Package ‘r2r’

October 14, 2022

Title  R-Object to R-Object Hash Maps
Version  0.1.1
Description  Implementation of hash tables (hash sets and hash maps) in R, featuring arbitrary R objects as keys, arbitrary hash and key-comparison functions, and customizable behaviour upon queries of missing keys.
License  GPL (>= 3)
Encoding  UTF-8
RoxygenNote  7.1.1
Imports  digest, rlang
URL  https://github.com/vgherard/r2r
BugReports  https://github.com/vgherard/r2r/issues
Suggests  knitr, covr, testthat (>= 3.0.0), rmarkdown, microbenchmark, hash
Config/testthat/edition  3
VignetteBuilder  knitr
NeedsCompilation  no
Author  Valerio Gherardi [aut, cre] (<https://orcid.org/0000-0002-8215-3013>)
Maintainer  Valerio Gherardi <vgherard@sissa.it>
Repository  CRAN
Date/Publication  2021-07-06 09:10:12 UTC

R topics documented:

  compare_fn ................................................................. 2
  default ................................................................. 3
  default_hash_fn .......................................................... 4
  delete ................................................................. 4
  hashtable .............................................................. 5
  hashtable_methods ..................................................... 7
compare_fn

Get key comparison function of an hash table

Description

Returns the key comparison function of an hash table (hashset or hashmap).

Usage

compare_fn(x)

```r
## S3 method for class 'r2r_hashtable'
compare_fn(x)
```

Arguments

- `x` an hashset or hashmap.

Value

- a function.

Author(s)

Valerio Gherardi

Examples

```r
s <- hashset()
compare_fn(s)
```
Description
These generics are used to get or set the default value of an hashmap, optionally returned upon query of a missing key.

Usage

default(x)
default(x) <- value

## S3 method for class 'r2r_hashmap'
default(x)

## S3 replacement method for class 'r2r_hashmap'
default(x) <- value

Arguments

x an hashmap.
value an arbitrary R object. Default value to be associated to missing keys in the hashmap.

Details
For more details, see the hashtable documentation page.

Value
an arbitrary R object.

Author(s)
Valerio Gherardi

Examples

m <- hashmap()
default(m)
default(m) <- 840
**default_hash_fn**  
*String hashes for arbitrary R objects*

### Description

generates string hashes for arbitrary R objects as follows. This is the default hash function used by hashsets and hashmaps objects.

### Usage

default_hash_fn(key)

### Arguments

- **key**: an arbitrary R object.

### Details

If `key` is an atomic vector (as tested by `is.atomic(key)`) of length one, `default_hash_fn(key)` simply coerces the input to character. For more complex inputs, the function calls `digest(key)` from the `digest` package.

### Value

a character vector of length one. Hash digest of `key`.

### Author(s)

Valerio Gherardi

---

**delete**  
*Delete keys or key/value pairs from an hash table.*

### Description

These generics are used for deleting a single key or key/value pair from an hashset or hashmap, respectively.

### Usage

deleate(x, key)

```r
## S3 method for class 'r2r_hashmap'
deleate(x, key)

## S3 method for class 'r2r_hashset'
deleate(x, key)
```
**hashtable**

**Arguments**

- `x`: an `hashset` or `hashmap`.
- `key`: an arbitrary R object. Key to be deleted from the hash table.

**Value**

`NULL`, invisibly.

**Author(s)**

Valerio Gherardi

**Examples**

```r
s <- hashset(1, 2, 3)
delete(s, 3)
s[[3]]
```

---

**Description**

Objects of class `hashmap` and `hashset` store collections of key/value pairs (hashmap), or just keys (hashset), providing constant time read and write operations. Both the keys and the optional values can be arbitrary R objects. `hashmap`s and `hashset`s provide an R implementation of hash tables.

See `hashtable_methods` for an overview of the available methods for `hashmap` and `hashset` class objects. Note that both these classes have a common parent class `hashtable`, from which they can also inherit S3 methods.

**Usage**

```r
hashmap(
  ..., 
  hash_fn = default_hash_fn,
  compare_fn = identical,
  key_preproc_fn = identity,
  on_missing_key = "default",
  default = NULL
)
```

```r
hashset(
  ..., 
  hash_fn = default_hash_fn,
  compare_fn = identical,
  key_preproc_fn = identity
)
```
Arguments

... these arguments can be used to specify a set of initial elements to be inserted in the hashmap or hashset. For hashmap(), each of these should be a list of two elements (a key-value pair).

hash_fn the (string valued) hash function applied to keys. Required for advanced use only; see Details.

compare_fn the (boolean valued) comparison function used for testing key equality. Required for advanced use only; see Details.

key_preproc_fn key pre-processing function applied to keys before hashing and comparison. Required for advanced use only; see Details.

on_missing_key either "throw" or "default". In the second case, an exception is thrown upon query of a missing key; otherwise, a default value (specified through the default argument) is returned.

default default value associated with missing keys. This will be returned only if on_missing_key is equal to "default".

Details

hashmaps and hashsets implement hash tables, building on top of base R built-in environments, which by themselves are, essentially, string -> R object hash maps. In order to handle keys of non-string type, a string valued hash function default_hash_fn() is provided, which leverages on digest() for handling arbitrary R object keys.

By default, key equality is tested through identical(). For some use cases, it may be sensible to employ a different comparison function, which can be assigned through the compare_fn argument. In this case, one must also make sure that equal (in the sense of compare_fn()) keys get also assigned the same hashes by hash_fn(). A simple way to ensure this is to use to use a key pre-processing function, to be applied before both key hashing and comparison. The key_preproc_fn argument provides a short-cut to this, by automatically composing both the provided hash_fn() and compare_fn() functions with key_preproc_fn() function. This is illustrated in an example below.

One might also want to set set specific hash and/or key comparison functions for efficiency reasons, e.g. if the default_hash_fn() function produces many collisions between inequivalent keys.

When on_missing_key is equal to "throw", querying a missing key will cause an error. In this case, an rlang abort condition of class "r2r_missing_key" is returned, which can be useful for testing purposes.

Value

a hashmap and a hashset class object for hashmap() and hashset(), respectively.

Author(s)

Valerio Gherardi

See Also

hashtable_methods
Examples

```r
m <- hashmap(
  list("foo", 1),
  list("bar", 1:5),
  list(data.frame(x = letters, y = LETTERS), "baz")
)
m[[ data.frame(x = letters, y = LETTERS) ]]

# Set of character keys, case insensitive.
s <- hashset("A", "B", "C", key_preproc = tolower)
s["a"]
```

Description

This page provides an overview of the available methods for hashmap and hashset objects (and for their common parent class hashtable). We list methods based on the general type of task addressed.

**Basic read/write operations:**
- `insert()`
- `delete()`
- `query()`
- `subsetting_hashtables`: `[[`, `[[<`, `[` and `[<``

**Size of hash table:**
- `length.r2r_hashtable()`

**Other key or value access operations:**
- `keys()`
- `values()`
- `has_key`, `%has_key%`

**Get/set hashtable properties:**
- `hash_fn()`
- `compare_fn()`
- `on_missing_key()`
- `default()`

**Author(s)**

Valerio Gherardi
hash_fn

Get hash function of an hash table

Description

Returns the hash function used for key hashing in an hash table (hashset or hashmap).

Usage

hash_fn(x)

## S3 method for class 'r2r Hashtable'
hash_fn(x)

Arguments

x an hashset or hashmap.

Value

a function.

Author(s)

Valerio Gherardi

Examples

s <- hashset()
hash_fn(s)

has_key

Key existence in hash tables

Description

This generics are used to check whether a key exists in a given hashset or hashmap.
Usage

has_key(x, key)

x %has_key% key

## S3 method for class 'r2r_hashmap'
has_key(x, key)

## S3 method for class 'r2r_hashset'
has_key(x, key)

Arguments

  x          an hashset or hashmap.
  key        an arbitrary R object. Key to be checked for existence in the hash table.

Value

TRUE or FALSE.

Author(s)

Valerio Gherardi

Examples

m <- hashmap(list("a", 1), list("b", 2))
has_key(m, "a")
m %has_key% "b"

---

**insert**  
*Insert keys or key/value pairs into an hash table.*

### Description

These generics are used for inserting a single key or key/value pair into an hashset or hashmap, respectively. For vectorized insertions, see the subsetting_hashtables documentation page.

Usage

insert(x, key, ...)

## S3 method for class 'r2r_hashmap'
insert(x, key, value, ...)

## S3 method for class 'r2r_hashset'
insert(x, key, ...)

**Arguments**

- **x** an hashset or hashmap.
- **key** an arbitrary R object. Key to be inserted into the hash table.
- **...** further arguments passed to or from other methods.
- **value** an arbitrary R object. Value associated to key.

**Value**

key for the hashset method, value for the hashmap method.

**Author(s)**

Valerio Gherardi

**Examples**

```r
s <- hashset()
insert(s, "foo")
s[["foo"]]
```

---

### keys

**List all keys from an hash table**

**Description**

These generics are used for listing all keys registered in an hashset or hashmap, respectively.

**Usage**

```r
keys(x)
```

```r
## S3 method for class 'r2r_hashtable'
keys(x)
```

**Arguments**

- **x** an hashset or hashmap.

**Value**

a list. Registered keys in the hash table `x`.

**Author(s)**

Valerio Gherardi
Examples

```r
s <- hashset(1, 2, 3)
keys(s)
```

Description

Returns the total number of keys in an hash table.

Usage

```r
## S3 method for class 'r2r_hashtable'
length(x)
```

Arguments

- `x` an hashset or hashmap.

Value

an integer. Number of keys in the hash table (or elements in a set).

Author(s)

Valerio Gherardi

Examples

```r
s <- hashset()
insert(s, "foo")
length(s)
```

Description

These generics are used to get or set the behaviour of an hashmap upon query of a missing key (currently, only an hashmap method is implemented).
Usage

```r
on_missing_key(x)

on_missing_key(x) <- value
```

```r
## S3 method for class 'r2r_hashmap'
on_missing_key(x)

## S3 replacement method for class 'r2r_hashmap'
on_missing_key(x) <- value
```

Arguments

- `x` an hashmap.
- `value` a string, either "throw" or "default". Action to be taken upon query of a missing key.

Details

For more details, see the hashtable documentation page.

Value

a string, either "throw" or "default".

Author(s)

Valerio Gherardi

Examples

```r
m <- hashmap()
on_missing_key(m)
on_missing_key(m) <- "throw"
```

query

*Query keys from an hash table.*

Description

These generics are used for querying a single key from an hashset or hashmap, respectively. For vectorized queries, see the subsetting_hashtables documentation page.
Usage

query(x, key)

## S3 method for class 'r2r_hashmap'
query(x, key)

## S3 method for class 'r2r_hashset'
query(x, key)

Arguments

x an hashset or hashmap.
key an arbitrary R object. Key to be queried from the hash table.

Value

TRUE or FALSE, for hashsets. For hashmaps, if the queried key exists in the hash table, returns the associated value (an a priori arbitrary R object); otherwise, behaves as specified by on_missing_key(x) (see also hashtable).

Author(s)

Valerio Gherardi

Examples

s <- hashset(1, 2, 3)
query(s, 3)

---

subsetting_hashtables Subsetting hashsets and hashmap

Description

Subsetting operators `[[` and `[' for hashsets and hashmaps provide an equivalent syntax for the basic read/write operations performed by insert(), delete() and query().

Usage

## S3 method for class 'r2r_hashmap'
x[[i]]

## S3 method for class 'r2r_hashmap'
x[i]

## S3 replacement method for class 'r2r_hashmap'
x[[i]] <- value
## S3 replacement method for class 'r2r_hashmap'

```r
x[i] <- value
```

## S3 method for class 'r2r_hashset'

```r
x[[i]]
```

## S3 method for class 'r2r_hashset'

```r
x[i]
```

## S3 replacement method for class 'r2r_hashset'

```r
x[[i]] <- value
```

## S3 replacement method for class 'r2r_hashset'

```r
x[i] <- value
```

### Arguments

- **x**
  - an hashset or hashmap.

- **i**
  - for `[[`-subsetting, an arbitrary R object, the key to be queried or inserted/deleted from the hash tables. For `[]`-subsetting, a list or an atomic vector whose individual elements correspond to the keys.

- **value**
  - for `[[`-subsetting: TRUE or FALSE if x is an hashset, an arbitrary R object if x is an hashmap. In the case of hashsets, setting a key’s value to TRUE and FALSE is equivalent to inserting and deleting, respectively, such key from the set. For `[]`-subsetting, value must be a list or an atomic vector of the same length of i, whose individual elements are the values associated to the corresponding keys in the hash table.

### Value

the replacement forms ([<- and <-) always return value. `[[` returns TRUE or FALSE if x is an hashset, an arbitrary R object if x is an hashmap and i is a valid key; when i is not a key, the behaviour for hashmap depends on the value of `on_missing_key` (x). The `[]` operator returns a list of the same length of i, whose k-th element is given by `x[[ i[[k]] ]]` (the remark on missing keys for hashmap applies also here).

### Author(s)

Valerio Gherardi

---

**values**

*List all values from an hash map*

### Description

This function is used to list all values associated to keys in an hashmap. Implemented as a generic, but currently only the hashmap method is defined.
values

Usage

values(x)

## S3 method for class 'r2r_hashmap'
values(x)

Arguments

x an hashmap.

Value

a list. Values associated to keys in the hash map x.

Author(s)

Valerio Gherardi

Examples

m <- hashmap(list("a", 1), list("b", 2))
values(m)
## Index

<table>
<thead>
<tr>
<th>Function/Method</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>[.r2r_hashmap(subsetting_hashtables), 13</td>
<td></td>
</tr>
<tr>
<td>[.r2r_hashset(subsetting_hashtables), 13</td>
<td></td>
</tr>
<tr>
<td>[&lt;-.r2r_hashmap(subsetting_hashtables), 13</td>
<td></td>
</tr>
<tr>
<td>[&lt;-.r2r_hashset(subsetting_hashtables), 13</td>
<td></td>
</tr>
<tr>
<td>[[:.r2r_hashmap(subsetting_hashtables), 13</td>
<td></td>
</tr>
<tr>
<td>[[:.r2r_hashset(subsetting_hashtables), 13</td>
<td></td>
</tr>
<tr>
<td>[&lt;&lt;-.r2r_hashmap(subsetting_hashtables), 13</td>
<td></td>
</tr>
<tr>
<td>[&lt;&lt;-.r2r_hashset(subsetting_hashtables), 13</td>
<td></td>
</tr>
<tr>
<td>%has_key%(has_key), 8</td>
<td></td>
</tr>
<tr>
<td>%has_key%, 7</td>
<td></td>
</tr>
<tr>
<td>abort, 6</td>
<td></td>
</tr>
<tr>
<td>compare_fn, 2, 7</td>
<td></td>
</tr>
<tr>
<td>default, 3, 7</td>
<td></td>
</tr>
<tr>
<td>default&lt;-(default), 3</td>
<td></td>
</tr>
<tr>
<td>default_hash_fn, 4, 6</td>
<td></td>
</tr>
<tr>
<td>delete, 4, 7, 13</td>
<td></td>
</tr>
<tr>
<td>digest, 4, 6</td>
<td></td>
</tr>
<tr>
<td>environment, 6</td>
<td></td>
</tr>
<tr>
<td>has_key, 7, 8</td>
<td></td>
</tr>
<tr>
<td>hash_fn, 7, 8</td>
<td></td>
</tr>
<tr>
<td>hashmap(hashtable), 5</td>
<td></td>
</tr>
<tr>
<td>hashtset(hashtable), 5</td>
<td></td>
</tr>
<tr>
<td>hashtable, 3, 5, 12, 13</td>
<td></td>
</tr>
<tr>
<td>hashtable_methods, 5, 6, 7</td>
<td></td>
</tr>
<tr>
<td>identical, 6</td>
<td></td>
</tr>
<tr>
<td>insert, 7, 9, 13</td>
<td></td>
</tr>
<tr>
<td>keys, 7, 10</td>
<td></td>
</tr>
<tr>
<td>length.r2r hashtable, 7, 11</td>
<td></td>
</tr>
<tr>
<td>on_missing_key, 7, 11, 13, 14</td>
<td></td>
</tr>
<tr>
<td>on_missing_key&lt;-(on_missing_key), 11</td>
<td></td>
</tr>
<tr>
<td>query, 7, 12, 13</td>
<td></td>
</tr>
<tr>
<td>subsetting_hashtables, 7, 9, 12, 13</td>
<td></td>
</tr>
<tr>
<td>values, 7, 14</td>
<td></td>
</tr>
</tbody>
</table>