

## S3 method for class 'mutalist'
length(x)

## S3 replacement method for class 'mutalist'
names(x, ...) <- value

## S3 method for class 'mutalist'
names(x)

## S3 method for class 'mutalist'
x[[i, j, ...]]

## S3 replacement method for class 'mutalist'
x[[i, j, ...]] <- value

## S3 replacement method for class 'mutalist'
x$name <- value

## S3 method for class 'mutalist'
x[i, j, ..., drop]

## S3 replacement method for class 'mutalist'
x[i, j, ...] <- value

## S3 method for class 'mutalist'
```

```

head(x, n = 6L, ...)

## S3 method for class 'mutalist'
tail(x, n = 6L, ...)

## S3 method for class 'mutalist'
c(x, ..., recursive = FALSE)

## S3 method for class 'mutalist'
lapply(X, FUN, ...)

## S3 method for class 'mutalist'
as.list(x, ...)

## S3 method for class 'mutalist'
as.data.frame(x, ...)

## S3 method for class 'mutalist'
unlist(x, recursive = TRUE, use.names
= TRUE)

mutalist2env(x, envir = new.env(hash, parent, size),
parent = parent.frame(), hash = FALSE, size = 29L)

## S3 method for class 'mutalist'
rev(x)

## S3 method for class 'mutalist'
rep(x, ...)

## S3 method for class 'mutalist'
print(x, ...)

```

### Arguments

...	elements to include in the list or arguments passed to methods
x	a mutalist
value	replacement value
i	element indices
j	unused
name	element name
drop	unused
n	number of elements in subset
recursive	whether to perform recursively
X	a mutalist
FUN	a function to apply over the elements

use.names	whether to preserve the names
envir	environment to populate
parent	parent for new environment, if created
hash	whether to hash the new environment
size	initial size of hash table

**Value**

a new mutalist

**Author(s)**

Michael Lawrence

---

notify_listeners	<i>Notify listeners that data has changed.</i>
------------------	--

---

**Description**

Notify listeners that data has changed.

**Usage**

```
notify_listeners(mf, i, j)
```

**Arguments**

mf	mutaframe
i, j	row and column indices

---

pause	<i>Pause (cache) events.</i>
-------	------------------------------

---

**Description**

When a mutaframe is paused, it accumulates events without passing them on. When unpaused, it accumulates all events into a single event and passes it on.

**Usage**

```
pause(mf)
```

**Arguments**

mf	mutaframe
----	-----------

**Details**

This is a performance optimisation for when you expect many changes: pause the mutaframe, perform all the changes and then unpause.

---

proxy_bindings	<i>Generate binding for proxies.</i>
----------------	--------------------------------------

---

**Description**

Generate binding for proxies.

**Usage**

```
proxy_bindings(mf, j = names(mf))
```

**Arguments**

mf	mutaframe to inherit from
j	columns to generate bindings for

---

raw_binding	<i>Generate binding for raw values</i>
-------------	--

---

**Description**

Generate binding for raw values

**Usage**

```
raw_binding(mf, name, data)
```

**Arguments**

mf	mutaframe
name	name
data	vector to store

**Value**

named list of binding functions

---

raw_bindings	<i>Generate binding for raw values</i>
--------------	--

---

### Description

Generate binding for raw values

### Usage

```
raw_bindings(mf, data)
```

### Arguments

mf	mutaframe
data	list of values

### Value

named list of binding functions

---

RegionSelection-class *The ItemSelection class implements [Selection](#) for the selection of 1D and 2D regions in plot/data space.*

---

### Description

The ItemSelection class implements [Selection](#) for the selection of 1D and 2D regions in plot/data space.

### Constructor

- `RegionSelection(delegate = NULL)`: Constructs an `RegionSelection` object with the underlying selection provided by `delegate`, which may be a function or any other R object. If it is not a function, `delegate` must support coercion to a matrix as described in the next section. However, `delegate` is usually a function that is invoked whenever the selection is stored or retrieved. If the function is called with no arguments, it should return the selection. Otherwise, the argument is the new selection status, and the function should store it. This is the same semantic as [active bindings](#). This dynamic functionality allows proxying of other `Selection` objects or external sources, such as a selection model from a GUI toolkit.

## Interpreting the Selection

Any R object can represent the underlying selection, so for simplicity we recommend that the client interpret the selection through coercion. Currently, there is only one supported coercion of `RegionSelection`:

- `as.matrix(x)`: returns a matrix with a column for each dimension and a row for each point. In the 2D case, the points describe one or more polygons. As with the `polygon` function, polygons are separated by rows of NA, and the last point is connected with the first. In the 1D case, the single column might encode, for example, selections of factor levels in an area plot.

We will probably need to add more coercions as use cases arise. This is still very preliminary.

## Supported Selection Calculus

For now, `RegionSelection` only supports the add operation described in the documentation for [Selection](#).

## Author(s)

Michael Lawrence

## See Also

[Selection](#) for the rest of the details.

## Examples

```
## forthcoming
```

---

<code>remove_listener</code>	<i>Remove a listener, identified by the ID returned by <code>add_listener</code>.</i>
------------------------------	---

---

## Description

Remove a listener, identified by the ID returned by `add_listener`.

## Usage

```
remove_listener(mf, id)
```

## Arguments

<code>mf</code>	mutaframe
<code>id</code>	value returned by <code>add_listener</code> when originally connecting the handler

---

Selection-class	<i>Selection</i>
-----------------	------------------

---

## Description

A virtual base class for data models that store a selection, which might be of items, regions, or whatever. Clients can register handlers for selection changes and can create proxy models to transform selections, link across datasets and map selections to actions on the data.

**This design is preliminary and subject to change.**

## Interpreting The Selection

Internally, the selection may be stored as any object, including as a function that is invoked whenever the selection is stored or retrieved. The function allows dynamic mapping of selections. Due to this generality, the client should not access the selection directly. Instead, it should explicitly coerce the selection object to an interpretable representation. The set of supported coercions depends on the subclass. For example, `ItemSelection` has a `as.logical` method that coerces it to a logical vector, where an element is TRUE if the corresponding element in the dataset is selected.

## Responding to Selection Changes

Whenever the selection is changed, the `changed` signal is emitted. The signal has zero arguments. See the `objectSignals` package for details on using signals.

Eventually, a selection leads to the execution of some action by the application. In interactive graphics, that action usually involves scaling/transforming the selection to a modification on the data. The `x$scale(scaler, data)` method tries to facilitate these operations. All it does is create a handler for the `changed` signal on `x` that passes `x` and `data` to the function `scaler`, which implements the change.

## The Selection Calculus

Since any type of object can represent a selection, setting the selection has very few constraints. There are several ways to modify the selection. Not all of them will be supported by every subclass. In the code snippets below, `x` represents a `Selection` object and `selection` represents the primary representation of a selection, like a logical vector.

**Replacement** `x$replace(selection)`: this is supported by all implementations.

**Or/Addition** `x$add(selection)`: the result contains the union of the original selection and `selection`.

**Setdiff/Subtract** `x$subtract(selection)`: the result contains the original selection except that indicated by `selection`.

**And/Intersect** `x$intersect(selection)`: the result contains the intersection of the original selection and `selection`.

**Xor/Toggle** `x$toggle(selection)`: The intersection of the original selection and `selection` is deselected, that only in `selection` is selected.



## Linking Selections

In interactive graphics, it is often necessary to link selections within and across datasets. The `x$link(linker)` method creates a new `Selection` object that proxies `x` and maps the selection in `x` through `linker`. Changes to the selection in `x` will propagate via `linker` to changes in the proxy. Analogously, the `linker` will pass modifications to the proxy down to `x`.

The `linker` may be provided as an integer vector, like that returned by `match`, but it is usually a function, as that allows very general linking strategies. As an example, let us consider a simple linker between two datasets based on key matching. We assume that the keys, `source_keys` and `dest_keys`, are in the enclosure of our linker function.

```
function(source_selection, new_dest_value) {
  if (missing(new_dest_value))
    dest_keys
  else source_keys
}
```

The linker function takes one or two arguments, depending on whether the selection is being retrieved or stored. When the selection is being retrieved, `source_selection` is passed as the only argument. The duty of the linker is then to retrieve the underlying selection from `source_selection` (through coercion, see above) and figure out which keys in the destination selection match the selected source keys. The `new_dest_value` argument is provided whenever the selection is being stored/set. In that case, the analogous operation is performed, in the opposite direction. The symmetry here is fairly obvious, and `duplex_data_linker` is a utility for facilitating the implementation of such two-way linking functions.

## Author(s)

Michael Lawrence

## See Also

The `ItemSelection` and `RegionSelection` subclasses, which have examples.

---

shape\_changed

*Is the event a shape changed event?*

---

## Description

Is the event a shape changed event?

## Usage

```
shape_changed(i, j)
```

**Arguments**

i	col index
j	row index

---

unpause	<i>Unpause (reply) events.</i>
---------	--------------------------------

---

**Description**

Unpause (reply) events.

**Usage**

unpause(mf)

**Arguments**

mf	mutaframe
----	-----------

---

variable_names	<i>Make valid variable names</i>
----------------	----------------------------------

---

**Description**

Make valid variable names

**Usage**

variable\_names(var\_names)

**Arguments**

var_names	variable names
-----------	----------------

**Description**

These functions extract, subset and replace data in a mutaframe. For the most part, these behave much like those for `data.frame`.

**Arguments**

<code>x</code>	A mutaframe
<code>name</code>	Name of the column to extract
<code>i</code>	The row indices
<code>j</code>	The column indices
<code>...</code>	Arguments passed to methods
<code>value</code>	The replacement column
<code>drop</code>	If TRUE and the result of subsetting is a single column or row, that column or row is extracted as the result. By default, this is TRUE if the result has one column.

**Details**

The subset function, `[`, does not copy the data; it establishes a dynamic filter.

Replacing an existing variable will pass the replacement data up the reverse pipeline, towards the root. When defining a new variable, the variable is stored in the current mutaframe; not at the root.

**Value**

The selected column

The selected column

A dynamic, filtering mutaframe

# Index

`[.mutaframe ($.mutaframe)`, 19  
`[.mutalist (mutalist)`, 10  
`[<-.mutaframe ($.mutaframe)`, 19  
`[<-.mutalist (mutalist)`, 10  
`[[.mutaframe ($.mutaframe)`, 19  
`[[.mutalist (mutalist)`, 10  
`[[<-.mutaframe ($.mutaframe)`, 19  
`[[<-.mutalist (mutalist)`, 10  
`$.mutaframe`, 19  
`$.mutalist (mutalist)`, 10  
`$<-.mutaframe ($.mutaframe)`, 19  
`$<-.mutalist (mutalist)`, 10

`active bindings`, 7, 14  
`add_listener`, 2, 15  
`as.data.frame`, 3  
`as.data.frame.mutaframe`, 3  
`as.data.frame.mutalist (mutalist)`, 10  
`as.factor`, ItemSelection-method (ItemSelection-class), 7  
`as.integer`, ItemSelection-method (ItemSelection-class), 7  
`as.list.mutaframe`, 3  
`as.list.mutalist (mutalist)`, 10  
`as.logical`, ItemSelection-method (ItemSelection-class), 7  
`as.matrix`, RegionSelection-method (RegionSelection-class), 14  
`as.mutaframe`, 4  
`as.numeric`, ItemSelection-method (ItemSelection-class), 7  
`as.vector`, ItemSelection-method (ItemSelection-class), 7

`c.mutalist (mutalist)`, 10  
`changed`, 4  
`class:ItemSelection` (ItemSelection-class), 7  
`class:RegionSelection` (RegionSelection-class), 14

`class:Selection (Selection-class)`, 16  
`coerce`, ItemSelection, factor-method (ItemSelection-class), 7  
`coerce`, ItemSelection, integer-method (ItemSelection-class), 7  
`coerce`, ItemSelection, logical-method (ItemSelection-class), 7  
`coerce`, ItemSelection, numeric-method (ItemSelection-class), 7  
`coerce`, ItemSelection, vector-method (ItemSelection-class), 7  
`coerce`, RegionSelection, matrix-method (RegionSelection-class), 14  
`combine_data_events`, 5

`DataSelection`, 5  
`duplex_data_linker`, 6, 17

`head.mutalist (mutalist)`, 10

`is.mutaframe`, 6  
`is_paused`, 7  
`ItemSelection`, 5, 16, 17  
`ItemSelection (ItemSelection-class)`, 7  
`ItemSelection-class`, 7

`lapply.mutalist (mutalist)`, 10  
`length.mutalist (mutalist)`, 10

`match`, 17  
`match_any_linker`, 6, 9  
`mutaframe`, 5, 9  
`mutaframe-class (mutaframe)`, 9  
`mutalist`, 10  
`mutalist-class (mutalist)`, 10  
`mutalist2env (mutalist)`, 10

`names.mutalist (mutalist)`, 10  
`names<-.mutalist (mutalist)`, 10  
`notify_listeners`, 12

pause, [12](#)  
polygon, [15](#)  
print.mutalist (mutalist), [10](#)  
proxy\_bindings, [13](#)  
  
raw\_binding, [13](#)  
raw\_bindings, [14](#)  
RegionSelection, [17](#)  
RegionSelection  
    (RegionSelection-class), [14](#)  
RegionSelection-class, [14](#)  
remove\_listener, [15](#)  
rep.mutalist (mutalist), [10](#)  
rev.mutalist (mutalist), [10](#)  
  
Selection, [7](#), [8](#), [14](#), [15](#)  
Selection-class, [16](#)  
shape\_changed, [17](#)  
  
tail.mutalist (mutalist), [10](#)  
  
unlist.mutalist (mutalist), [10](#)  
unpause, [18](#)  
  
variable\_names, [18](#)  
  
which, ItemSelection-method  
    (ItemSelection-class), [7](#)