Package 'SamsaRaLight'

November 23, 2026

Title Simulate Tree Light Transmission Using Ray-Tracing

Version 1.1.0

Description Provides tools to simulate light transmission in forest stands using three-dimensional ray-tracing. The package allows users to build virtual stands from tree inventories and to estimate (1) light intercepted by individual trees, (2) light reaching the forest floor, and (3) light at virtual sensors. The package is designed for ecological and forestry applications, including the analysis of light competition, tree growth, and forest regeneration. The implementation builds on the individual-based ray-tracing model SamsaraLight developed by Courbaud et al. (2003) <doi:10.1016/S0168-1923(02)00254-X>.

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 4.0)

Imports concaveman, data.table, dplyr, ggforce, ggnewscale, ggplot2, grid, httr, patchwork, Rcpp (>= 1.1.0), RhpcBLASctl, scales, sf, sfheaders, tidyr

LinkingTo Rcpp (>= 1.1.0)

LazyData true

RoxygenNote 7.3.3

Suggests cowplot, knitr, purrr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

URL https://natheob.github.io/SamsaRaLight/

BugReports https://github.com/natheob/SamsaRaLight/issues/

NeedsCompilation yes

Author Natheo Beauchamp [aut, cre] (R package developer, ORCID: <https://orcid.org/0009-0007-9103-5194>),
Gauthier Ligot [aut] (Java Capsis developer and algorithm improvement, ORCID: <https://orcid.org/0000-0002-5508-4358>),
Francois de Coligny [aut] (Java Capsis developer, ORCID: <https://orcid.org/0000-0002-8538-3009>),

Maxime Jaunatre [aut] (R package support developer, ORCID:
<https://orcid.org/0009-0002-2816-1677>),
Benoit Courbaud [aut, cph] (Algorithm and theory creator, ORCID:
<https://orcid.org/0000-0002-3050-9559>)

Maintainer Natheo Beauchamp <beauchamp.natheo@gmail.com>

Repository CRAN

Date/Publication 2026-11-23 14:10:38 UTC

Contents

check_coordinates . . . 2
check_inventory . . . 3
check_monthly_radiations . . . 5
check_polygon . . . 6
check_sensors . . . 7
create_sl_stand . . . 8
create_xy_from_lonlat . . . 10
get_bottom_azimut . . . 11
get_monthly_radiations . . . 12
get_z . . . 13
plot.sl_output . . . 13
plot.sl_stand . . . 14
plot_inventory . . . 15
print.sl_output . . . 16
print.sl_stand . . . 17
run_sl . . . 17
run_sl_advanced . . . 20
SamsaRaLight_data . . . 22
summary.sl_output . . . 24
summary.sl_stand . . . 25

Index **26**

`check_coordinates` *Check coordinate columns and determine whether conversion is required*

Description

This function checks whether a data frame contains valid planar coordinates (`x`, `y`) or geographic coordinates (`lon`, `lat`). If geographic coordinates are detected, the user is informed that they must be converted to a planar coordinate system using `create_xy_from_lonlat()`.

Usage

```
check_coordinates(df, verbose = TRUE)
```

Arguments

df	A data.frame containing spatial coordinates.
verbose	Logical. If TRUE, informative messages are printed.

Value

Logical. Invisibly returns TRUE if coordinates must be converted from longitude/latitude to planar coordinates, and FALSE if planar coordinates are already present.

Examples

```
df_xy <- data.frame(x = 1:3, y = 4:6)
check_coordinates(df_xy)

df_lonlat <- data.frame(lon = c(10, 11), lat = c(45, 46))
check_coordinates(df_lonlat, verbose = TRUE)
```

---

check_inventory *Check the format and validity of a tree inventory data.frame*

Description

This function checks whether a tree inventory data.frame is correctly formatted to be used as input for the ray-tracing model. It verifies the presence, type, and validity of mandatory and optional variables describing the geometry and attributes of trees within a forest stand.

Usage

```
check_inventory(trees_inv, verbose = TRUE)
```

Arguments

trees_inv	A data.frame with one row per tree and the following columns: **id_tree** Unique identifier of the tree (positive integer, no duplicates). **x** X position of the tree within the stand (numeric, meters, planar coordinates). Optional if lon and lat are provided. **y** Y position of the tree within the stand (numeric, meters, planar coordinates). Optional if lon and lat are provided. **lon** Longitude of the tree location (numeric, decimal degrees). Optional; if provided with lat, coordinates are converted internally to a UTM planar coordinate system in create_sl_stand(). **lat** Latitude of the tree location (numeric, decimal degrees). Optional; see lon. **species** Species name (character). **dbh_cm** Diameter at breast height (1.3 m, in cm). **crown_type** Type of crown geometry. One of "E", "P", "2E", "8E", or "4P".

h_m Total tree height (numeric, meters).

hbase_m Height of the crown base (numeric, meters).

hmax_m Height of the maximum crown radius (numeric, meters). Required only if at least one tree has a crown type `"2E"` or `"8E"`. For other crown types, the column is optional and the value is internally computed.

rn_m Crown radius toward North (numeric, meters).

rs_m Crown radius toward South (numeric, meters).

re_m Crown radius toward East (numeric, meters).

rw_m Crown radius toward West (numeric, meters).

crown_lad Leaf Area Density (m2/m3). Required for turbid medium interception (default, and always used in `run_sl()`); optional if `turbid_medium = FALSE` in `run_sl_advanced()`.

crown_openness Crown openness (unitless), optional if turbid medium interception. Required if the argument `turbid_medium = FALSE` in the advanced function `run_sl_advanced()` (for porous envelope interception). Otherwise, the basic function `run_sl()` will automatically computed interception in a turbid medium using the `crown_lad` variable.

`verbose` Logical; if `TRUE`, informative messages and warnings are printed.

Details

The function performs the following checks and validations:

1. Ensures `trees_inv` is a non-empty data.frame with all required columns.
2. Checks that `id_tree` values are unique.
3. Checks that either planar coordinates (`x`, `y`) or geographic coordinates (`lon`, `lat`) are provided. If only `lon` and `lat` are supplied, they are converted to planar UTM coordinates in `create_sl_stand()`. The `plot_inventory()` function requires planar coordinates and cannot be used before this conversion.
4. Validates numeric columns (`dbh_cm`, `h_m`, `hbase_m`, `rn_m`, `rs_m`, `re_m`, `rw_m`) are numeric and non-negative.
5. Verifies that `crown_type` values are one of `"E"`, `"P"`, `"2E"`, `"8E"`, or `"4P"`.
6. Ensures crown radii are present according to crown type.
7. Checks that `hmax_m` is provided when required and lies between `hbase_m` and `h_m`.
8. Ensures `hbase_m < h_m`.
9. Verifies that each tree has the crown LAD defined.
10. Provides informative error messages and warnings for all invalid conditions.

Value

Invisibly returns `TRUE` if all checks pass.

Examples

```
# Using the example dataset from the package
data_prenovel <- SamsaRaLight::data_prenovel
trees <- data_prenovel$trees

# Check the inventory
check_inventory(trees)

# Quiet mode
check_inventory(trees, verbose = FALSE)
```

`check_monthly_radiations` *Validate monthly radiation input*

Description

Checks that a monthly radiation data.frame is correctly formatted and physically valid for use by the light interception and ray-tracing model. The table must contain exactly 12 months of radiation data.

Usage

```
check_monthly_radiations(x, verbose = TRUE)
```

Arguments

`x`	A data.frame with monthly radiation values, typically produced by `get_monthly_radiations`.
`verbose`	Logical; if `TRUE`, informative messages are printed.

Details

The input must contain the following columns:

month Integer month number (1-12)

Hrad Monthly global horizontal irradiation (MJ/m2)

DGratio Diffuse-to-global radiation ratio (unitless, 0-1)

The function checks:

1. The object is a data.frame.
2. Required columns are present.
3. There are exactly 12 months.
4. Each month (1-12) is present exactly once.
5. Data are numeric and finite.
6. Hrad strictly positive.
7. DGratio between 0 and 1.
8. Months are in increasing order.

Value

Invisibly returns TRUE if all checks pass.

Examples

```
# Using the example dataset from the package
data_prenovel <- SamsaRaLight::data_prenovel
radiations <- data_prenovel$radiations

# Check the inventory
check_monthly_radiations(radiations)

# Quiet mode
check_monthly_radiations(radiations, verbose = FALSE)
```

---

check_polygon *Check and validate a polygon defined by vertices*

Description

This function converts a data.frame of polygon vertices into an sf POLYGON and checks its validity. If the polygon is invalid, it attempts to fix it.

Usage

```
check_polygon(core_polygon_df, trees_inv, sensors = NULL, verbose = TRUE)
```

Arguments

core_polygon_df	A data.frame with columns x and y defining polygon vertices
trees_inv	A data.frame with one row per tree. See check_inventory for the required structure and validated columns.
sensors	Optional data.frame defining position and height of the sensor within the stand. See check_sensors for the required structure and validated columns.
verbose	Logical. If TRUE, warnings are printed

Value

A data.frame of polygon vertices (x, y):

- unchanged if valid
- modified if fixed (with a warning)

Examples

```
data_prenovel <- SamsaRaLight::data_prenovel

# Validate polygon
check_polygon(data_prenovel$core_polygon, data_prenovel$trees)
```

`check_sensors` *Check the format and validity of a sensor position data.frame*

Description

This function checks whether a sensor data.frame is correctly formatted to be used as input for the ray-tracing model. It verifies the presence, type, and validity of mandatory variables describing the position and height of sensors within a forest stand.

Usage

```
check_sensors(sensors, verbose = TRUE)
```

Arguments

`sensors`	A data.frame with one row per sensor and the following columns: **id_sensor** Unique identifier of the sensor (positive integer, no duplicates) **x** X position of the sensor (numeric, meters) **y** Y position of the sensor (numeric, meters) **h_m** Height above ground of the sensor (numeric, meters)
`verbose`	Logical; if TRUE, informative messages are printed.

Value

Invisibly returns TRUE if all checks pass.

Examples

```
sensors <- data.frame(
  id_sensor = 1:3,
  x = c(10, 20, 30),
  y = c(5, 15, 25),
  h_m = c(1.5, 2.0, 1.8)
)
check_sensors(sensors)
```

`create_sl_stand` *Create a virtual stand from a tree inventory*

Description

This function builds a virtual forest stand from a user-provided tree inventory. The inventory zone (core polygon) can optionally be modified (e.g., replaced by an enclosing rectangle or an axis-aligned rectangle) before constructing the stand. Trees are spatially shifted so that the resulting inventory zone is centered within a regular grid of square cells. Optionally, additional trees can be added around the core inventory area to match its basal area per hectare.

Usage

```
create_sl_stand(
  trees_inv,
  cell_size,
  latitude,
  slope,
  aspect,
  north2x,
  sensors = NULL,
  core_polygon_df = NULL,
  modify_polygon = c("none", "rect", "aarect"),
  fill_around = FALSE,
  verbose = TRUE
)
```

Arguments

`trees_inv`	A data.frame with one row per tree. See check_inventory for the required structure and validated columns.
`cell_size`	Numeric. Side length of square cells composing the stand (meters).
`latitude`	Numeric. Latitude of the stand (degrees).
`slope`	Numeric. Slope of the plot (degrees).
`aspect`	Numeric. Aspect of the slope, defined as the azimuth of the downslope direction, clockwise from North (degrees). (0 degrees: North-facing slope, 90 degrees: East-facing slope, 180 degrees: South-facing slope, 270 degrees: West-facing slope)
`north2x`	Numeric. Clockwise angle from North to the X-axis (degrees). The default 90 degrees corresponds to a Y-axis oriented toward true North (0 degrees: x-axis points North, 90 degrees: x-axis points East, 180 degrees: x-axis points South, 270 degrees: x-axis points West).
`sensors`	Optional data.frame defining position and height of the sensor within the stand. See check_sensors for the required structure and validated columns.

`core_polygon_df`
: Optional data.frame defining the core inventory polygon. Must contain columns `x` and `y`. If `NULL`, a concave hull is automatically computed from tree positions.

`modify_polygon`
: Character. Defines how the inventory polygon is modified. One of:
  - `"none"`: the core polygon is used as provided or computed.
  - `"rect"`: the polygon is replaced by its minimum enclosing rectangle.
  - `"aarect"`: the polygon is replaced by an axis-aligned enclosing rectangle.

`fill_around`
: Logical. If `TRUE`, trees are added outside the core polygon until the basal area per hectare of the full stand matches that of the core inventory.

`verbose`
: Logical. If `TRUE` (default), messages and warnings are printed during processing. If `FALSE`, output is silent.

Details

The function supports sloping terrain and coordinate system rotation, and returns a fully prepared stand ready for use in the ray-tracing pipeline (see run_sl).

The returned `trees` data.frame conforms to the inventory format checked by check_inventory, with the following controlled modifications:

- Tree vertical position `z` is computed from terrain slope and aspect.
- Crown maximum radius height `hmax_m` is computed when fixed by crown geometry conventions:
  - `"P"` and `"4P"`: `hmax_m = hbase_m`
  - `"E"`: `hmax_m = hbase_m + 0.5 * (h_m - hbase_m)`

  For crown types `"2E"` and `"8E"`, `hmax_m` must be provided by the user.
- For symmetric crown types (`"P"`, `"E"`, `"2E"`), a unique crown radius is computed as the mean of the provided radii (`rn_m`, `rs_m`, `re_m`, `rw_m`; `NA` values ignored).
- If missing, column `crown_openness` is added and filled with `NA` (only needed when `turbid_medium = FALSE`).
- If missing, column `crown_lad` is added and filled with `NA` (only needed when `turbid_medium = TRUE`, the default).

All input data.frames (`trees_inv`, `sensors`, and `core_polygon_df`) are automatically checked for coordinate type:

- If `x` and `y` columns exist, they are assumed to be planar.
- If `lon` and `lat` columns exist, they are converted into planar UTM coordinates automatically using `create_xy_from_lonlat()`.
- The UTM projection (EPSG) is determined from the mean coordinates of `trees_inv`. All inputs must share the same EPSG; otherwise, the function stops with an error. If conversion occurred, the EPSG is stored in the output.

The function ensures that all trees fall within the core inventory polygon, applying small buffers if necessary to handle numerical precision issues. Invalid polygons are automatically repaired when possible.

When `fill_around = TRUE`, trees are randomly sampled from the original inventory and positioned outside the core polygon until the target basal area per hectare is reached for the full stand.

Value

A named list with the following elements:

`trees` Data.frame of the final tree inventory, including added trees if `fill_around = TRUE`. The structure matches the inventory format validated by check_inventory, with additional derived variables required for ray tracing. A logical column `added_to_fill` indicates whether each tree originates from the initial inventory or was added to fill around the inventory zone.

`core_polygon` List describing the inventory zone:

- `df`: data.frame of polygon vertices
- `sf`: corresponding `sf` POLYGON
- `modify_polygon`: applied polygon modification

`transform` List of transformation and filling information, including core area, target and final basal area, number of added trees, and applied spatial transformations.

`geometry` List describing stand geometry and terrain parameters (cell size, number of cells, slope, aspect, and orientation).

Examples

```
data_prenovel <- SamsaRaLight::data_prenovel
trees_inv <- data_prenovel$trees

stand <- create_sl_stand(
  trees_inv = trees_inv,
  cell_size = 10,
  latitude = 46.52666,
  slope = 6,
  aspect = 144,
  north2x = 54
)
```

`create_xy_from_lonlat` *Create planar (x, y) coordinates from longitude / latitude*

Description

This function converts geographic coordinates (`lon`, `lat`) into planar coordinates (`x`, `y`) using an automatically selected UTM projection.

Usage

```
create_xy_from_lonlat(df)
```

Arguments

`df`	A data.frame containing geographic coordinates (`lon` and `lat`).

Details

The input data.frame must contain geographic coordinates (`lon` and `lat`). Planar coordinates (`x`, `y`) are automatically computed using the UTM zone inferred from the mean longitude and hemisphere inferred from the mean latitude.

Value

A list with the following elements:

`df` The input data.frame with added `x` and `y` columns (meters, UTM).

`epsg` EPSG code of the UTM projection used.

Examples

```
# Example data with longitude / latitude (WGS84)
df <- data.frame(
  lon = c(4.35, 4.36),
  lat = c(50.85, 50.86)
)

# Convert to planar coordinates (UTM)
res <- create_xy_from_lonlat(df)

# Access results
res$df   # data.frame with x, y columns
res$epsg # EPSG code used
```

`get_bottom_azimut` *Compute bottom azimut*

Description

Compute bottom azimut

Usage

```
get_bottom_azimut(aspect, north2x)
```

Arguments

`aspect`	double - Angle of slope bottom on the compass from the North, clockwise rotation (in degrees) northern aspect : 0, eastern aspect : 90, southern aspect : 180, western aspect : 270
`north2x`	double - Angle from North to x axis clockwise. (in degrees) Default correspond to a Y axis oriented toward the North.

Value

a bottom azimuth numeric value

`get_monthly_radiations` *Create the SamsaraLight*

Description

Fetch monthly radiation data from PVGIS website (by API) between start and end year (limit years are from 2005 to 2020). Fetched variables are Hrad = horizontal plane irradiation and DGratio = ratio of diffuse to global radiation (in horizontal plane).

! YOU NEED AN INTERNET CONNECTION TO ACCESS THE DATA BY API !

Usage

```
get_monthly_radiations(latitude, longitude, start_year = 2005, end_year = 2020)
```

Arguments

`latitude`	latitude of the plot
`longitude`	longitude of the plot
`start_year`	positive integer between 2005 and 2020 - start year on which to fetch monthly data
`end_year`	positive integer between 2005 and 2020 - end year on which to fetch monthly data

Value

Monthly horizontal radiation (Hrad) and diffuse to global ratio (DGratio) averaged between start_year and end_year

Source

https://joint-research-centre.ec.europa.eu/pvgis-photovoltaic-geographical-information-system_en

Examples

```
# Example: fetch monthly radiation somewhere in Belgium
rad <- get_monthly_radiations(
  latitude = 50.85,
  longitude = 4.35,
  start_year = 2010,
  end_year = 2015
)

head(rad)
```

get_z *Compute z coordinate of a point (x,y).*

Description

Compute z coordinate of a point (x,y).

Usage

```
get_z(x, y, slope_rad, bottom_azimut_rad)
```

Arguments

x	X-coordinate of the point
y	Y-coordinate of the point
slope_rad	Slope of the stand (in radians)
bottom_azimut_rad	Azimuth of the vector orthogonal to the ground in the x,y system (in radians)

Value

a z coordinate numeric value

plot.sl_output *Plot a SamsaRaLight output*

Description

Visualize ground light and tree energy metrics for a `sl_output` object.

Usage

```
## S3 method for class 'sl_output'
plot(
  x,
  ...,
  what_trees = c("compet", "compet_rays", "intercepted", "potential"),
  what_cells = c("relative", "absolute"),
  show_trees = TRUE,
  direct_energy = NULL
)
```

Arguments

x	An object of class sl_output, returned by run_sl().
...	Additional arguments passed to lower-level plotting functions.
what_trees	Character; which tree metric to plot. Choices are: **"compet"** Light competition index (LCI), reversed viridis scale. **"compet_rays"** Ray competition index (RCI), reversed viridis scale. **"intercepted"** Intercepted energy (MJ). **"potential"** Potential intercepted energy (MJ). Default is "compet".
what_cells	Character; which cell (ground) metric to plot. Choices are: **"relative"** Proportion of above canopy light (PACL)). **"absolute"** Energy on the ground (MJ/m2). Default is "relative".
show_trees	Logical; whether to display trees on top of the ground light map. Default is TRUE.
direct_energy	Logical or NULL. If NULL (default), total radiation outputs are plotted (direct + diffuse). If TRUE, only direct radiation components are plotted. If FALSE, only diffuse radiation components are plotted. This option requires detailed_output = TRUE when running the simulation

Value

A ggplot object.

---

plot.sl_stand *Plot a SamsaRaLight virtual stand*

Description

This function plots a virtual forest stand (sl_stand) produced by SamsaRaLight. It can display a top-down view with tree crowns or a side/top view with cells and trees.

Usage

```
## S3 method for class 'sl_stand'
plot(x, ..., top_down = FALSE, only_inv = FALSE, add_sensors = TRUE)
```

Arguments

x	An object of class sl_stand.
...	Additional arguments passed to lower-level plotting functions.
top_down	Logical, if TRUE, creates a top-down view with multiple directions (south, north, west, east).
only_inv	Logical, if TRUE, plot only trees from the initial inventory (i.e. not trees added to fill around the core polygon)
add_sensors	Logical; if TRUE (default), sensors are drawn on the plot. In top-down mode, sensors are shown as segment from ground to their height; in map view, sensors are drawn as squares on the ground.

Details

For the sake of the representation in top-down plot, z are offset such as minimum altitude tree is at Y-axis height = 0

The plot function also generates a compass indicating:

- Plot orientation (north2x, north in red)
- Terrain aspect (downslope direction)
- Slope degrees (annotation only)

Value

A ggplot object representing the stand.

---

plot_inventory *Plot a from-above view of a tree inventory*

Description

Visualizes a forest inventory as ellipses representing tree crowns.

Usage

```
plot_inventory(trees_inv, core_polygon_df = NULL, show_id = TRUE)
```

Arguments

trees_inv	A data.frame of trees that passed check_inventory.
core_polygon_df	Optional data.frame defining the core inventory polygon. Must contain columns x and y.
show_id	Logical; if TRUE (default), displays tree identifiers at crown centers.

Details

Because the north2x variable is unknown, trees are plotted as circles by considering the mean radius on the four cardinals.

Value

A ggplot object displaying the trees in a from-above view.

Examples

```
data_prenovel <- SamsaRaLight::data_prenovel
trees_inv <- data_prenovel$trees

plot_inventory(trees_inv)
```

---

print.sl_output *Print a* sl_output *object*

Description

Prints a concise one-line summary of a SamsaRaLight simulation result, reporting the number of cells, trees, and sensors, and whether detailed outputs are available.

Usage

```
## S3 method for class 'sl_output'
print(x, ...)
```

Arguments

x	An object of class sl_output.
...	Further arguments passed to or from other methods (ignored).

Value

The object x, invisibly.

print.sl_stand *Print a* sl_stand *object*

Description

Prints a compact, human-readable, one-line description of a sl_stand object, summarizing the stand size, number of trees, number of sensors, and grid geometry.

Usage

```
## S3 method for class 'sl_stand'
print(x, ...)
```

Arguments

x	An object of class sl_stand.
...	Unused. Included for S3 method compatibility.

Details

This method is automatically called when a sl_stand object is printed in the console.

Value

The input object x, invisibly.

run_sl *Run SamsaRaLight radiative balance*

Description

Computes light interception and radiative balance for a forest stand using the SamsaRaLight ray-tracing engine.

Usage

```
run_sl(
  sl_stand,
  monthly_radiations,
  sensors_only = FALSE,
  detailed_output = FALSE,
  parallel_mode = FALSE,
  n_threads = NULL,
  verbose = TRUE
)
```

Arguments

`sl_stand`	An object of class `"sl_stand"` describing the forest stand, typically created with create_sl_stand. It contains tree, sensor, terrain, and grid geometry information.
`monthly_radiations`	A data frame containing monthly horizontal radiation (`Hrad`, in MJ m^{-2}) and diffuse-to-global radiation ratio (`DGratio`), typically obtained with get_monthly_radiations. The input is validated internally using check_monthly_radiations.
`sensors_only`	Logical. If `TRUE`, compute light interception only for sensors. This can substantially reduce computation time when only sensor-level results are required. Defaults to `FALSE`.
`detailed_output`	Logical. If `TRUE`, retain detailed ray, energy, and interception information, including the monthly ray discretization and interception matrices. If `FALSE`, only the main light-interception metrics are returned. Defaults to `FALSE`.
`parallel_mode`	Logical. If `TRUE`, ray–target computations are parallelised using OpenMP. If `FALSE`, the model runs in single-thread mode. BLAS is automatically forced to single-thread mode during the simulation to avoid competition between BLAS and OpenMP. Parallel execution can substantially reduce computation time for large stands or fine ray discretization. Defaults to `FALSE`.
`n_threads`	Integer or `NULL`. Number of CPU threads to use when `parallel_mode = TRUE`. If `NULL`, OpenMP automatically selects the number of available threads. If supplied, must be a positive integer. Defaults to `NULL`.
`verbose`	Logical. If `TRUE`, print informative messages during the simulation, including OpenMP status. Defaults to `TRUE`.

Details

This function is the standard user interface to SamsaRaLight. Advanced ray-tracing, sky-discretization, and interception parameters are internally set to standard values and do not need to be provided.

Internally, `run_sl()` calls run_sl_advanced using standard ray-tracing, sky-discretization, and interception parameters:

- torus borders are enabled;
- crowns are represented as turbid media;
- the leaf extinction coefficient is `0.5`;
- the clumping factor is `1`;
- trunk interception is enabled;
- the minimum ray altitude angle is `10` degrees;
- direct rays use a starting offset of `0` degrees and an angular step of `5` degrees;
- diffuse rays use an angular step of `15` degrees;
- the Standard Overcast Sky model is used;
- the full year (days 1–365) is simulated.

These defaults are intended for standard SamsaRaLight simulations. Users requiring control over these parameters can use run_sl_advanced directly.

When `parallel_mode = TRUE`, SamsaRaLight uses OpenMP to parallelise ray–target computations. BLAS is configured to use a single thread during the simulation to avoid competing parallel execution.

Value

An object of class `"sl_output"` (a list) containing:

- `output`: A list containing the simulation results:
  - `light`: A list containing light-interception results for trees, ground cells, and sensors.
  - `monthly_rays`: The monthly ray discretization and associated radiation energies. Returned only when `detailed_output = TRUE`.
  - `interceptions`: Detailed interception matrices. Returned only when `detailed_output = TRUE`.
- `params`: A list containing the simulation parameters, including the simulated period, sky model, ray-discretization parameters, and interception-model parameters.
- `input`: A list containing the original `sl_stand` and `monthly_radiations` objects.

When `detailed_output = FALSE`, the main output tables contain:

- `light$sensors`: `id_sensor`, `e`, `pacl`, and `punobs`;
- `light$cells`: `id_cell`, `e`, `pacl`, and `punobs`;
- `light$trees`: `id_tree`, `epot`, `e`, `lci`, `eunobs`, and `rci`.

See Also

create_sl_stand, check_inventory, check_sensors, get_monthly_radiations, check_monthly_radiations, run_sl_advanced

Examples

```
data_prenovel <- SamsaRaLight::data_prenovel

stand <- create_sl_stand(
  trees_inv = data_prenovel$trees,
  sensors = data_prenovel$sensors,
  cell_size = 10,
  latitude = data_prenovel$info$latitude,
  slope = data_prenovel$info$slope,
  aspect = data_prenovel$info$aspect,
  north2x = data_prenovel$info$north2x
)

out <- run_sl(
  sl_stand = stand,
  monthly_radiations = data_prenovel$radiations
)
```

```
str(out)
```

`run_sl_advanced` *Compute advanced SamsaRaLight radiative balance*

Description

Runs the full light interception and radiative balance simulation for a virtual forest stand with advanced ray-tracing and interception parameters. This function provides full control over ray discretization, sky model, crown interception model, trunk interception, and parallel computation.

Usage

```
run_sl_advanced(
  sl_stand,
  monthly_radiations,
  sensors_only = FALSE,
  use_torus = TRUE,
  turbid_medium = TRUE,
  extinction_coef = 0.5,
  clumping_factor = 1,
  trunk_interception = TRUE,
  height_anglemin = 10,
  direct_startoffset = 0,
  direct_anglestep = 5,
  diffuse_anglestep = 15,
  soc = TRUE,
  start_day = 1,
  end_day = 365,
  detailed_output = FALSE,
  parallel_mode = FALSE,
  n_threads = NULL,
  verbose = TRUE
)
```

Arguments

`sl_stand`	An object of class `"sl_stand"` representing the virtual forest stand. Each row of `sl_stand$trees` represents a tree, with required and optional columns describing crown geometry, tree height, crown radius, crown openness, leaf area density, and related attributes. See validate_sl_stand.
`monthly_radiations`	A data frame containing monthly horizontal radiation (`Hrad`) and diffuse-to-global radiation ratios (`DGratio`), typically computed with get_monthly_radiations.
`sensors_only`	Logical. If `TRUE`, compute light interception only for sensors. Defaults to `FALSE`.

use_torus Logical. If TRUE, use a torus system to handle stand borders. Defaults to TRUE.

turbid_medium Logical. If TRUE, crowns are represented as turbid media using crown_lad. If FALSE, crowns are represented as porous envelopes using crown_openess. Defaults to TRUE.

extinction_coef
Numeric scalar. Leaf extinction coefficient controlling the attenuation of radiation by foliage. It represents the effective light attenuation per unit leaf area and is related to average leaf orientation. Higher values increase light interception. Defaults to 0.5.

clumping_factor
Numeric scalar controlling the aggregation of foliage within the crown volume. A value of 1 corresponds to homogeneous foliage distribution, values below 1 indicate clumped foliage, and values above 1 indicate more regular spacing. This parameter affects light interception in the turbid-medium model. Defaults to 1.

trunk_interception
Logical. If TRUE, account for interception by tree trunks. Defaults to TRUE.

height_anglemin
Numeric. Minimum altitude angle of rays, in degrees. Defaults to 10.

direct_startoffset
Numeric. Starting angle of the first direct ray, in degrees. Defaults to 0.

direct_anglestep
Numeric. Angular step between direct rays, in degrees. Defaults to 5.

diffuse_anglestep
Numeric. Angular step between diffuse rays, in degrees. Defaults to 15.

soc Logical. If TRUE, use the Standard Overcast Sky model; if FALSE, use the Uniform Overcast Sky model. Defaults to TRUE.

start_day Numeric. First day of the simulated vegetative period, between 1 and 365. Defaults to 1.

end_day Numeric. Last day of the simulated vegetative period, between 1 and 365. Must be greater than or equal to start_day. Defaults to 365.

detailed_output
Logical. If TRUE, retain detailed ray, energy, and interception information in the output. If FALSE, only the main light-interception metrics are retained. Defaults to FALSE.

parallel_mode Logical. If TRUE, ray–target computations are parallelised using OpenMP. If FALSE, the model runs in single-thread mode. Defaults to FALSE.

n_threads Integer or NULL. Number of CPU threads to use when parallel_mode = TRUE. If NULL, OpenMP automatically selects the number of available threads. If supplied, must be a positive integer. Defaults to NULL.

verbose Logical. If TRUE, print informative messages during the simulation, including OpenMP status. Defaults to TRUE.

Details

For typical use, see run_sl, which provides standard values for the ray-discretization parameters.

This advanced function exposes all ray-tracing parameters used by `create_sl_rays` and the interception model used by the underlying C++ simulation. It is intended for users who need fine control over ray discretization, sky conditions, crown representation, or computational settings.

Before running the simulation, the function validates the stand, monthly radiation data, interception-model configuration, logical and numeric parameters, simulation period, and number of OpenMP threads.

BLAS and OpenMP thread counts are configured to avoid competing parallel execution. When `parallel_mode = TRUE`, the number of OpenMP threads can be controlled with `n_threads`.

For most users, run_sl is recommended because it provides standard ray-discretization parameters.

Value

An object of class `"sl_output"` (a list) containing:

- `output`: A list containing the simulation results:
  - `light`: Light-interception results for trees, cells, and sensors. When `detailed_output = FALSE`, these contain the main output metrics only.
  - `monthly_rays`: The generated monthly ray discretization and associated radiation energies. Returned only when `detailed_output = TRUE`.
  - `interceptions`: Detailed tree/cell interception matrices. Returned only when `detailed_output = TRUE`.
- `params`: A list containing the simulation parameters, including the simulated period, sky model, ray discretization, crown interception model, and interception parameters.
- `input`: A list containing the original `sl_stand` and `monthly_radiations` objects.

When `detailed_output = FALSE`, the main output tables contain:

- `light$sensors`: `id_sensor`, `e`, `pacl`, and `punobs`;
- `light$cells`: `id_cell`, `e`, `pacl`, and `punobs`;
- `light$trees`: `id_tree`, `epot`, `e`, `lci`, `eunobs`, and `rci`.

`SamsaRaLight_data` *Example forest inventory datasets for SamsaRaLight*

Description

These datasets provide example forest inventories used for light interception simulations with the SamsaRaLight package. Each dataset is a named list with 5 elements: `trees`, `sensors`, `core_polygon`, `radiations`, and `info`.

Usage

```
data(data_prenovel)
data(data_IRRES1)
data(data_bechefa)
data(data_cloture20)

data_prenovel

data_IRRES1

data_bechefa

data_cloture20
```

Format

A named list with 5 elements:

trees A `data.frame` with tree-level data:

- **id_tree** Unique id of the tree (integer).
- **species** Latin species name (character).
- **x, y** Coordinates of the base of the tree in meters (numeric).
- **dbh_cm** Diameter at breast height in cm (numeric).
- **crown_type** Crown shape type, e.g. "P", "E", or complex types like "8E" (character).
- **h_m** Height of the tree trunk in meters (numeric).
- **hbase_m** Height of crown base in meters (numeric).
- **hmax_m** Height at which crown radius is maximum, NA if not used (numeric).
- **rn_m, rs_m, re_m, rw_m** Crown radii in meters (numeric).
- **crown_lad** Leaf area density of the crown (m2/m3) (numeric).

sensors A `data.frame` with sensor data:

- **id_sensor** Unique sensor ID (integer).
- **x, y** Coordinates of the sensor in meters (numeric).
- **h_m** Height of the sensor in meters (numeric).
- **pacl, pacl_direct, pacl_diffuse** Proportion of above-canopy light measured at the sensor (numeric).

May be `NULL` if no sensors are present.

core_polygon A `data.frame` with vertices of the inventory polygon:

- **x, y** Coordinates of polygon vertices in meters (numeric).

radiations A `data.frame` of monthly radiation:

- **month** Month number (integer).
- **Hrad** Monthly radiation in MJ (numeric).
- **DGratio** Diffuse-to-global radiation ratio (numeric).

info A named list with site information:

latitude, longitude Coordinates of the site in decimal degrees (numeric).

slope Mean slope in degrees (numeric).

aspect Aspect in degrees from north (numeric).

north2x Angle from north to x-axis clockwise in degrees (numeric).

An object of class `list` of length 5.

An object of class `list` of length 5.

An object of class `list` of length 5.

An object of class `list` of length 5.

Source

Courbaud Benoit (INRAe LESSEM Grenoble)

Gauthier Ligot (Gembloux Agro-Bio Tech)

Gauthier Ligot (Gembloux Agro-Bio Tech)

Gauthier Ligot (Gembloux Agro-Bio Tech)

`summary.sl_output` *Summary of a SamsaRaLight simulation*

Description

Summary of a SamsaRaLight simulation

Usage

```
## S3 method for class 'sl_output'
summary(object, ...)
```

Arguments

`object`	an object of class `sl_output`
`...`	unused

Value

A list containing summary statistics of the simulation output, including tree-level intercepted light and stand-level incident light.

summary.sl_stand *Summary of a SamsaRaLight stand*

Description

Summary of a SamsaRaLight stand

Usage

```
## S3 method for class 'sl_stand'
summary(object, ...)
```

Arguments

object	A sl_stand object
...	Unused

Value

Invisibly returns a list of summary tables

Index

∗ **datasets**
    SamsaRaLight_data, 22
∗ **internals**
    get_bottom_azimut, 11
    get_z, 13

check_coordinates, 2
check_inventory, 3, *6*, *8–10*, *15*, *19*
check_monthly_radiations, 5, *18, 19*
check_polygon, 6
check_sensors, *6*, 7, *8*, *19*
create_sl_rays, *22*
create_sl_stand, 8, *18, 19*
create_xy_from_lonlat, 10

data_bechefa *(*SamsaRaLight_data*)*, 22
data_cloture20 *(*SamsaRaLight_data*)*, 22
data_IRRES1 *(*SamsaRaLight_data*)*, 22
data_prenovel *(*SamsaRaLight_data*)*, 22

get_bottom_azimut, 11
get_monthly_radiations, *5*, 12, *18–20*
get_z, 13

plot.sl_output, 13
plot.sl_stand, 14
plot_inventory, 15
print.sl_output, 16
print.sl_stand, 17

run_sl, *9*, 17, *22*
run_sl_advanced, *18, 19*, 20

SamsaRaLight_data, 22
summary.sl_output, 24
summary.sl_stand, 25

validate_sl_stand, *20*