Package: aniframe (via r-universe)

June 3, 2026

Type Package

Title An R package providing core data structures for movement data

Version 0.5.0

Description An R package providing core data structures for movement data.

License MIT + file LICENSE

URL http://animovement.dev/aniframe/, https://github.com/animovement/aniframe/

BugReports https://github.com/animovement/aniframe/issues

Encoding UTF-8

LazyData true

Depends R (>= 4.1.0)

Imports cli, dplyr, rlang, pillar, anytime, lifecycle

Suggests knitr, rmarkdown, testthat (>= 3.0.0), covr, pkgdown

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

VignetteBuilder knitr

Config/Needs/website rmarkdown

Repository https://animovement.r-universe.dev

Date/Publication 2026-05-04 16:23:25 UTC

RemoteUrl https://github.com/animovement/aniframe

RemoteRef HEAD

RemoteSha f10ffed53057a00be0ddd69013f5b7b641c991be

Contents

`[.aniframe` *Subset aniframe with [*

Description

Subset aniframe with [

Usage

```
## S3 method for class 'aniframe'
x[i, j, ..., drop = FALSE]
```

Arguments

`x`	An aniframe object
`i`	Row indices
`j`	Column indices
`...`	Additional arguments
`drop`	If TRUE, simplify to vector when possible

Value

A subset aniframe

`[[.aniframe` *Extract single column from aniframe with [[*

Description

Extract single column from aniframe with [[

Usage

```
## S3 method for class 'aniframe'
x[[i, ...]]
```

Arguments

x	An aniframe object
i	Column index or name
...	Additional arguments

Value

A vector or data frame

[[<-.aniframe *Column assignment for aniframe with [[<-*

Description

Column assignment for aniframe with [[<-

Usage

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

Arguments

x	An aniframe object
i	Column index or name
...	Additional arguments
value	Replacement value

Value

Modified aniframe

[<-.aniframe *Subset assignment for aniframe with [<-*

Description

Subset assignment for aniframe with [<-

Usage

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

Arguments

x	An aniframe object
i	Row indices
j	Column indices
...	Additional arguments
value	Replacement values

Value

Modified aniframe

$.aniframe *Extract column from aniframe with $*

Description

Extract column from aniframe with $

Usage

```
## S3 method for class 'aniframe'
x$name
```

Arguments

x	An aniframe object
name	Column name

Value

A vector

$<-.aniframe *Column assignment for aniframe with $<-*

Description

Column assignment for aniframe with $<-

Usage

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

Arguments

x	An aniframe object
name	Column name
value	Replacement value

Value

Modified aniframe

---

add_connections *Add connections to an aniframe*

Description

[Experimental]

Append one or more from/to pairs to the existing connections for a variable. from and to may be either single strings or character vectors of equal length (one connection per element).

Usage

```
add_connections(data, from, to, variable = "keypoint")
```

Arguments

data	An aniframe object.
from	Character vector of source endpoints.
to	Character vector of target endpoints. Must be the same length as from.
variable	Character scalar. Name of the variable the connections relate to (must be in variables_what or variables_when). Defaults to "keypoint".

Details

No deduplication is performed — duplicates of an existing pair will appear twice in the resulting table. Endpoints not found in data[[variable]] produce a warning but are still appended.

Value

The aniframe with the new connections appended.

See Also

set_connections(), get_connections(), remove_connections()

Examples

```
## Not run:
data <- example_aniframe()
data <- add_connections(data, from = "head", to = "neck")
data <- add_connections(
  data,
  from = c("neck", "neck"),
  to = c("shoulder_right", "shoulder_left")
)

## End(Not run)
```

`arrange.aniframe` *Arrange rows of an aniframe*

Description

Arrange rows of an aniframe

Usage

```
## S3 method for class 'aniframe'
arrange(.data, ..., .by_group = FALSE)
```

Arguments

`.data`	An aniframe object
`...`	Variables to order by
`.by_group`	If TRUE, arrange within groups

Value

An arranged aniframe

`as.data.frame.aniframe` *Convert aniframe to regular data frame*

Description

Convert aniframe to regular data frame

Usage

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

Arguments

x	An aniframe object
...	Additional arguments

Value

A regular data frame

---

as_aniframe *Convert a data frame to aniframe*

Description

Convert a data frame to aniframe

Usage

```
as_aniframe(
  data,
  metadata = list(),
  variables_what = NULL,
  variables_when = NULL,
  variables_where = NULL
)
```

Arguments

data	A data frame with movement data.
metadata	A list of metadata to attach to the aniframe.
variables_what	Character vector of identity columns that together define a unique entity. Defaults to c("individual", "keypoint").
variables_when	Character vector of temporal columns that together define a unique timepoint. Defaults to "time".
variables_where	Character vector of spatial columns that together define position. If NULL, detected from data.

Value

An aniframe object

convert_nan_to_na *Convert NaN to NA in numeric columns*

Description

Replaces all NaN values with NA in numeric columns of a data frame.

Usage

```
convert_nan_to_na(data)
```

Arguments

data	A data frame.

Value

A data frame with NaN values replaced by NA in numeric columns.

default_metadata *Default metadata structure*

Description

Returns a list containing the default metadata fields and their initial values for an aniframe object. Most fields are initialized as NA and should be set appropriately for your data.

Usage

```
default_metadata()
```

Value

A named list with the following fields:

- source: Data source identifier (character, NA)
- source_version: Version of the data source (character, NA)
- filename: Original filename(s) (character vector, NA). Accepts a vector of length 1 or more — readers that load from multiple files (e.g. aniread::read_trackball()) populate this with all source paths.
- sampling_rate: Sampling rate in Hz (numeric, NA)
- start_datetime: Start date and time of recording (POSIXct, NA)
- reference_frame: Reference frame (factor, "allocentric")
- coordinate_system: Coordinate system (factor, "cartesian")

- `origin`: Location of the (0,0) coordinate relative to the recording frame (factor, "bottom_left"). Permitted values: "bottom_left", "top_left".
- `y_height`: Height of the recording frame in y-axis units (numeric, NA). Used by `set_origin()` to reflect y coordinates when switching origin conventions.
- `connections`: Named list of connection tables, one per identity or temporal variable (typically `keypoint` for skeletons; could also be `individual` for social networks). Each entry is a 2-column tibble of `from`/`to` pairs. Default is an empty list. Manage via `set_connections()`, `get_connections()`, `add_connections()` and `remove_connections()`.

See Also

`set_metadata()`, `get_metadata()`

---

deg_to_rad *Convert degrees to radians*

Description

Convert degrees to radians

Usage

```
deg_to_rad(x)
```

Arguments

`x`	Numeric vector of angles (degrees).

Value

Numeric vector of angles expressed in radians.

---

ensure_is_aniframe *Ensure object is an aniframe*

Description

Ensure object is an aniframe

Usage

```
ensure_is_aniframe(x)
```

Arguments

`x`	An object to test

Value

Error if not an aniframe

`ensure_is_cartesian` *Internal guard for Cartesian checks*

Description

Stops with a clear error message if `data` is not Cartesian.

Usage

```
ensure_is_cartesian(data)
```

Arguments

`data`	A data frame.

`ensure_is_cartesian_1d` *Internal guard for 1-D Cartesian checks*

Description

Internal guard for 1-D Cartesian checks

Usage

```
ensure_is_cartesian_1d(data)
```

Arguments

`data`	A data frame.

`ensure_is_cartesian_2d` *Internal guard for 2-D Cartesian checks*

Description

Internal guard for 2-D Cartesian checks

Usage

```
ensure_is_cartesian_2d(data)
```

Arguments

`data`	A data frame.

`ensure_is_cartesian_3d` *Internal guard for 3-D Cartesian checks*

Description

Internal guard for 3-D Cartesian checks

Usage

```
ensure_is_cartesian_3d(data)
```

Arguments

`data`	A data frame.

`ensure_is_cylindrical` *Internal guard for cylindrical checks*

Description

Internal guard for cylindrical checks

Usage

```
ensure_is_cylindrical(data)
```

Arguments

`data`	A data frame.

ensure_is_polar *Internal guard for polar checks*

Description

Internal guard for polar checks

Usage

```
ensure_is_polar(data)
```

Arguments

data	A data frame.

ensure_is_spherical *Internal guard for spherical checks*

Description

Internal guard for spherical checks

Usage

```
ensure_is_spherical(data)
```

Arguments

data	A data frame.

example_aniframe *Create example aniframe data*

Description

Generates a synthetic aniframe object with random coordinates for testing and demonstration purposes. The function creates a complete design with all combinations of time points, individuals, keypoints, trials, and sessions.

Usage

```
example_aniframe(
  n_obs = 50,
  n_individuals = 3,
  n_keypoints = 11,
  n_trials = 1,
  n_sessions = 1,
  n_dims = 2
)
```

Arguments

n_obs	Integer. Number of time observations per combination. Default is 50.
n_individuals	Integer. Number of individuals to simulate. Default is 3.
n_keypoints	Integer. Number of keypoints per individual (max 11). Default is 11. When set to 1, only "centroid" is used. Otherwise, anatomical keypoints are used (head, neck, shoulders, etc.).
n_trials	Integer. Number of trials per session. Default is 1.
n_sessions	Integer. Number of sessions. Default is 1.
n_dims	Integer. Number of spatial dimensions (1, 2, or 3). Default is 2. If 1, only x coordinates are generated. If 2, x and y coordinates are generated. If 3, x, y, and z coordinates are generated.

Value

An aniframe object containing randomly generated tracking data with columns for individual, keypoint, time, trial, session, and spatial coordinates (x, y, and/or z depending on n_dims). The coordinates are drawn from a standard normal distribution.

Examples

```
# Create a basic example with default parameters (2D)
example_aniframe()

# Create a 1D example
example_aniframe(n_dims = 1)

# Create a 3D example
example_aniframe(n_dims = 3)

# Create a smaller example with 2 individuals and 5 keypoints
example_aniframe(n_individuals = 2, n_keypoints = 5)

# Create example with multiple trials and sessions
example_aniframe(n_obs = 100, n_trials = 3, n_sessions = 2)

# Create minimal example with just centroid in 3D
example_aniframe(n_keypoints = 1, n_dims = 3)
```

filter.aniframe *Filter rows of an aniframe*

Description

Filter rows of an aniframe

Usage

```
## S3 method for class 'aniframe'
filter(.data, ..., .preserve = FALSE)
```

Arguments

.data	An aniframe object
...	Logical predicates
.preserve	Keep group structure

Value

A filtered aniframe

get_connections *Get connections from an aniframe*

Description

[Experimental]

Read the connections currently stored on an aniframe. Returns the full named list of from/to tibbles by default, or a single tibble when variable is supplied.

Usage

```
get_connections(data, variable = NULL)
```

Arguments

data	An aniframe object.
variable	Optional character scalar. When NULL (default), returns the full named list of connection tables (one per variable). When a variable name, returns just that variable's from/to tibble (an empty tibble if no connections are set for that variable).

Value

A named list of tibbles (when `variable` is NULL), or a single 2-column tibble.

See Also

`set_connections()`, `add_connections()`, `remove_connections()`

`get_metadata` *Get metadata*

Description

Get metadata

Usage

```
get_metadata(data, fields = NULL)
```

Arguments

`data`	aniframe
`fields`	If only specific metadata fields should be returned.

Value

the metadata associated with the aniframe

`group_by.aniframe` *Group an aniframe*

Description

Group an aniframe

Usage

```
## S3 method for class 'aniframe'
group_by(.data, ...)
```

Arguments

`.data`	An aniframe object
`...`	Variables to group by

Value

A grouped aniframe

is_aniframe *Check if object is an aniframe*

Description

Check if object is an aniframe

Usage

```
is_aniframe(x)
```

Arguments

x	An object to test

Value

Logical: TRUE if x inherits from aniframe

is_cartesian *Test whether a data frame uses a Cartesian coordinate system*

Description

Returns TRUE if the data frame satisfies *any* of the 1-D, 2-D or 3-D Cartesian checks defined in the helper functions.

Usage

```
is_cartesian(data)
```

Arguments

data	A data frame.

Value

Logical scalar.

is_cartesian_1d *Test for a 1-D Cartesian coordinate system*

Description

The data frame must contain **exactly one** of `x`, `y` or `z` and none of the polar columns (`rho`, `phi`, `theta`).

Usage

```
is_cartesian_1d(data, stop = FALSE)
```

Arguments

`data`	A data frame.
`stop`	Unused placeholder kept for API compatibility.

Value

Logical scalar (invisible).

is_cartesian_2d *Test for a 2-D Cartesian coordinate system*

Description

Requires columns `x` and `y`. Column `z` may be present only if it is completely `NA`.

Usage

```
is_cartesian_2d(data)
```

Arguments

`data`	A data frame.

Value

Logical scalar (invisible).

is_cartesian_3d *Test for a 3-D Cartesian coordinate system*

Description

Requires non-missing columns `x`, `y` and `z`.

Usage

```
is_cartesian_3d(data)
```

Arguments

`data`	A data frame.

Value

Logical scalar (invisible).

is_cylindrical *Test whether a data frame uses a cylindrical coordinate system*

Description

Requires `rho`, `phi` and `z`; forbids `theta`.

Usage

```
is_cylindrical(data)
```

Arguments

`data`	A data frame.

Value

Logical scalar.

is_polar *Test whether a data frame uses a polar coordinate system*

Description

Requires columns `rho` and `phi` and forbids `theta` or `z`.

Usage

```
is_polar(data)
```

Arguments

`data`	A data frame.

Value

Logical scalar.

is_spherical *Test whether a data frame uses a spherical coordinate system*

Description

Requires `rho`, `phi` and `theta`; forbids `z`.

Usage

```
is_spherical(data)
```

Arguments

`data`	A data frame.

Value

Logical scalar.

mutate.aniframe *Mutate columns in an aniframe*

Description

Mutate columns in an aniframe

Usage

```
## S3 method for class 'aniframe'
mutate(.data, ...)
```

Arguments

.data	An aniframe object
...	Name-value pairs of expressions

Value

An aniframe with modified columns

names<-.aniframe *Rename columns with names<-*

Description

Rename columns with names<-

Usage

```
## S3 replacement method for class 'aniframe'
names(x) <- value
```

Arguments

x	An aniframe object
value	New column names

Value

Modified aniframe

rad_to_deg *Convert radians to degrees*

Description

Convert radians to degrees

Usage

```
rad_to_deg(x)
```

Arguments

x	Numeric vector of angles (radians).

Value

Numeric vector of angles expressed in degrees.

relocate.aniframe *Relocate columns in an aniframe*

Description

Relocate columns in an aniframe

Usage

```
## S3 method for class 'aniframe'
relocate(.data, ...)
```

Arguments

.data	An aniframe object
...	Columns to relocate

Value

An aniframe with relocated columns

`remove_connections` *Remove connections from an aniframe*

Description

[Experimental]

Remove `from`/`to` pairs from the connections of a variable. Matching is exact and order-sensitive: `remove_connections(data, "a", "b")` removes the pair `(from = "a", to = "b")` but does *not* remove `(from = "b", to = "a")`. Call twice with swapped arguments if you want both directions gone.

Usage

```
remove_connections(data, from, to, variable = "keypoint")
```

Arguments

`data`	An aniframe object.
`from`	Character vector of source endpoints to remove.
`to`	Character vector of target endpoints to remove. Must be the same length as `from`.
`variable`	Character scalar. Name of the variable. Defaults to `"keypoint"`.

Value

The aniframe with matching connections removed.

See Also

`set_connections()`, `get_connections()`, `add_connections()`

Examples

```
## Not run:
data <- example_aniframe() |>
  add_connections(from = c("head", "neck"), to = c("neck", "shoulder_right"))
data <- remove_connections(data, from = "head", to = "neck")

## End(Not run)
```

rename.aniframe *Rename columns in an aniframe*

Description

Rename columns in an aniframe

Usage

```
## S3 method for class 'aniframe'
rename(.data, ...)
```

Arguments

.data	An aniframe object
...	Name-value pairs for renaming

Value

An aniframe with renamed columns

select.aniframe *Select columns from an aniframe*

Description

Select columns from an aniframe

Usage

```
## S3 method for class 'aniframe'
select(.data, ...)
```

Arguments

.data	An aniframe object
...	Columns to select

Value

An aniframe with selected columns

`set_connections` *Set the connections for a variable*

Description

[Experimental]

Replace the connections (e.g. skeleton edges between keypoints, edges of a social network between individuals) for a single variable. Connections are stored as a 2-column `from`/`to` tibble; storage preserves the order supplied so downstream consumers can interpret the table as either directed or undirected.

Usage

```
set_connections(data, connections, variable = "keypoint")
```

Arguments

`data`	An aniframe object.
`connections`	One of: • a 2-column data.frame with columns `from` and `to`, • a list of length-2 character vectors (each `c(from, to)`), • `NULL` to clear the connections for `variable`.
`variable`	Character scalar. Name of the identity (`variables_what`) or temporal (`variables_when`) variable the connections relate to. Defaults to `"keypoint"`.

Details

If any `from`/`to` value isn't found in the corresponding column of `data`, a warning is emitted but the connection is kept — the value may legitimately be missing in this particular recording while being valid elsewhere.

Value

The aniframe with updated `connections` metadata.

See Also

`get_connections()`, `add_connections()`, `remove_connections()`

Examples

```
## Not run:
data <- example_aniframe()

# Implicit by position (element[1] = from, element[2] = to)
data <- set_connections(
  data,
```

```
  list(
    c("head", "neck"),
    c("neck", "shoulder_right"),
    c("neck", "shoulder_left"),
    c("shoulder_right", "hip_right"),
    c("shoulder_left", "hip_left")
  )
)

# Explicit names within each pair
data <- set_connections(
  data,
  list(
    c(from = "head", to = "neck"),
    c(from = "neck", to = "shoulder_right")
  )
)

# Or as a 2-column data.frame
data <- set_connections(
  data,
  data.frame(
    from = c("head", "neck"),
    to   = c("neck", "shoulder_right")
  )
)

## End(Not run)
```

`set_metadata` *Set metadata for an aniframe*

Description

Sets or updates metadata for an aniframe object. Metadata can be provided either as named arguments or as a list. If the aniframe already has metadata, the new values will be merged with existing values, with new values taking precedence.

Character values for factor fields will be automatically converted to factors if they match allowed levels.

Default metadata fields include:

- `source`: Data source identifier
- `source_version`: Version of the data source
- `filename`: Original filename(s) — accepts a character vector (length 1 or more) for readers that load from multiple files
- `sampling_rate`: Sampling rate in Hz
- `start_datetime`: Start date and time of recording

- `reference_frame`: Reference frame (default: "allocentric")
- `coordinate_system`: Coordinate system (default: "cartesian")
- `origin`: Location of the (0,0) coordinate (default: "bottom_left")
- `y_height`: Height of the recording frame in y-axis units (default: NA)

For backwards compatibility, the deprecated field `point_of_reference` is accepted as an alias for `origin` and emits a deprecation warning.

Usage

```
set_metadata(data, ..., metadata = NULL)
```

Arguments

`data`	An aniframe object
`...`	Named metadata values (e.g., `sampling_rate = 30, source = "sleap"`)
`metadata`	Alternatively, a named list of metadata. Cannot be used simultaneously with `...`

Value

The aniframe object with updated metadata

See Also

`get_metadata()`, `default_metadata()`

Examples

```
## Not run:
# Set metadata using named arguments
data <- set_metadata(data, sampling_rate = 30, source = "sleap")

# Set metadata using a list
md <- list(sampling_rate = 30, source = "sleap")
data <- set_metadata(data, metadata = md)

## End(Not run)
```

`set_origin` *Set the coordinate origin*

Description

Sets or updates the `origin` metadata field, which records where the (0,0) coordinate sits relative to the recording frame. When the new origin differs from the current one, the y coordinates are reflected around `y_height` so the data is expressed in the new convention.

Usage

```
set_origin(data, origin)
```

Arguments

`data`	An aniframe object.
`origin`	Character. One of `"bottom_left"` or `"top_left"`.

Details

The flip uses the formula `y_new = y_height - y_old`. The `y_height` metadata field must therefore be set when the origin actually changes; if it is `NA`, this function errors and asks the user to set it via `set_y_height()`. When the supplied `origin` matches the current value, the data is returned unchanged.

Value

The aniframe with reflected y coordinates (when the origin changed) and updated `origin` metadata.

See Also

`set_y_height()`

Examples

```
## Not run:
data <- example_aniframe()
data <- set_y_height(data, y_height = 1080)
data <- set_origin(data, origin = "top_left")

## End(Not run)
```

`set_sampling_rate` *Set the sampling rate of an aniframe object*

Description

Sets the sampling rate (in Hz) for an aniframe object and optionally converts time values from frames to seconds. If the data is already in SI time units, only the metadata is updated without modifying the time values.

Usage

```
set_sampling_rate(data, sampling_rate)
```

Arguments

`data`	An aniframe object containing time data.
`sampling_rate`	Numeric value specifying the sampling rate in Hz (samples per second). For example, a sampling rate of 30 means 30 frames per second.

Details

The function performs the following operations:

- Checks the current unit_time in the object's metadata
- If unit_time is "frame" or "unknown", converts time values to seconds using the formula: time_in_seconds = time_in_frames / sampling_rate
- If unit_time is already an SI unit (ms, s, m, h), leaves time values unchanged and issues an informational message
- Updates the sampling_rate in the object's metadata regardless of the current unit_time

This function is particularly useful when working with motion capture or video data where time is initially recorded as frame numbers.

Value

An aniframe object with updated sampling_rate metadata and, if applicable, time values converted from frames to seconds.

Examples

```
## Not run:
# Set sampling rate for data in frames (converts to seconds)
data_with_rate <- set_sampling_rate(data, sampling_rate = 30)

# Set sampling rate for data already in SI units (updates metadata only)
data_with_rate <- set_sampling_rate(data, sampling_rate = 100)

## End(Not run)
```

set_unit_angle *Set the angular unit of an aniframe object*

Description

Converts angular columns in an aniframe between degrees (`"deg"`) and radians (`"rad"`), and updates the `unit_angle` metadata to match.

Spatial angular columns (`phi`, `theta`) are converted automatically whenever they are present in the data, so polar/cylindrical/spherical coordinates always stay consistent with the declared unit. Additional angular columns (e.g. heading or orientation columns named outside the polar family) can be supplied via `cols`.

Usage

```
set_unit_angle(data, to_unit, cols = NULL)
```

Arguments

`data`	An aniframe object containing angular data.
`to_unit`	Character string specifying the target angular unit. Must be one of `c("rad", "deg")` (the levels of `default_metadata()$unit_angle`).
`cols`	Optional character vector of additional angular column names to convert. The spatial angular columns `phi` and `theta` are detected automatically and need not be listed; pass `cols` only for non-spatial angular columns (e.g. `"heading"`). All listed columns must be present and numeric.

Details

If the current `unit_angle` already matches `to_unit`, an informational message is shown and the data are returned unchanged (apart from the metadata round-trip).

Value

An aniframe object with the relevant angular columns converted to the specified unit and `unit_angle` metadata updated accordingly.

Examples

```
## Not run:
# Polar data: phi is converted automatically
df <- data.frame(time = 1:3, rho = 1:3, phi = c(0, pi / 2, pi))
anif <- as_aniframe(df)
anif_deg <- set_unit_angle(anif, to_unit = "deg")

# Custom angular columns alongside the spatial ones
anif2 <- set_unit_angle(anif, to_unit = "deg", cols = "heading")
```

```
## End(Not run)
```

`set_unit_space` *Set the spatial unit of an aniframe object*

Description

Converts spatial coordinates (x, y, z) in an aniframe object to a different unit of measurement. The function handles both automatic unit conversion between standard units and custom calibration from pixel or arbitrary units.

Usage

```
set_unit_space(data, to_unit, calibration_factor = 1)
```

Arguments

`data`	An aniframe object containing spatial coordinate data.
`to_unit`	Character string specifying the target spatial unit. Must be one of the permitted units defined in `default_metadata()$unit_space`.
`calibration_factor`	Numeric value for scaling spatial coordinates. Default is 1. When converting from standard units (mm, cm, m), this is ignored and the appropriate conversion factor is calculated automatically. When converting from "px" or "unknown", you must provide a calibration factor to define the relationship between the current units and the target unit.

Details

The function performs the following operations:

- Validates that `to_unit` is a permitted spatial unit
- Determines the current spatial unit from the object's metadata
- If converting from standard units (mm, cm, m) to another standard unit, automatically calculates the conversion factor
- If converting from "px" or "unknown" units with `calibration_factor = 1`, issues an informational message and returns data unchanged
- Applies the calibration factor to all spatial columns (x, y, z) that exist in the data
- Updates the object's unit_space metadata

Value

An aniframe object with spatial coordinates converted to the specified unit and updated metadata reflecting the new unit_space.

Examples

```
## Not run:
# Convert from millimeters to centimeters (automatic conversion)
data_cm <- set_unit_space(data, to_unit = "cm")

# Convert from pixels to millimeters with custom calibration
# (e.g., 1 pixel = 0.5 mm)
data_mm <- set_unit_space(data, to_unit = "mm", calibration_factor = 0.5)

## End(Not run)
```

`set_unit_time` *Set the temporal unit of an aniframe object*

Description

Converts time values in an aniframe object to a different unit of measurement. The function handles both automatic unit conversion between standard time units and custom calibration from frame or arbitrary units.

Usage

```
set_unit_time(data, to_unit, calibration_factor = 1)
```

Arguments

`data`	An aniframe object containing time data.
`to_unit`	Character string specifying the target time unit. Must be one of the permitted units defined in `default_metadata()$unit_time` (typically "ms", "s", "m", "h" for milliseconds, seconds, minutes, hours).
`calibration_factor`	Numeric value for scaling time values. Default is 1. When converting from standard time units (ms, s, m, h), this is ignored and the appropriate conversion factor is calculated automatically. When converting from "frame" or "unknown" units, you must provide a calibration factor to define the relationship between the current units and the target unit.

Details

The function performs the following operations:

- Validates that `to_unit` is a permitted time unit
- Determines the current time unit from the object's metadata
- If converting from standard time units (ms, s, m, h) to another standard unit, automatically calculates the conversion factor

- If converting from "frame" or "unknown" units with `calibration_factor = 1`, issues an informational message and returns data unchanged
- Applies the calibration factor to the time column
- Updates the object's unit_time metadata

Value

An aniframe object with time values converted to the specified unit and updated metadata reflecting the new unit_time.

Examples

```
## Not run:
# Convert from milliseconds to seconds (automatic conversion)
data_s <- set_unit_time(data, to_unit = "s")

# Convert from frames to seconds with custom calibration
# (e.g., 30 frames per second means 1 frame = 1/30 seconds)
data_s <- set_unit_time(data, to_unit = "s", calibration_factor = 1/30)

# Convert from hours to minutes (automatic conversion)
data_m <- set_unit_time(data, to_unit = "m")

## End(Not run)
```

`set_y_height` *Set the y-axis frame height*

Description

Sets or updates the `y_height` metadata field, which records the height of the recording frame in y-axis units. Used by `set_origin()` when reflecting y coordinates between origin conventions (e.g. `bottom_left` <-> `top_left`).

Reader functions in `aniread` populate `y_height` automatically from the source (e.g. video frame height). For aniframes constructed manually, `as_aniframe()` falls back to `max(y)`. Use this function to set the true frame height when the auto-fallback is not appropriate.

Usage

```
set_y_height(data, y_height)
```

Arguments

`data`	An aniframe object.
`y_height`	A single positive finite numeric value.

Value

The aniframe with updated `y_height` metadata.

See Also

`set_origin()`

Examples

```
## Not run:
data <- example_aniframe()
data <- set_y_height(data, y_height = 1080)

## End(Not run)
```

`slice.aniframe` *Slice rows from an aniframe*

Description

Slice rows from an aniframe

Usage

```
## S3 method for class 'aniframe'
slice(.data, ..., .preserve = FALSE)
```

Arguments

`.data`	An aniframe object
`...`	Integer row positions
`.preserve`	Keep group structure

Value

A sliced aniframe

tbl_sum.aniframe *Custom tibble summary for aniframe*

Description

Builds the print header rows shown above an aniframe. The set of rows is driven by the metadata: one row per column listed in `variables_what` and one row per column in `variables_when` (excluding `time`). This means custom identity/temporal variables (e.g. `track`, `model`, `session`) appear automatically, and rows are omitted entirely when their column is absent.

Usage

```
## S3 method for class 'aniframe'
tbl_sum(x, ...)
```

Arguments

`x`	An aniframe object
`...`	Additional arguments (unused)

Value

Named character vector with summary information

ungroup.aniframe *Ungroup an aniframe*

Description

Ungroup an aniframe

Usage

```
## S3 method for class 'aniframe'
ungroup(x, ...)
```

Arguments

`x`	An aniframe object
`...`	Additional arguments passed to dplyr::ungroup

Value

An ungrouped aniframe

Index