Package 'interplex'

October 13, 2022

Title Coercion Methods for Simplicial Complex Data Structures

Version 0.1.0

Description Computational topology, which includes topological data analysis
(TDA), makes pervasive use of abstract mathematical objects called
simplicial complexes; see Edelsbrunner and Harer (2010)
<doi:10.1090/mbk/069>.
Several R packages and other software libraries used through an R interface
construct and use data structures that represent simplicial complexes,
including mathematical graphs viewed as 1-dimensional complexes.
This package provides coercers (converters) between these data structures.
Currently supported structures are complete lists of simplices as used by
'TDA'; the simplex trees of Boissonnat and Maria (2014)
<doi:10.1007/s00453-014-9887-3> as implemented in 'simplextree' and in
Python GUDHI (by way of 'reticulate'); and the graph classes of 'igraph' and
'network', by way of the 'intergraph' package.

Suggests TDA, simplextree (>= 0.9.1), igraph (>= 0.6-0), network (>=
1.4-2), intergraph, reticulate, roxygen2, rmarkdown, testthat

License GPL (>= 3)

Encoding UTF-8

URL https://github.com/corybrunson/interplex

BugReports https://github.com/corybrunson/interplex/issues

RoxygenNote 7.2.0

NeedsCompilation no

Author Jason Cory Brunson [aut, cre] (<https://orcid.org/0000-0003-3126-9494>),
Yara Skaf [ctb]

Maintainer Jason Cory Brunson <cornelioid@gmail.com>

Repository CRAN

Date/Publication 2022-06-21 09:40:05 UTC

1

R **topics documented:**

---

as_cmplx   *Coerce objects to lists of simplices*

Description

Coerce objects to lists of simplices, as used by the TDA package.

Usage

```
as_cmplx(x, ...)

## Default S3 method:
as_cmplx(x, ...)

## S3 method for class 'Rcpp_SimplexTree'
as_cmplx(x, ...)

## S3 method for class 'gudhi.simplex_tree.SimplexTree'
as_cmplx(x, ...)

## S3 method for class 'igraph'
as_cmplx(x, index = NULL, ...)

## S3 method for class 'network'
as_cmplx(x, index = NULL, ...)
```

Arguments

x	An R object to be coerced. See Details.
...	Additional arguments passed to methods.
index	Integer-valued vertex attribute to be used as 0-simplex indices. Ignored if NULL (the default).

Details

as_cmplx() is a generic function with specific methods for different simplicial complex S3 classes. It returns a list of integer vectors, each of which represents a simplex, and *all* simplices are included in the list. When a filtration is constructed using TDA::*Filtration(), the first named element of the returned list, cmplx, is a list whose *i*th element contains the vertices of the *i*th simplex.

Value

A list of integer vectors, each encoding one simplex.

Examples

```
# pick the simplicial complex from a TDA filtration object
t <- 2 * pi * c(0, 1, 3, 6) / 7
rf <- TDA::ripsFiltration(
  cbind(x = cos(t), y = sin(t)),
  maxdimension = 2L, maxscale = 1.7
)
print(rf$cmplx)
cp_rf <- as_cmplx(rf)
print(cp_rf)

# convert a simplextree object
st <- simplextree::simplex_tree()
st$insert(list(3:5, 5:6, 8))
cp_st <- as_cmplx(st)
print(cp_st)

## Not run:

# convert a Python GUDHI simplex tree
gd <- reticulate::import("gudhi")
gd_st <- gd$SimplexTree()
for (s in list(3:5, 5:6, 8)) gd_st$insert(as.list(s))
cp_gd <- as_cmplx(gd_st)
print(cp_gd)

## End(Not run)

# convert an igraph object
ig <- igraph::graph(c(1,2, 2,3, 1,3, 3,4))
set.seed(0L)
ig <- igraph::set_vertex_attr(ig, "id", value = sample(igraph::vcount(ig)))
print(ig)
cp_ig <- as_cmplx(ig, index = "id")
print(cp_ig)

# convert a network object
el <- data.frame(tails = c(1, 2, 1, 3), heads = c(2, 3, 3, 4))
nw <- network::network.edgelist(el, network::network.initialize(4))
print(nw)
cp_nw <- as_cmplx(nw)
```

```
print(cp_nw)
```

---

as_igraph   *Coerce objects to class 'igraph'*

Description

Coerce objects to 'igraph' objects, as implemented in the igraph package.

Usage

```
as_igraph(x, ...)

## Default S3 method:
as_igraph(x, index = NULL, ...)

## S3 method for class 'Rcpp_SimplexTree'
as_igraph(x, index = NULL, ...)

## S3 method for class 'gudhi.simplex_tree.SimplexTree'
as_igraph(x, index = NULL, ...)

## S3 method for class 'igraph'
as_igraph(x, ...)

## S3 method for class 'network'
as_igraph(x, ...)
```

Arguments

x	An R object to be coerced. See Details.
...	Additional arguments passed to methods.
index	Character string to be added as a vertex attribute containing 0-simplex indices. Ignored if NULL (the default).

Details

as_igraph() is a generic function with specific methods for different simplicial complex S3 classes. It returns an igraph object.

Value

An object of class 'igraph'.

Examples

```
# convert a TDA filtration object
t <- 2 * pi * c(0, 1, 3, 6) / 7
rf <- TDA::ripsFiltration(
  cbind(x = cos(t), y = sin(t)),
  maxdimension = 2L, maxscale = 1.7
)
print(rf$cmplx)
ig_rf <- as_igraph(rf)
print(ig_rf)
ig_rf2 <- as_igraph(rf$cmplx)
print(ig_rf2)

# convert a simplextree object
st <- simplextree::simplex_tree()
st$insert(list(3:5, 5:6, 8))
ig_st <- as_igraph(st)
print(ig_st)

## Not run:

# convert a Python GUDHI simplex tree
gd <- reticulate::import("gudhi")
gd_st <- gd$SimplexTree()
for (s in list(3:5, 5:6, 8)) gd_st$insert(as.list(s))
ig_gd <- as_igraph(gd_st, index = "id")
print(ig_gd)

## End(Not run)

# convert a network object
el <- data.frame(tails = c(1, 2, 1, 3), heads = c(2, 3, 3, 4))
nw <- network::network.edgelist(el, network::network.initialize(4))
print(nw)
ig_nw <- as_igraph(nw)
print(ig_nw)
```

---

as_network   *Coerce objects to class 'network'*

Description

Coerce objects to 'network' objects, as implemented in the network package.

Usage

```
as_network(x, ...)

## Default S3 method:
```

```
as_network(x, index = NULL, ...)

## S3 method for class 'Rcpp_SimplexTree'
as_network(x, index = NULL, ...)

## S3 method for class 'gudhi.simplex_tree.SimplexTree'
as_network(x, index = NULL, ...)

## S3 method for class 'igraph'
as_network(x, ...)

## S3 method for class 'network'
as_network(x, ...)
```

Arguments

x	An R object to be coerced. See Details.
...	Additional arguments passed to methods.
index	Character string to be added as a vertex attribute containing 0-simplex indices. Ignored if NULL (the default).

Details

as_network() is a generic function with specific methods for different simplicial complex S3 classes. It returns a network object.

Value

An object of class 'network'.

Examples

```
# convert a TDA filtration object
t <- 2 * pi * c(0, 1, 3, 6) / 7
rf <- TDA::ripsFiltration(
  cbind(x = cos(t), y = sin(t)),
  maxdimension = 2L, maxscale = 1.7
)
print(rf$cmplx)
nw_rf <- as_network(rf)
print(nw_rf)
nw_rf2 <- as_network(rf$cmplx)
print(nw_rf2)

# convert a simplextree object
st <- simplextree::simplex_tree()
st$insert(list(3:5, 5:6, 8))
nw_st <- as_network(st)
print(nw_st)
```

```
## Not run:

# convert a Python GUDHI simplex tree
gd <- reticulate::import("gudhi")
gd_st <- gd$SimplexTree()
for (s in list(3:5, 5:6, 8)) gd_st$insert(as.list(s))
nw_gd <- as_network(gd_st, index = "id")
print(nw_gd)

## End(Not run)

# convert an igraph object
ig <- igraph::graph(c(1,2, 2,3, 1,3, 3,4))
print(ig)
nw_ig <- as_network(ig)
print(nw_ig)
```

as_py_gudhi_simplextree

Coerce objects to Python GUDHI simplex trees

Description

Coerce objects to 'SimplexTree' objects in Python GUDHI, accessed via the reticulate package.

Usage

```
as_py_gudhi_simplextree(x, ...)

## Default S3 method:
as_py_gudhi_simplextree(x, ...)

## S3 method for class 'Rcpp_SimplexTree'
as_py_gudhi_simplextree(x, ...)

## S3 method for class 'igraph'
as_py_gudhi_simplextree(x, index = NULL, ...)

## S3 method for class 'network'
as_py_gudhi_simplextree(x, index = NULL, ...)
```

Arguments

x	An R object to be coerced. See Details.
...	Additional arguments passed to methods.
index	Integer-valued vertex attribute to be used as 0-simplex indices. Ignored if NULL (the default).

Details

as_py_gudhi_simplextree() is a generic function with specific methods for different simplicial complex S3 classes. It returns an object of class 'gudhi.simplex_tree.SimplexTree', which is a [reticulate](#) accessor to a Python object of class 'SimplexTree' implemented in GUDHI.

Value

A simplex tree instantiated in Python GUDHI accessed through reticulate.

Author(s)

Jason Cory Brunson

Yara Skaf

Examples

```
## Not run:

# print GUDHI simplices
print_py_gudhi <- function(x) {
  reticulate::iterate(
    x$get_skeleton(x$dimension()),
    function(s) print(s[[1]]),
    simplify = FALSE
  )
}

# convert a TDA filtration object
t <- 2 * pi * c(0, 1, 3, 6) / 7
rf <- TDA::ripsFiltration(
  cbind(x = cos(t), y = sin(t)),
  maxdimension = 2L, maxscale = 1.7
)
print(rf$cmplx)
gd_rf <- as_py_gudhi_simplextree(rf)
print_py_gudhi(gd_rf)
gd_rf2 <- as_py_gudhi_simplextree(rf$cmplx)
print_py_gudhi(gd_rf2)

# convert a simplextree object
st <- simplextree::simplex_tree()
st$insert(list(3:5, 5:6, 8))
gd_st <- as_py_gudhi_simplextree(st)
print_py_gudhi(gd_st)

# convert an igraph object
ig <- igraph::graph(c(1,2, 2,3, 1,3, 3,4))
print(ig)
gd_ig <- as_py_gudhi_simplextree(ig)
print_py_gudhi(gd_ig)
```

```
# specify 0-simplex indices
set.seed(0L)
ig <- igraph::set_vertex_attr(ig, "id", value = sample(igraph::vcount(ig)) + 1L)
igraph::V(ig)$id
igraph::as_edgelist(ig)
gd_ig2 <- as_py_gudhi_simplextree(ig, index = "id")
print_py_gudhi(gd_ig2)

# convert a network object
el <- data.frame(tails = c(1, 2, 1, 3), heads = c(2, 3, 3, 4))
nw <- network::network.edgelist(el, network::network.initialize(4))
print(nw)
gd_nw <- as_py_gudhi_simplextree(nw)
print_py_gudhi(gd_nw)

## End(Not run)
```

---

as_rcpp_simplextree   *Coerce objects to class 'Rcpp_SimplexTree'*

Description

Coerce objects to 'Rcpp_SimplexTree' objects, as implemented in the simplextree package.

Usage

```
as_rcpp_simplextree(x, ...)

## Default S3 method:
as_rcpp_simplextree(x, ...)

## S3 method for class 'Rcpp_SimplexTree'
as_rcpp_simplextree(x, ...)

## S3 method for class 'gudhi.simplex_tree.SimplexTree'
as_rcpp_simplextree(x, ...)

## S3 method for class 'igraph'
as_rcpp_simplextree(x, index = NULL, ...)

## S3 method for class 'network'
as_rcpp_simplextree(x, index = NULL, ...)
```

Arguments

x	An R object to be coerced. See Details.
...	Additional arguments passed to methods.
index	Integer-valued vertex attribute to be used as 0-simplex indices. Ignored if NULL (the default).

Details

as_rcpp_simplextree() is a generic function with specific methods for different simplicial complex S3 classes. It returns an object of class 'Rcpp_SimplexTree', which is an Rcpp Module that exposes an instance of a C++ instance of a simplex tree.

Value

An instance of a simplex tree, exposed as an Rcpp Module with class 'Rcpp_SimplexTree'.

Examples

```
# convert a TDA filtration object
t <- 2 * pi * c(0, 1, 3, 6) / 7
rf <- TDA::ripsFiltration(
  cbind(x = cos(t), y = sin(t)),
  maxdimension = 2L, maxscale = 1.7
)
print(rf$cmplx)
st_rf <- as_rcpp_simplextree(rf)
print(st_rf)
st_rf2 <- as_rcpp_simplextree(rf$cmplx)
print(st_rf2)

## Not run:

# convert a Python GUDHI simplex tree
gd <- reticulate::import("gudhi")
gd_st <- gd$SimplexTree()
for (s in list(3:5, 5:6, 8)) gd_st$insert(as.list(s))
st_gd <- as_rcpp_simplextree(gd_st)
st_gd$as_list()

## End(Not run)

# convert an igraph object
ig <- igraph::graph(c(1,2, 2,3, 1,3, 3,4))
print(ig)
st_ig <- as_rcpp_simplextree(ig)
print(st_ig)

# specify 0-simplex indices
set.seed(0L)
ig <- igraph::set_vertex_attr(ig, "id", value = sample(igraph::vcount(ig)) + 1L)
igraph::V(ig)$id
igraph::as_edgelist(ig)
st_ig <- as_rcpp_simplextree(ig, index = "id")
st_ig$vertices
st_ig$edges

# convert a network object
el <- data.frame(tails = c(1, 2, 1, 3), heads = c(2, 3, 3, 4))
nw <- network::network.edgelist(el, network::network.initialize(4))
```

```
print(nw)
st_nw <- as_rcpp_simplextree(nw)
print(st_nw)
```

---

interplex	**interplex** *package*

Description

This is a helper package to coerce simplicial complexes between different data structures.

Details

This package helps interface between different topological data analytic packages and workflows by coercing simplicial complex objects stored using different data structures. Each coercion is designed to retain as much annotation as possible, whether of simplices or of the complex.

The package supports coercions between simplicial complexes stored using the following data structures:

- a complete list of simplices, as stored as cmplx values of filtration objects in the TDA package
- an object of class 'Rcpp_SimplexTree' as implemented in the simplextree package
- an object of class 'gudhi.simplex_tree.SimplexTree' as implemented in Python GUDHI and imported via reticulate
- an 'igraph' object, as implemented in the igraph package
- a 'network' object, as implemented in the network package

Index