

Package: fiber (via r-universe)

May 31, 2026

Type Package

Title S7 Data Structures for Diffusion MRI Tractography

Version 0.1.2

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

LinkingTo cpp11

Repository <https://tractoverse.r-universe.dev>

Date/Publication 2026-05-27 17:35:58 UTC

RemoteUrl <https://github.com/tractoverse/fiber>

RemoteRef HEAD

RemoteSha afd0a17df1b2a6fca7b32b7de98c55ed1045fb65

Contents

add_shape_descriptors	3
add_shape_descriptors-fiber-bundle-method	4
add_shape_descriptors-fiber-streamline-method	4
as_bundle	5
as_bundle-fiber-bundle-method	6
as_bundle-fiber-streamline-method	6
as_bundle_set	7
as_dwifiber	8
as_dwifiber-fiber-bundle-method	9
as_dwifiber-fiber-streamline-method	9
as_streamline	10
as_streamline-fiber-bundle-method	11
as_streamline-fiber-streamline-method	11
bind_bundle_sets	12
bind_bundles	13
bundle	14
bundle_set	15
compute_hausdorff_distance	16
compute_hausdorff_distance-fiber-bundle-method	18
compute_hausdorff_distance-fiber-streamline-method	19
compute_hausdorff_distance,any-method	19
get_curvature	20
get_curvilinear_length	20
get_euclidean_length	21
get_sinusosity	22
get_torsion	22
is_bundle	23
is_bundle_set	24
is_streamline	24
reparametrize	25
reparametrize-fiber-bundle-method	26
reparametrize-fiber-streamline-method	27
streamline	28

Index

30

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_sinusosity()`, `get_curvature()`, `get_torsion()`) for more details on how each descriptor is computed.

Usage

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

Value

An object of the same class as `x` with the specified shape descriptors added to the @streamline_data or @point_data slots of each streamline.

Examples

```
# add multiple shape descriptors to a single streamline  
pts <- matrix(runif(30), ncol = 3)  
colnames(pts) <- c("X", "Y", "Z")  
sl <- streamline(points = pts)  
sl <- add_shape_descriptors(  
  sl,  
  descriptors = c("euclidean_length", "curvilinear_length", "sinusosity")  
)  
# add multiple shape descriptors to a bundle
```

```

s11 <- streamline(points = pts)
pts2 <- matrix(runif(60), ncol = 3)
colnames(pts2) <- c("X", "Y", "Z")
s12 <- streamline(points = pts2)
b <- bundle(streamlines = list(s11, s12))
b <- add_shape_descriptors(
  b,
  descriptors = c("euclidean_length", "curvilinear_length", "sinuosity")
)

```

add_shape_descriptors-fiber-bundle-method

[add_shape_descriptors\(\)](#) *method for bundle objects*

Description

Adds multiple shape descriptors to every [streamline](#) inside a [bundle](#).

Arguments

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

Value

A [bundle](#) with the specified shape descriptors added to the `@streamline_data` or `@point_data` slots of each streamline as appropriate.

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: <code>c("euclidean_length", "curvilinear_length", "sinuosity", "curvature", "torsion")</code> .

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	<i>Coerce an object to a bundle</i>
-----------	-------------------------------------

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 in @bundle_data.

Value

A [bundle](#) object.

See Also

[as_streamline\(\)](#), [as_dwifiber\(\)](#)

Examples

```
pts <- matrix(runif(15), ncol = 3, dimnames = list(NULL, c("X", "Y", "Z")))
sl <- streamline(points = pts)
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	<i>Coerce an object to a bundle_set</i>
---------------	-----------------------------------------

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 "bundle_1").

Value

A [bundle_set](#) object.

See Also

[bundle_set\(\)](#), [bind_bundle_sets\(\)](#)

Examples

```
pts <- matrix(runif(15), ncol = 3, dimnames = list(NULL, c("X", "Y", "Z")))
b <- bundle(streamlines = list(streamline(points = pts)))
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`, `ddim`, `ddim0`, `voxelext`, `orientation`, `rotation`, `level`, and `source` are transferred to the corresponding `dwiFiber` / `dwi` slots when present. MRI-acquisition metadata that cannot be recovered from a fiber object (gradient directions, b-values, etc.) are filled with neutral placeholders.

Value

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

See Also

[as_streamline\(\)](#), [as_bundle\(\)](#)

Examples

```
if (requireNamespace("dti", quietly = TRUE)) {  
  pts <- matrix(runif(15), ncol = 3, dimnames = list(NULL, c("X", "Y", "Z")))  
  sl <- streamline(points = pts)  
  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	<i>Coerce an object to a streamline</i>
---------------	-----------------------------------------

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

```
pts <- matrix(runif(15), ncol = 3, dimnames = list(NULL, c("X", "Y", "Z")))
sl <- streamline(points = pts)
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_bundle_sets	<i>Combine bundles and/or bundle_sets into a single bundle_set</i>
------------------	--------------------------------------------------------------------

Description

Accepts any mix of named `bundle` objects (passed as `name = bundle`) or `bundle_set` objects. All bundles are collected into a flat named list and wrapped in a new `bundle_set`.

Usage

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

Arguments

<code>...</code>	Named <code>bundle</code> objects or <code>bundle_set</code> objects to combine. Each bare <code>bundle</code> argument must be named so that its label in the resulting set is unambiguous.
<code>set_data</code>	A named list of set-level metadata to attach to the resulting <code>bundle_set</code> . Defaults to the <code>set_data</code> of the first <code>bundle_set</code> input (if present) or an empty list.

Value

A `bundle_set` containing all input bundles.

Examples

```
pts <- matrix(runif(15), ncol = 3, dimnames = list(NULL, c("X", "Y", "Z")))
b1 <- bundle(streamlines = list(streamline(points = pts)))
b2 <- bundle(streamlines = list(streamline(points = pts)))

# two named bare bundles
bs <- bind_bundle_sets("sub-01" = b1, "sub-02" = b2)
bs@n_bundles # 2
bs@bundle_names # 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
```

bind_bundles	<i>Combine streamlines and/or bundles into a single bundle</i>
--------------	----------------------------------------------------------------

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.

Usage

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

Arguments

`...` One or more [streamline](#) or [bundle](#) objects.

`bundle_data` A named list of bundle-level 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

```
pts <- matrix(runif(15), ncol = 3, dimnames = list(NULL, c("X", "Y", "Z")))
s11 <- streamline(points = pts)
s12 <- streamline(points = pts)
b1 <- bundle(streamlines = list(s11))
b2 <- bundle(streamlines = list(s12))

# 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, s12)
b_mixed@n_streamlines # 2
```

 bundle

Bundle S7 class

Description

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

- `@streamlines` — a list of `streamline` objects.
- `@bundle_data` — a named list of bundle-level metadata (arbitrary R objects, e.g. the affine transform used during tracking).

Usage

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

Arguments

`streamlines` A list of `streamline` objects.
`bundle_data` A named list of bundle-level metadata.

Value

A `bundle S7` object.

Methods for standard generics

The following methods are defined for `bundle` objects:

- `format(x, ...)`: Returns a compact character string such as `<bundle [2 streamlines | 10-20 pts/streamline]>`.
- `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.
- `x[i]`: Returns a new `bundle` containing only the selected streamlines, preserving `@bundle_data`.

Additional properties

`@n_streamlines` An integer scalar giving the number of streamlines in the bundle (read-only).
`@bundle_attributes` A character vector of the names of the bundle-level attributes (read-only).

Examples

```

pts <- matrix(runif(15), ncol = 3, dimnames = list(NULL, c("X", "Y", "Z")))
s1 <- streamline(points = pts)
b <- bundle(streamlines = list(s1))
b@n_streamlines # 1
b@bundle_attributes # NULL (no bundle-level attributes)

# bundle_data is stored
b2 <- bundle(
  streamlines = list(s1),
  bundle_data = list(subject = "sub-01")
)
b2@bundle_data$subject # "sub-01"

# format(), print(), length() and indexing methods
format(b2)
print(b2)
length(b2) # 1
b2[[1]] # first streamline

# subsetting preserves bundle_data
pts2 <- matrix(runif(15), ncol = 3, dimnames = list(NULL, c("X", "Y", "Z")))
s12 <- streamline(points = pts2)
b3 <- bundle(streamlines = list(s1, s12), bundle_data = list(subject = "sub-01"))
b3[1]@n_streamlines # 1, bundle_data preserved

```

bundle_set

Bundle set S7 class

Description

A `bundle_set` is a **named 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 two compartments:

- `@bundles` — a *named* list of `bundle` objects. Names typically encode subject or session identifiers (e.g. "sub-01", "sub-02").
- `@set_data` — a named list of set-level metadata (arbitrary R objects, e.g. study name, atlas used, acquisition protocol).

Usage

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

Arguments

<code>bundles</code>	A named list of <code>bundle</code> objects.
<code>set_data</code>	A named list of set-level metadata.

Value

A `bundle_set S7` object.

Methods for standard generics

The following methods are defined for `bundle_set` objects:

- `format(x, ...)`: Returns a compact 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.
- `x[i]`: Returns a new `bundle_set` containing only the selected bundles, preserving `@set_data`.
- `names(x)`: Returns the names of the bundles.

Additional properties

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

`@bundle_names` A character vector of the names of the bundles (read-only).

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

Examples

```
pts <- matrix(runif(15), ncol = 3, dimnames = list(NULL, c("X", "Y", "Z")))
b1 <- bundle(streamlines = list(streamline(points = pts)),
             bundle_data = list(subject = "sub-01"))
b2 <- bundle(streamlines = list(streamline(points = pts)),
             bundle_data = list(subject = "sub-02"))
bs <- bundle_set(bundles = list("sub-01" = b1, "sub-02" = b2))
bs@n_bundles      # 2
bs@bundle_names   # c("sub-01", "sub-02")
bs[["sub-01"]]    # first bundle
```

`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

```
pts1 <- matrix(runif(30), ncol = 3)
colnames(pts1) <- c("X", "Y", "Z")
s11 <- streamline(points = pts1)
pts2 <- matrix(runif(30), ncol = 3)
colnames(pts2) <- c("X", "Y", "Z")
s12 <- streamline(points = pts2)

# streamline x streamline -> scalar
compute_hausdorff_distance(s11, s12)

# bundle x missing -> pairwise dist object
b <- bundle(streamlines = list(s11, s12))
compute_hausdorff_distance(b)
as.matrix(compute_hausdorff_distance(b))

# bundle x streamline -> vector
compute_hausdorff_distance(b, s11)

# bundle x bundle -> combined pairwise matrix
```

```
b2 <- bundle(streamlines = list(s12))
compute_hausdorff_distance(b, b2)
```

```
compute_hausdorff_distance-fiber-bundle-method
      compute_hausdorff_distance() method for bundle objects
```

Description

Dispatches to one of three behaviours depending on `y`:

Arguments

<code>x</code>	A bundle object.
<code>y</code>	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

```
pts1 <- matrix(runif(30), ncol = 3)
colnames(pts1) <- c("X", "Y", "Z")
pts2 <- matrix(runif(30), ncol = 3)
colnames(pts2) <- c("X", "Y", "Z")
s11 <- streamline(points = pts1)
s12 <- streamline(points = pts2)
b <- bundle(streamlines = list(s11, s12))

# 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 ob-
    jects
```

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

See Also

[compute_hausdorff_distance\(\)](#)

Examples

```
pts1 <- matrix(runif(30), ncol = 3)
colnames(pts1) <- c("X", "Y", "Z")
pts2 <- matrix(runif(30), ncol = 3)
colnames(pts2) <- c("X", "Y", "Z")
sl1 <- streamline(points = pts1)
sl2 <- streamline(points = pts2)
compute_hausdorff_distance(sl1, sl2)
```

```
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.

get_curvature	<i>Curvature of a streamline</i>
---------------	----------------------------------

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

```
pts <- matrix(runif(30), ncol = 3)
colnames(pts) <- c("X", "Y", "Z")
sl <- streamline(points = pts)
get_curvature(sl)
```

get_curvilinear_length	<i>Curvilinear length of a streamline</i>
------------------------	-------------------------------------------

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

```
pts <- matrix(runif(30), ncol = 3)
colnames(pts) <- c("X", "Y", "Z")
sl <- streamline(points = pts)
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

```
pts <- matrix(runif(30), ncol = 3)
colnames(pts) <- c("X", "Y", "Z")
sl <- streamline(points = pts)
get_euclidean_length(sl)
```

get_sinusosity	<i>Sinusosity of a streamline</i>
----------------	-----------------------------------

Description

get_sinusosity() 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_sinusosity(x)
```

Arguments

x A [streamline](#) object.

Value

A numeric scalar ≥ 1 .

Examples

```
pts <- matrix(runif(30), ncol = 3)
colnames(pts) <- c("X", "Y", "Z")
sl <- streamline(points = pts)
get_sinusosity(sl)
```

get_torsion	<i>Torsion of a streamline</i>
-------------	--------------------------------

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

```
pts <- matrix(runif(30), ncol = 3)
colnames(pts) <- c("X", "Y", "Z")
sl <- streamline(points = pts)
get_torsion(sl)
```

is_bundle	<i>Test whether an object is a bundle</i>
-----------	-------------------------------------------

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

```
pts <- matrix(runif(15), ncol = 3, dimnames = list(NULL, c("X", "Y", "Z")))
sl <- streamline(points = pts)
b <- bundle(streamlines = list(sl))
is_bundle(b) # TRUE
is_bundle(sl) # FALSE
```

is_bundle_set	<i>Test whether an object is a bundle_set</i>
---------------	-----------------------------------------------

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

```
pts <- matrix(runif(15), ncol = 3, dimnames = list(NULL, c("X", "Y", "Z")))
b <- bundle(streamlines = list(streamline(points = pts)))
bs <- bundle_set(bundles = list("sub-01" = b))
is_bundle_set(bs) # TRUE
is_bundle_set(b)  # FALSE
```

is_streamline	<i>Test whether an object is a streamline</i>
---------------	-----------------------------------------------

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

```
pts <- matrix(runif(15), ncol = 3, dimnames = list(NULL, c("X", "Y", "Z")))
s1 <- streamline(points = pts)
is_streamline(s1)      # 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 @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 `nrow(x@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
pts <- matrix(runif(30), ncol = 3)
colnames(pts) <- c("X", "Y", "Z")
s1 <- streamline(points = pts)
s1_reparam <- reparametrize(s1, n_points = 10)
# reparametrize a bundle to the mean number of points across its streamlines
s11 <- streamline(points = pts)
pts2 <- matrix(runif(60), ncol = 3)
colnames(pts2) <- c("X", "Y", "Z")
s12 <- streamline(points = pts2)
b <- bundle(streamlines = list(s11, s12))
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 `nrow(x@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`).

See Also

[reparametrize\(\)](#)

Examples

```
pts1 <- matrix(runif(30), ncol = 3)
colnames(pts1) <- c("X", "Y", "Z")
pts2 <- matrix(runif(60), ncol = 3)
colnames(pts2) <- c("X", "Y", "Z")
b <- bundle(streamlines = list(streamline(points = pts1),
                              streamline(points = pts2)))
b_reparam <- reparametrize(b, n_points = 15)
b_reparam[[1]]@n_points # 15
b_reparam[[2]]@n_points # 15
```

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 `nrow(x@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 the same class as the input but with `@points` resampled to exactly `n_points` rows and all `@point_data` vectors resampled correspondingly via linear interpolation.

See Also

[reparametrize\(\)](#)

Examples

```
pts <- matrix(runif(30), ncol = 3)
colnames(pts) <- c("X", "Y", "Z")
sl <- streamline(points = pts, 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 that mirror the conceptual levels found in tractography file formats:

- @points — an $n \times 3$ numeric matrix whose columns are named "X", "Y", and "Z", holding the ordered 3-D coordinates of the n points along the tract.
- @point_data — a named list of numeric vectors, each of length n , holding per-point scalar attributes (e.g. fractional anisotropy sampled at every point).
- @streamline_data — a named list of numeric scalars (length-1 vectors) holding per-streamline attributes (e.g. a tract-level weight or mean FA).

Usage

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

Arguments

points A numeric matrix with columns "X", "Y", and "Z".
point_data A named list of per-point numeric vectors.
streamline_data A named list of per-streamline numeric scalars.

Value

A streamline S7 object.

Methods for standard generics

The following methods are defined for streamline objects:

- `format(x, ...)`: Returns a compact character string such as `<streamline [10 pts] | point: FA>`.
- `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 (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 = matrix(
    c(0, 0, 0,
      1, 0, 0,
      1, 1, 0,
      1, 1, 1,
      0, 1, 1),
    ncol = 3,
    byrow = TRUE,
    dimnames = list(NULL, c("X", "Y", "Z"))
  ),
  point_data = list(FA = c(0.5, 0.6, 0.7, 0.8, 0.9)),
  streamline_data = list(mean_FA = 0.7)
)
sl@n_points # 5
sl@point_attributes # "FA"
sl@streamline_attributes # "mean_FA"

# format() and print() methods
format(sl) # "<streamline [5 pts] | point: FA | streamline: mean_FA>"
print(sl)
```

Index

add_shape_descriptors, 3
add_shape_descriptors(), 4, 5
add_shape_descriptors, fiber::bundle-method
 (add_shape_descriptors-fiber-bundle-method), 4
add_shape_descriptors, fiber::streamline-method
 (add_shape_descriptors-fiber-streamline-method), 4
add_shape_descriptors-fiber-bundle-method, 4
add_shape_descriptors-fiber-streamline-method, 4
as.matrix(), 17, 18
as_bundle, 5
as_bundle(), 6, 8, 10
as_bundle, fiber::bundle-method
 (as_bundle-fiber-bundle-method), 6
as_bundle, fiber::streamline-method
 (as_bundle-fiber-streamline-method), 6
as_bundle-fiber-bundle-method, 6
as_bundle-fiber-streamline-method, 6
as_bundle_set, 7
as_dwifiber, 8
as_dwifiber(), 5, 9, 10
as_dwifiber, fiber::bundle-method
 (as_dwifiber-fiber-bundle-method), 9
as_dwifiber, fiber::streamline-method
 (as_dwifiber-fiber-streamline-method), 9
as_dwifiber-fiber-bundle-method, 9
as_dwifiber-fiber-streamline-method, 9
as_streamline, 10
as_streamline(), 5, 8, 11
as_streamline, fiber::bundle-method
 (as_streamline-fiber-bundle-method), 11
as_streamline, fiber::streamline-method
 (as_streamline-fiber-streamline-method), 11
as_streamline-fiber-bundle-method, 11
as_streamline-fiber-streamline-method, 11
bind_bundle_sets, 12
bind_bundle_sets(), 7
bind_bundles, 13
bundle, 3-9, 11-13, 14, 14-18, 23, 25-27
bundle_set, 7, 12, 15, 16, 24
bundle_set(), 7
compute_hausdorff_distance, 16
compute_hausdorff_distance(), 18, 19
compute_hausdorff_distance, any-method, 19
compute_hausdorff_distance, fiber::bundle-method
 (compute_hausdorff_distance-fiber-bundle-method), 18
compute_hausdorff_distance, fiber::streamline-method
 (compute_hausdorff_distance-fiber-streamline-method), 19
compute_hausdorff_distance-fiber-bundle-method, 18
compute_hausdorff_distance-fiber-streamline-method, 19
dist, 17, 18
fiber::bundle, 3, 16, 25
fiber::streamline, 3, 16, 25
get_curvature, 20
get_curvilinear_length, 20
get_euclidean_length, 21
get_sinusosity, 22
get_torsion, 22
is_bundle, 23

is_bundle_set, [24](#)
is_streamline, [24](#)

reparametrize, [25](#)
reparametrize(), [26](#), [27](#)
reparametrize, fiber::bundle-method
 (reparametrize-fiber-bundle-method),
 [26](#)
reparametrize, fiber::streamline-method
 (reparametrize-fiber-streamline-method),
 [27](#)
reparametrize-fiber-bundle-method, [26](#)
reparametrize-fiber-streamline-method,
 [27](#)

streamline, [3-6](#), [8-11](#), [13](#), [14](#), [16-22](#), [24-27](#),
 [28](#)