Package 'fiber'

July 4, 2026

Type Package

Title S7 Data Structures for Diffusion MRI Tractography

Version 0.2.0

Description Provides three S7 classes — streamline, bundle, and bundle_set — for representing diffusion MRI tractography data in R, together with a concise set of methods for computing shape descriptors (arc-length, curvature, torsion, sinuosity), the Hausdorff distance between streamlines, arc-length reparametrization of streamlines and bundles onto uniform grids, combination of streamlines or bundles into a single bundle, combination of bundles from multiple subjects or sessions into a bundle_set, and coercion to and from the dwiFiber S4 class of the 'dti' package. See Dell'Acqua, F., Descoteaux, M. and Leemans, A. (2024) ``Handbook of Diffusion MR Tractography'' <doi:10.1016/C2018-0-02520-7> for more about the mathematical and computational underpinnings of diffusion MRI tractography.

License MIT + file LICENSE

Encoding UTF-8

URL https://github.com/tractoverse/fiber, https://tractoverse.github.io/fiber/

BugReports https://github.com/tractoverse/fiber/issues

Imports cli, methods, S7

Config/roxygen2/version 8.0.0

Config/roxygen2/markdown TRUE

Suggests dti, tinytest

Collate 'bind.R' 'bundle-set.R' 'bundle.R' 'streamline.R' 'coerce.R' 'cpp11.R' 'fiber-package.R' 'parametrize.R' 'shape.R' 'utils.R' 'zzz.R'

LinkingTo cpp11

NeedsCompilation yes

Author Aymeric Stamm [aut, cre] (ORCID: <https://orcid.org/0000-0002-8725-3654>)

Maintainer Aymeric Stamm <aymeric.stamm@cnrs.fr>

Repository CRAN

Date/Publication 2026-07-04 14:00:02 UTC

Contents

add_shape_descriptors *Adds shape descriptors to a streamline or bundle*

Description

`add_shape_descriptors()` is an S7 generic that computes a number of shape descriptors for each streamline object and stores them in the `@streamline_data` or `@point_data` slots as appropriate, with methods available for the following classes:

- `fiber::bundle`
- `fiber::streamline`

This function provides a convenient way to compute shape descriptors and attach them to streamline or bundle objects. See the documentation for each individual shape descriptor function (e.g. `get_euclidean_length()`, `get_curvilinear_length()`, `get_sinuosity()`, `get_curvature()`, `get_torsion()`) for more details on how each descriptor is computed.

For bundle objects, scalar descriptors (`euclidean_length`, `curvilinear_length`, `sinuosity`) are stored as length-S vectors in `bundle@streamline_data`. Per-point descriptors (`curvature`, `torsion`) continue to be stored in each individual streamline's `@point_data`. Both are accessible via `bundle[[i]]@streamline_data` and `bundle[[i]]@point_data` respectively, through the subsetting push-down mechanism.

Usage

```
add_shape_descriptors(
  x,
 descriptors = c("euclidean_length", "curvilinear_length", "sinuosity", "curvature",
    "torsion")
)
```

Arguments

`x`	A streamline or bundle object.
`descriptors`	A character vector of shape descriptors to add. Defaults to all available descriptors: `c("euclidean_length", "curvilinear_length", "sinuosity", "curvature", "torsion")`.

Value

An object of the same class as `x` with the specified shape descriptors added to the appropriate slots.

Examples

```
# add multiple shape descriptors to a single streamline
sl <- streamline(points = cbind(X = runif(10), Y = runif(10), Z = runif(10)))
sl <- add_shape_descriptors(
  sl,
```

```
  descriptors = c("euclidean_length", "curvilinear_length", "sinuosity")
)
sl@streamline_data$euclidean_length

# add multiple shape descriptors to a bundle
sl2 <- streamline(points = cbind(X = runif(20), Y = runif(20), Z = runif(20)))
b <- bundle(streamlines = list(sl, sl2))
b <- add_shape_descriptors(
  b,
  descriptors = c("euclidean_length", "curvilinear_length", "sinuosity")
)
b@streamline_data$euclidean_length  # length-2 vector
```

`add_shape_descriptors-fiber-bundle-method`
`add_shape_descriptors()` *method for* `bundle` *objects*

Description

Adds shape descriptors to a bundle. Scalar descriptors (`euclidean_length`, `curvilinear_length`, `sinuosity`) are stored as length-S vectors in `bundle@streamline_data`. Per-point descriptors (`curvature`, `torsion`) are stored in each individual streamline's `@point_data`. Both are accessible via the subsetting push-down (`bundle[[i]]`).

Arguments

`x`	A bundle object.
`descriptors`	A character vector of shape descriptors to add. Defaults to all available descriptors: `c("euclidean_length", "curvilinear_length", "sinuosity", "curvature", "torsion")`.

Value

A bundle with the specified shape descriptors added.

See Also

`add_shape_descriptors()`

add_shape_descriptors-fiber-streamline-method

add_shape_descriptors() *method for* streamline *objects*

Description

Adds multiple shape descriptors to a single streamline object.

Arguments

x	A streamline object.
descriptors	A character vector of shape descriptors to add. Defaults to all available descriptors: c("euclidean_length", "curvilinear_length", "sinuosity", "curvature", "torsion").

Value

A streamline with the specified shape descriptors added to the @streamline_data or @point_data slots as appropriate.

See Also

add_shape_descriptors()

as_bundle *Coerce an object to a bundle*

Description

as_bundle() converts a supported object into a bundle.

Usage

```
as_bundle(x, ...)
```

Arguments

x	An object to coerce.
...	Additional arguments (currently unused).

Details

Currently supported input classes:

- streamline: wrapped in a single-element bundle (lossless).
- bundle: returned unchanged.
- `dwiFiber` (from **dti**): each fiber becomes a streamline. The per-point direction vectors (columns 4–6 of `@fibers`) are stored as `@point_data$direction_x`, `@point_data$direction_y`, and `@point_data$direction_z`. Tracking metadata (`method`, `minfa`, `maxangle`) are stored as scalars in `@bundle_data`. Vector metadata (`ddim`, `ddim0`, `voxelext`, `orientation`) are stored as individual scalar entries (e.g. `voxelext_1`, `voxelext_2`, `voxelext_3`).

Value

A bundle object.

See Also

`as_streamline()`, `as_dwifiber()`

Examples

```
sl <- streamline(points = cbind(X = runif(5), Y = runif(5), Z = runif(5)))
b  <- as_bundle(sl)
b@n_streamlines  # 1
```

---

as_bundle-fiber-bundle-method

`as_bundle()` *method for* `bundle` *objects*

Description

`as_bundle()` method for `bundle` objects

Arguments

`x`	A bundle object.
`...`	Additional arguments (currently unused).

Value

`x` unchanged.

See Also

`as_bundle()`

as_bundle-fiber-streamline-method

as_bundle() *method for* streamline *objects*

Description

as_bundle() method for streamline objects

Arguments

x	A streamline object.
...	Additional arguments (currently unused).

Value

A bundle containing x as its sole streamline.

See Also

as_bundle()

as_bundle_set *Coerce an object to a bundle_set*

Description

as_bundle_set() converts a supported object into a bundle_set.

Usage

```
as_bundle_set(x, ...)
```

Arguments

x	An object to coerce.
...	Additional arguments passed to methods (e.g. name for bundles).

Details

Currently supported input classes:

- bundle_set: returned unchanged.
- bundle: wrapped in a single-element bundle_set. An optional name argument sets the element name (defaults to NULL, producing an unnamed single-element set).

Value

A bundle_set object.

See Also

bundle_set(), bind_bundle_sets()

Examples

```
sl <- streamline(points = cbind(X = runif(5), Y = runif(5), Z = runif(5)))
b <- bundle(streamlines = list(sl))
bs <- as_bundle_set(b, name = "sub-01")
bs@n_bundles  # 1
```

---

as_dwifiber *Coerce a streamline or bundle to a* dwiFiber *object*

Description

as_dwifiber() converts a streamline or bundle to the S4 class dwiFiber from the **dti** package.

Usage

```
as_dwifiber(x, ...)
```

Arguments

x	A streamline or bundle object.
...	Additional arguments (currently unused).

Details

Per-point direction vectors are taken from @point_data$direction_x, @point_data$direction_y, and @point_data$direction_z when present; otherwise they are estimated via finite differences of the coordinates (forward difference at the first point, backward difference at the last, central differences in between), then unit-normalised.

Bundle-level metadata stored in @bundle_data under the keys method, minfa, maxangle, level, and source are transferred to the corresponding dwiFiber slots when present. Vector-valued fields such as ddim, ddim0, voxelext, and orientation must be stored as individual scalars (e.g. ddim_1, ddim_2, ddim_3) and are reconstructed into vectors for the dwiFiber object.

Value

An S4 object of class dwiFiber (from **dti**).

See Also

as_streamline(), as_bundle()

Examples

```
if (requireNamespace("dti", quietly = TRUE)) {
  sl <- streamline(points = cbind(X = runif(5), Y = runif(5), Z = runif(5)))
  b  <- bundle(streamlines = list(sl))
  dfi <- as_dwifiber(b)
  class(dfi)  # "dwiFiber"
}
```

---

as_dwifiber-fiber-bundle-method

as_dwifiber() method for bundle *objects*

Description

as_dwifiber() method for bundle objects

Arguments

x	A bundle object.
...	Additional arguments (currently unused).

Value

An S4 dwiFiber object.

See Also

as_dwifiber()

---

as_dwifiber-fiber-streamline-method

as_dwifiber() method for streamline *objects*

Description

as_dwifiber() method for streamline objects

Arguments

x	A streamline object.
...	Additional arguments (currently unused).

Value

An S4 dwiFiber object.

See Also

`as_dwifiber()`

`as_streamline` *Coerce an object to a streamline*

Description

`as_streamline()` converts a supported object into a streamline.

Usage

```
as_streamline(x, ...)
```

Arguments

`x`	An object to coerce.
`...`	Additional arguments (currently unused).

Details

Currently supported input classes:

- streamline: returned unchanged.
- `dwiFiber` (from **dti**): the object must contain **exactly one** fiber. For multi-fiber objects use `as_bundle()` instead.

Value

A streamline object.

See Also

`as_bundle()`, `as_dwifiber()`

Examples

```
sl <- streamline(points = cbind(X = runif(5), Y = runif(5), Z = runif(5)))
identical(as_streamline(sl), sl)  # TRUE — identity coercion
```

as_streamline-fiber-bundle-method
as_streamline() *method for* bundle *objects*

Description

as_streamline() method for bundle objects

Arguments

x	A bundle object containing exactly one streamline.
...	Additional arguments (currently unused).

Value

The sole streamline inside x.

See Also

as_streamline()

as_streamline-fiber-streamline-method
as_streamline() *method for* streamline *objects*

Description

as_streamline() method for streamline objects

Arguments

x	A streamline object.
...	Additional arguments (currently unused).

Value

x unchanged.

See Also

as_streamline()

`bind_bundles` *Combine streamlines and/or bundles into a single bundle*

Description

Accepts any mix of streamline and bundle objects. All streamlines are collected into a flat list and wrapped in a new bundle. `bundle_data` from the first bundle argument (if any) is preserved; pass your own via the `bundle_data` argument to override. Similarly for `streamline_data`.

Usage

```
bind_bundles(..., streamline_data = NULL, bundle_data = NULL)
```

Arguments

`...`	One or more streamline or bundle objects.
`streamline_data`	A named list of per-streamline vectors to attach to the resulting bundle. Defaults to the `@streamline_data` of the first bundle input if one is present.
`bundle_data`	A named list of bundle-level scalar metadata to attach to the resulting bundle. Defaults to an empty list (or the `bundle_data` of the first bundle input if one is present and `bundle_data` is not supplied).

Value

A bundle containing all input streamlines.

Examples

```
sl1 <- streamline(points = cbind(X = runif(5), Y = runif(5), Z = runif(5)))
sl2 <- streamline(points = cbind(X = runif(5), Y = runif(5), Z = runif(5)))
b1 <- bundle(streamlines = list(sl1))
b2 <- bundle(streamlines = list(sl2))

# combine two bundles
b_all <- bind_bundles(b1, b2)
b_all@n_streamlines  # 2

# mix a bundle and a loose streamline
b_mixed <- bind_bundles(b1, sl2)
b_mixed@n_streamlines  # 2
```

`bind_bundle_sets` *Combine bundles and/or bundle_sets into a single bundle_set*

Description

Accepts any mix of bundle objects or bundle_set objects. All bundles are collected into a flat list and wrapped in a new bundle_set. Bare bundle arguments may optionally be named; unnamed bundles are included without a name label.

Usage

```
bind_bundle_sets(..., bundle_data = NULL, set_data = NULL)
```

Arguments

`...`	bundle objects or bundle_set objects to combine. Named bare bundle arguments will carry their name into the resulting set.
`bundle_data`	A named list of per-bundle vectors to attach to the resulting bundle_set. Defaults to the `@bundle_data` of the first bundle_set input if present.
`set_data`	A named list of set-level scalar metadata to attach to the resulting bundle_set. Defaults to the `set_data` of the first bundle_set input (if present) or an empty list.

Value

A bundle_set containing all input bundles.

Examples

```
sl <- streamline(points = cbind(X = runif(5), Y = runif(5), Z = runif(5)))
b1 <- bundle(streamlines = list(sl))
b2 <- bundle(streamlines = list(sl))

# two named bare bundles
bs <- bind_bundle_sets("sub-01" = b1, "sub-02" = b2)
bs@n_bundles   # 2
bs@bundle_data$if_from_input_list  # c("sub-01", "sub-02")

# combine two bundle_sets
bs1 <- bundle_set(list("sub-01" = b1))
bs2 <- bundle_set(list("sub-02" = b2))
bs_all <- bind_bundle_sets(bs1, bs2)
bs_all@n_bundles  # 2
```

bundle *Bundle S7 class*

Description

A bundle is an ordered collection of streamline objects representing a tractogram or white-matter bundle. It stores three compartments:

- @streamlines — a list of streamline objects.
- @streamline_data — a named list of per-streamline vectors (any type), each of length S (the number of streamlines). Values common to all streamlines may be lifted here automatically from the individual streamlines' @streamline_data slots at construction time.
- @bundle_data — a named list of scalars (length-1 values, any type) holding bundle-level metadata.

Usage

```
bundle(streamlines = list(), streamline_data = list(), bundle_data = list())
```

Arguments

streamlines	A list of streamline objects.
streamline_data	A named list of per-streamline vectors of length S. If not supplied, any @streamline_data keys common to **all** streamlines are lifted automatically.
bundle_data	A named list of bundle-level scalar metadata.

Value

A bundle S7 object.

Methods for standard generics

The following methods are defined for bundle objects:

- format(x, ...): Returns a cli-formatted string describing the bundle object.
- print(x, ...): Prints the formatted string to the console and invisibly returns x.
- length(x): Returns the number of streamlines (equivalent to x@n_streamlines).
- x[[i]]: Extracts the i-th streamline from the bundle, with bundle-level @streamline_data pushed back into the streamline.
- x[i]: Returns a new bundle containing only the selected streamlines, with @bundle_data and the subset of @streamline_data preserved.

Additional properties

`@n_streamlines` An integer scalar giving the number of streamlines in the bundle (read-only).

`@streamline_attributes` A character vector of the names of the per-streamline attributes stored at the bundle level (read-only).

`@bundle_attributes` A character vector of the names of the bundle-level attributes (read-only).

Examples

```
sl1 <- streamline(
  points = cbind(X = 0:4, Y = 0:4, Z = 0:4),
  streamline_data = list(mean_FA = 0.6, label = "CST")
)
sl2 <- streamline(
  points = cbind(X = 1:3, Y = 1:3, Z = 1:3),
  streamline_data = list(mean_FA = 0.7, label = "CST")
)
# mean_FA and label are common to both streamlines and are lifted
b <- bundle(streamlines = list(sl1, sl2))
b@n_streamlines  # 2
b@streamline_attributes  # c("mean_FA", "label")
b@streamline_data$mean_FA  # c(0.6, 0.7)

# Subsetting pushes streamline_data back down
b[[1]]@streamline_data$mean_FA  # 0.6
b[1]@streamline_data$mean_FA    # c(0.6)
```

---

bundle_set *Bundle set S7 class*

Description

A `bundle_set` is a collection of bundle objects, designed for multi-subject or multi-session studies where each element represents one subject's (or session's) tractogram. It stores three compartments:

- `@bundles` — a list of bundle objects (names are optional).
- `@bundle_data` — a named list of per-bundle vectors (any type), each of length B (the number of bundles). Values common to all bundles may be lifted here automatically from the individual bundles' `@bundle_data` slots at construction time.
- `@set_data` — a named list of scalars (length-1 values, any type) holding set-level metadata.

Usage

```
bundle_set(bundles = list(), bundle_data = list(), set_data = list())
```

Arguments

`bundles`	A list of bundle objects (may be named or unnamed).
`bundle_data`	A named list of per-bundle vectors of length B. If not supplied, any `@bundle_data` keys common to **all** bundles are lifted automatically.
`set_data`	A named list of set-level scalar metadata.

Value

A `bundle_set` S7 object.

Methods for standard generics

The following methods are defined for `bundle_set` objects:

- `format(x, ...)`: Returns a styled character string.
- `print(x, ...)`: Prints the formatted string to the console and invisibly returns `x`.
- `length(x)`: Returns the number of bundles.
- `x[[i]]`: Extracts the `i`-th (or named) bundle from the set, with set-level `@bundle_data` pushed back into the bundle.
- `x[i]`: Returns a new bundle_set containing only the selected bundles, with `@set_data` and the subset of `@bundle_data` preserved.

Additional properties

`@n_bundles` An integer scalar giving the number of bundles in the set (read-only).

`@bundle_attributes` A character vector of the names of the per-bundle attributes stored at the set level (read-only).

`@set_attributes` A character vector of the names of the set-level attributes (read-only).

Examples

```
sl <- streamline(points = cbind(X = 1:5, Y = 1:5, Z = 1:5))
b1 <- bundle(
  streamlines = list(sl),
  bundle_data = list(subject = "sub-01")
)
b2 <- bundle(
  streamlines = list(sl),
  bundle_data = list(subject = "sub-02")
)
# subject is common across bundles and lifted to bundle_set@bundle_data
bs <- bundle_set(bundles = list("sub-01" = b1, "sub-02" = b2))
bs@n_bundles          # 2
bs@bundle_attributes  # "subject"
bs@bundle_data$subject  # c("sub-01", "sub-02")
```

`compute_hausdorff_distance` *Computes the Hausdorff distance between streamlines*

Description

`compute_hausdorff_distance()` is an S7 generic that computes the symmetric Hausdorff distance between streamline objects based on their 3-D coordinate matrices, with methods available for the following classes:

- `ANY`
- `fiber::bundle`
- `fiber::streamline`

The four dispatch cases are:

- `streamline` + `streamline`: returns a single numeric scalar — the symmetric Hausdorff distance between the two streamlines.
- `bundle` + **missing**: returns a symmetric numeric distance matrix of dimension $n \times n$, where n is the number of streamlines in the bundle, giving all pairwise Hausdorff distances.
- `bundle` + `streamline`: returns a numeric vector of length n giving the Hausdorff distance from `y` to each streamline in `x`.
- `bundle` + `bundle`: returns a symmetric numeric distance matrix of dimension $(n_x + n_y) \times (n_x + n_y)$, treating the concatenation of all streamlines from `x` and `y` as one collection.

Usage

```
compute_hausdorff_distance(x, y = NULL)
```

Arguments

`x`	A streamline or bundle object.
`y`	A streamline or bundle object, or `NULL` (default). When `NULL` and `x` is a bundle, the pairwise distance matrix within `x` is returned.

Value

- A non-negative numeric scalar when both `x` and `y` are streamlines.
- A `dist` object of size n when `x` is a bundle and `y` is `NULL` or a bundle (use `as.matrix()` to expand to a full $n \times n$ matrix).
- A numeric vector of length n when `x` is a bundle and `y` is a streamline.

Examples

```
sl1 <- streamline(points = cbind(X = runif(10), Y = runif(10), Z = runif(10)))
sl2 <- streamline(points = cbind(X = runif(10), Y = runif(10), Z = runif(10)))

# streamline x streamline -> scalar
compute_hausdorff_distance(sl1, sl2)

# bundle x missing -> pairwise dist object
b <- bundle(streamlines = list(sl1, sl2))
compute_hausdorff_distance(b)
as.matrix(compute_hausdorff_distance(b))

# bundle x streamline -> vector
compute_hausdorff_distance(b, sl1)

# bundle x bundle -> combined pairwise matrix
b2 <- bundle(streamlines = list(sl2))
compute_hausdorff_distance(b, b2)
```

---

compute_hausdorff_distance,any-method

compute_hausdorff_distance() *catch-all method*

Description

compute_hausdorff_distance() catch-all method

Value

Does not return a value; always throws an error for unsupported input types.

---

compute_hausdorff_distance-fiber-bundle-method

compute_hausdorff_distance() *method for* bundle *objects*

Description

Dispatches to one of three behaviours depending on y:

Arguments

x	A bundle object.
y	NULL, a streamline, or a bundle.

Details

- y = NULL: pairwise distances within x as a dist object of size n, computed in C++ via a single linear loop.
- y is a streamline: numeric vector of distances from y to each streamline in x.
- y is a bundle: dist object for the concatenation of all streamlines from x and y.

Value

- A dist object of size x@n_streamlines when y is NULL or a bundle. The lower triangle stores all pairwise symmetric Hausdorff distances (computed in C++). Use as.matrix() to obtain the full $n \times n$ matrix.
- A numeric vector of length x@n_streamlines when y is a streamline.

See Also

compute_hausdorff_distance()

Examples

```
sl1 <- streamline(points = cbind(X = runif(10), Y = runif(10), Z = runif(10)))
sl2 <- streamline(points = cbind(X = runif(10), Y = runif(10), Z = runif(10)))
b <- bundle(streamlines = list(sl1, sl2))

# pairwise dist object (size 2)
compute_hausdorff_distance(b)
as.matrix(compute_hausdorff_distance(b))

# distances from sl1 to each streamline in b
compute_hausdorff_distance(b, sl1)
```

---

compute_hausdorff_distance-fiber-streamline-method

compute_hausdorff_distance() *method for two* streamline *objects*

Description

compute_hausdorff_distance() method for two streamline objects

Arguments

x	A streamline object.
y	A streamline object.

Value

A non-negative numeric scalar equal to $\max(d_H(x \to y), d_H(y \to x))$, where $d_H(A \to B) = \max_{a \in A} \min_{b \in B} \|a - b\|_2$ is the directed Hausdorff distance. The core computation is performed in C++ via `hausdorff_distance_cpp()`.

See Also

`compute_hausdorff_distance()`

Examples

```
sl1 <- streamline(points = cbind(X = runif(10), Y = runif(10), Z = runif(10)))
sl2 <- streamline(points = cbind(X = runif(10), Y = runif(10), Z = runif(10)))
compute_hausdorff_distance(sl1, sl2)
```

---

get_curvature *Curvature of a streamline*

Description

`get_curvature()` is function that computes the curvature of a streamline object. The curvature $\kappa(s)$ at each point along the arc-length abscissa is computed using cubic smoothing splines (3 degrees of freedom per component).

Usage

```
get_curvature(x)
```

Arguments

x	A streamline object.

Value

A non-negative numeric vector of length `x@n_points` giving the curvature $\kappa(s)$ at each sampled point along the streamline. Higher values indicate sharper bending at that location.

Examples

```
sl <- streamline(points = cbind(X = runif(10), Y = runif(10), Z = runif(10)))
get_curvature(sl)
```

get_curvilinear_length *Curvilinear length of a streamline*

Description

`get_curvilinear_length()` is a function that computes the total arc-length of a streamline object as the sum of Euclidean segment lengths between consecutive points.

Usage

```
get_curvilinear_length(x)
```

Arguments

x	A streamline object.

Value

A non-negative numeric scalar.

Examples

```
sl <- streamline(points = cbind(X = runif(10), Y = runif(10), Z = runif(10)))
get_curvilinear_length(sl)
```

get_euclidean_length *Euclidean length of a streamline*

Description

`get_euclidean_length()` is a function that computes the Euclidean (straight-line) distance of a streamline object.

Usage

```
get_euclidean_length(x)
```

Arguments

x	A streamline object.

Value

A non-negative numeric scalar.

Examples

```
sl <- streamline(points = cbind(X = runif(10), Y = runif(10), Z = runif(10)))
get_euclidean_length(sl)
```

---

get_sinuosity *Sinuosity of a streamline*

Description

`get_sinuosity()` is a function that computes the ratio of curvilinear length to Euclidean length for a streamline object, with a value of 1 indicating a perfectly straight streamline and larger values indicating greater curviness.

Usage

```
get_sinuosity(x)
```

Arguments

`x`	A streamline object.

Value

A numeric scalar ≥ 1.

Examples

```
sl <- streamline(points = cbind(X = runif(10), Y = runif(10), Z = runif(10)))
get_sinuosity(sl)
```

---

get_torsion *Torsion of a streamline*

Description

`get_torsion()` is function that computes the torsion of a streamline object. The torsion $\tau(s)$ at each point along the arc-length abscissa is computed using cubic smoothing splines (4 degrees of freedom per component).

Usage

```
get_torsion(x)
```

Arguments

`x`	A streamline object.

Value

A numeric vector of length `x@n_points` giving the torsion $\tau(s)$ at each sampled point along the streamline. Positive values indicate right-handed twisting; negative values indicate left-handed twisting; zero indicates a planar curve at that location.

Examples

```
sl <- streamline(points = cbind(X = runif(10), Y = runif(10), Z = runif(10)))
get_torsion(sl)
```

`is_bundle` *Test whether an object is a bundle*

Description

Test whether an object is a bundle

Usage

```
is_bundle(x)
```

Arguments

`x`	An object.

Value

`TRUE` if `x` is of class bundle, otherwise `FALSE`.

Examples

```
sl <- streamline(points = cbind(X = 1:5, Y = 1:5, Z = 1:5))
b <- bundle(streamlines = list(sl))
is_bundle(b)   # TRUE
is_bundle(sl)  # FALSE
```

is_bundle_set *Test whether an object is a bundle_set*

Description

Test whether an object is a bundle_set

Usage

```
is_bundle_set(x)
```

Arguments

x	An object.

Value

TRUE if x is of class bundle_set, otherwise FALSE.

Examples

```
sl <- streamline(points = cbind(X = 1:5, Y = 1:5, Z = 1:5))
b <- bundle(streamlines = list(sl))
bs <- bundle_set(bundles = list(b))
is_bundle_set(bs)  # TRUE
is_bundle_set(b)   # FALSE
```

is_streamline *Test whether an object is a streamline*

Description

Test whether an object is a streamline

Usage

```
is_streamline(x)
```

Arguments

x	An object.

Value

TRUE if x is of class streamline, otherwise FALSE.

Examples

```
sl <- streamline(points = cbind(X = 1:5, Y = 1:5, Z = 1:5))
is_streamline(sl) # TRUE
is_streamline(42) # FALSE
```

`reparametrize` *Reparametrize a streamline or bundle onto a uniform arc-length grid*

Description

`reparametrize()` is an S7 generic that resamples the 3-D coordinates (and any numeric `@point_data` attributes) of a tractography object onto a uniform arc-length grid using linear interpolation, with methods available for the following classes:

- `fiber::bundle`
- `fiber::streamline`

Usage

```
reparametrize(x, n_points = NULL)
```

Arguments

`x`	A streamline or bundle object.
`n_points`	Number of equally-spaced arc-length points to use. • For a single streamline, defaults to `x@n_points`. • For a bundle, defaults to the rounded mean number of points across all streamlines. Pass `NULL` to use these defaults explicitly.

Value

An object of the same class as `x` reparametrized onto the new grid.

Examples

```
# reparametrize a single streamline to 10 points
sl <- streamline(points = cbind(X = runif(5), Y = runif(5), Z = runif(5)))
sl_reparam <- reparametrize(sl, n_points = 10)
# reparametrize a bundle to the mean number of points across its streamlines
sl1 <- streamline(points = cbind(X = runif(5), Y = runif(5), Z = runif(5)))
sl2 <- streamline(points = cbind(X = runif(10), Y = runif(10), Z = runif(10)))
b <- bundle(streamlines = list(sl1, sl2))
bundle_reparam <- reparametrize(b)
```

`reparametrize-fiber-bundle-method`
`reparametrize()` *method for* `bundle` *objects*

Description

Resamples every streamline inside a bundle onto a common uniform arc-length grid. See `reparametrize()` for the full parameter documentation.

Arguments

`x`	A bundle object.
`n_points`	Number of equally-spaced arc-length points to use. • For a single streamline, defaults to `x@n_points`. • For a bundle, defaults to the rounded mean number of points across all streamlines. Pass `NULL` to use these defaults explicitly.

Value

A bundle reparametrized onto the new grid. Every streamline in the returned bundle has exactly `n_points` rows in `@points` (defaulting to the rounded mean number of points across all streamlines when `n_points` is `NULL`). `@streamline_data` and `@bundle_data` are preserved unchanged.

See Also

`reparametrize()`

Examples

```
sl1 <- streamline(points = cbind(X = runif(5), Y = runif(5), Z = runif(5)))
sl2 <- streamline(points = cbind(X = runif(10), Y = runif(10), Z = runif(10)))
b <- bundle(streamlines = list(sl1, sl2))
b_reparam <- reparametrize(b, n_points = 8)
b_reparam[[1]]@n_points  # 8
b_reparam[[2]]@n_points  # 8
```

`reparametrize-fiber-streamline-method` *reparametrize() method for* `streamline` *objects*

Description

Resamples a single streamline onto a uniform arc-length grid. See `reparametrize()` for the full parameter documentation.

Arguments

`x`	A streamline object.
`n_points`	Number of equally-spaced arc-length points to use. • For a single streamline, defaults to `x@n_points`. • For a bundle, defaults to the rounded mean number of points across all streamlines. Pass `NULL` to use these defaults explicitly.

Value

A streamline reparametrized onto the new grid. The returned object has `@points` resampled to exactly `n_points` rows via linear interpolation, and any numeric `@point_data` entries likewise resampled. Non-numeric `@point_data` entries are dropped with a warning. `@streamline_data` is preserved unchanged.

See Also

`reparametrize()`

Examples

```
sl <- streamline(
  points = cbind(X = runif(10), Y = runif(10), Z = runif(10)),
  point_data = list(FA = runif(10))
)
sl_reparam <- reparametrize(sl, n_points = 20)
sl_reparam@n_points  # 20
```

`streamline` *Streamline S7 class*

Description

A `streamline` represents a single fibre tract. It stores three data compartments:

- `@points` — a $P \times 3$ numeric matrix with column names `"X"`, `"Y"`, and `"Z"` holding the 3-D coordinates of the P points along the tract.
- `@point_data` — a named list of numeric vectors, each of length P, holding additional per-point scalar attributes (e.g. fractional anisotropy). Coordinates are **not** stored here.
- `@streamline_data` — a named list of scalars (length-1 values of any type) holding per-streamline attributes (e.g. a tract-level weight or mean FA, or a character label).

Usage

```
streamline(points = NULL, point_data = list(), streamline_data = list())
```

Arguments

`points`	A $P \times 3$ numeric matrix with column names `"X"`, `"Y"`, and `"Z"` giving the 3-D coordinates of the streamline points.
`point_data`	A named list of numeric vectors, each of length P, holding additional per-point attributes. Must **not** include `"X"`, `"Y"`, or `"Z"` (those live in `@points`).
`streamline_data`	A named list of per-streamline scalars (length-1, any type).

Value

A `streamline` S7 object.

Methods for standard generics

The following methods are defined for `streamline` objects:

- `format(x, ...)`: Returns a cli-formatted string describing the streamline object.
- `print(x, ...)`: Prints the formatted string to the console and invisibly returns `x`.

Additional properties

`@n_points` An integer scalar giving the number of points in the streamline (read-only).

`@point_attributes` A character vector of the names of the per-point attributes stored in `@point_data` (read-only).

`@streamline_attributes` A character vector of the names of the per-streamline attributes (read-only).

Examples

```
# Create a streamline with 5 points and some attributes
sl <- streamline(
  points = cbind(
    X = c(0, 1, 1, 1, 0),
    Y = c(0, 0, 1, 1, 1),
    Z = c(0, 0, 0, 1, 1)
  ),
  point_data = list(FA = c(0.5, 0.6, 0.7, 0.8, 0.9)),
  streamline_data = list(mean_FA = 0.7, label = "CST")
)
sl@n_points         # 5
sl@point_attributes # "FA"
sl@streamline_attributes # c("mean_FA", "label")

# format() and print() methods
format(sl)
print(sl)
```

Index