Package: framecpp (via r-universe)

November 13, 2026

Type Package

Title Forest Flammability via FRaME Mechanics (C++/Rcpp)

Version 0.1.0.9005

Description A C++/Rcpp reimplementation of the FRaME forest flammability model (Zylstra et al.), generalised from discrete strata to continuous plant size distributions, with the classic discrete behaviour retained as a special case. Inputs may be supplied as full parameters, resolved from AusTraits / the AusTraits Plant Dictionary, or driven from field data or the 'plant' demographic model. Part of the AusTraits plant-family.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

LinkingTo Rcpp

Imports Rcpp

Suggests testthat (>= 3.0.0), knitr, rmarkdown, dplyr, tidyr, tibble

VignetteBuilder knitr

SystemRequirements C++17

Config/roxygen2/version 8.0.0

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

Date/Publication 2026-07-20 23:02:03 UTC

RemoteUrl https://github.com/traitecoevo/framecpp

RemoteRef HEAD

RemoteSha 158b84e62bab04ee992a690e704f573eec61402d

Contents

crown_polygon 2
crown_ray_intersection 3
fire_scenario 3

flame_angle 4
framecpp_version 5
parse_fire_result 5
run_fire_model 6
scenario_from_community 7
scenario_from_params 8
scenario_from_plant 9
scenario_from_population 10
species_derived 11
surface_fire 12

Index **14**

`crown_polygon` *Crown polygon from height/width parameters*

Description

Builds a plant's crown as the convex polygon FRaME uses (a hexagon defined by the four crown heights and half-width), the shape the ignition-path geometry intersects rays against.

Usage

```
crown_polygon(hc, he, ht, hp, w)
```

Arguments

`hc`, `he`, `ht`, `hp`	Crown heights (m): base, height of the lower edge of maximum width, height of the upper edge of maximum width, and top.
`w`	Crown width (m).

Value

A list with the polygon `centroid`, `area` (m^2), `volume` (m^3, solid of revolution), `lowerLeft`/`lowerRight` base corners, and the `vertices` matrix (columns x, y in metres).

Examples

```
crown_polygon(hc = 0, he = 0.05, ht = 0.5, hp = 0.6, w = 0.5)$area
```

crown_ray_intersection *Intersection of a ray with a crown polygon*

Description

Returns the chord where a ray (used, e.g., as a flame or heating ray in the ignition-path model) passes through a crown polygon.

Usage

```
crown_ray_intersection(hc, he, ht, hp, w, origin_x, origin_y, angle)
```

Arguments

hc, he, ht, hp	Crown heights (m): base, height of the lower edge of maximum width, height of the upper edge of maximum width, and top.
w	Crown width (m).
origin_x, origin_y	Ray origin (m).
angle	Ray direction (radians).

Value

A numeric vector `c(x0, y0, x1, y1)` giving the entry and exit points of the chord; a degenerate (zero-length) segment if the ray misses.

Examples

```
crown_ray_intersection(0, 0.05, 0.5, 0.6, 0.5,
                       origin_x = 0, origin_y = -1, angle = pi/2)
```

fire_scenario *Construct a validated fire-model scenario*

Description

`fire_scenario()` is the one input the compiled single-site fire model runs. It wraps the model-native list the engine consumes in a typed, validated constructor so every builder (`scenario_from_params()`, `scenario_from_community()`) produces the same first-class object.

Usage

```
fire_scenario(site, strata, overlaps = NULL)
```

Arguments

site	A named list (or numeric vector) of the eight site scalars: slope, dfmc, fuelLoad, meanFuelDiameter, meanFinenessLeaves, temp, wind, fireLineLength.
strata	A list with one element per stratum, each a list of level (integer 1-4: near surface, elevated, midstorey, canopy), plantSep (numeric, m) and species (a non-empty list). Each species is a list of params (a length-16 numeric vector), leafForm ("flat" or "round") and weighting (numeric composition).
overlaps	An optional list of stratum overlaps, each a list of lower, upper (integer levels) and type (0 overlapped, 1 not overlapped, 2 automatic). NULL gives no overlaps.

Details

All values are model-native SI: slope in radians, wind in m/s, fuelLoad in kg/m2, lengths in m, moisture as a proportion. The builders are responsible for any unit conversion before calling this constructor.

Value

An object of class fire_scenario: the flat named list the engine reads, with the site scalars at the top level plus strata and overlaps.

See Also

scenario_from_params(), scenario_from_community(), run_fire_model()

---

flame_angle *Flame angle from wind and slope*

Description

The angle (from horizontal) a flame of a given length is tilted to by wind and slope, combining the wind- and slope-effect contributions.

Usage

```
flame_angle(flame_length, wind_speed, slope, fireline_length)
```

Arguments

flame_length	Flame length (m).
wind_speed	Wind speed (m/s).
slope	Ground slope (radians).
fireline_length	Fire-line length (m).

Value

The flame angle (radians).

Examples

```
flame_angle(flame_length = 2, wind_speed = 5, slope = 0, fireline_length = 100)
```

---

framecpp_version *framecpp package version*

Description

framecpp package version

Usage

```
framecpp_version()
```

Value

A version string for the compiled framecpp core.

---

parse_fire_result *Parse the compiled fire-model output into a* fire_result

---

Description

Turns the compiled single-site model's output into a structured result. The full path (Phase 3A) accepts the nested list `framecpp_fireResult()` produces and returns a first-class `fire_result` object exposing the complete run — the second half of the engine interface contract (`dev/package-boundary.md`). A legacy path still accepts the flat numeric vector `framecpp_runFireModel()` produces and returns the earlier subset list.

Usage

```
parse_fire_result(x)
```

Arguments

x	Either the nested list returned by the compiled `framecpp_fireResult()` (full object) or the flat numeric vector returned by `framecpp_runFireModel()` (legacy subset).

Details

A `fire_result` carries all seven `frame` result tables' worth as tidy data frames in `$tables` (`Sites`, `Runs`, `Strata`, `FlameSummaries`, `SurfaceResults`, `IgnitionPaths`, `ROS`), plus a nested per-run convenience view (`$run1`, `$run2`) and the scalars `$canopyROS` / `$hasSecondRun`. The two runs are the first pass (with the canopy wind effect) and the crown-fire second pass (without it); when there is no crown fire the second run is empty.

Species in `$tables$IgnitionPaths` are identified by the engine's own `speciesId` (build order, stratum-major) and a `speciesRank` within each (`level`, `pathType`) group, not by name: names are a data-source concern that lives on the consumer side of the boundary (`frame` resolves ids to names).

Value

For the full path, a `fire_result` object (see Details). For the legacy numeric path, a plain list of `canopyROS`, `hasSecondRun` and two runs (`run1`, `run2`), each with `surfaceROS`, a named `flameSummaries` list and a named `ros` vector.

See Also

`run_fire_model()`, `fire_scenario()`

`run_fire_model` *Run the whole single-site fire model*

Description

The classic-mode entry point: chains a scenario through the compiled single-site model and `parse_fire_result()` into a full `fire_result` object.

Usage

```
run_fire_model(x)
```

Arguments

x	Either a fire_scenario (run directly) or a FRaME `paramBuilder` parameter table (converted first via `scenario_from_params()`).

Value

A `fire_result`: the full run, with all seven `frame` result tables in `$tables` and a nested per-run view.

Examples

```
ex <- readRDS(system.file("extdata", "example.rds", package = "framecpp"))
res <- run_fire_model(ex$mild$params)
res$run1$ros
res$tables$FlameSummaries
```

scenario_from_community *Build a fire-model scenario from a described community (Tier A)*

Description

The native replacement for `frame::paramBuilder()`: turn a classic, layered community description (tidy data frames) into a validated fire_scenario, with no `frame` dependency and no parameter-table round trip. Strata are *given* (one row per stratum); this is the direct analogue of the discrete-strata FRaME workflow.

Usage

```
scenario_from_community(
  site,
  strata,
  species,
  traits,
  overlaps = NULL,
  units = c("si", "frame")
)
```

Arguments

`site`	A one-row data frame of site scalars: `slope`, `wind`, `temp`, `dead_fuel_moisture`, `surface_fuel_load`, `fireline_length`, and the optional `mean_fuel_diameter` (default 0.005) and `mean_fineness_leaves` (default 0.00025).
`strata`	A data frame with one row per stratum: `level` (one of `"near surface"`, `"elevated"`, `"midstorey"`, `"canopy"`) and `plant_separation` (m).
`species`	A data frame with one row per stratum x species: `stratum` (matching a `strata$level`), `name`, `composition`, crown `base`, `he`, `ht`, `top`, `width`, `live_moisture`, `openness`, `clump`.
`traits`	A data frame with one row per species (joined by `name`): `leaf_form`, `leaf_thickness`, `leaf_width`, `leaf_length`, `leaf_separation`, `stem_order`, `ignition_temp`, `prop_dead`.
`overlaps`	Optional data frame of pairwise stratum overlaps: `lower`, `upper` (level names) and `type` (`"t"`/`"f"`/`"automatic"`). Carried for fidelity; framecpp's vertical association is height-only and ignores it.
`units`	Input unit system: `"si"` (default; wind m/s, slope rad, `surface_fuel_load` kg/m2) or `"frame"` (wind km/h, slope deg, `surface_fuel_load` t/ha).

Details

The builder reproduces the derivations in FRaME's `Building.R` (`siteBuilder`/`speciesBuilder`/`paramBuilder`) bit-for-bit: `he`←`base` and `ht`←`top` when absent, `live_moisture`→1 when absent, `clumpDiameter = min(top - base, width) * clump`, `clumpSeparation = openness * clumpDiameter`, and per-stratum `plantSep = max(declared separation, composition-weighted mean crown width)`.

Value

A fire_scenario.

See Also

`fire_scenario()`, `scenario_from_params()`, `run_fire_model()`

`scenario_from_params` *Build a fire-model scenario from a FRaME* `paramBuilder` *table*

Description

Converts the long parameter table that `frame::paramBuilder()` produces (the same table `CSVRunner` consumes) into the nested list the compiled model expects. Applies the unit conversions FRaME's `Sites` table records: wind km/h -> m/s (`/3.6`), fuel load t/ha -> kg/m2 (`/10`), slope deg -> rad.

Usage

```
scenario_from_params(params)
```

Arguments

`params`	A data frame with columns `stratum`, `species`, `param`, `value`, `units`. Site-level rows have `stratum` and `species` `NA`; stratum-level rows (`levelName`, `plantSeparation`) have `species` `NA`.

Value

A fire_scenario: the site scalars (`slope`, `dfmc`, `fuelLoad`, `meanFuelDiameter`, `meanFinenessLeaves`, `temp`, `wind`, `fireLineLength`), a `strata` list (each with `level`, `plantSep`, and a `species` list of `params`/`leafForm`/`weighting`), and an `overlaps` list.

See Also

`fire_scenario()`, `run_fire_model()`, `parse_fire_result()`

scenario_from_plant *Build a fire-model scenario from* plant *demographic output (Tier C)*

Description

The direct plant -> framecpp front door: a population of plant cohorts meets the engine at fire_scenario with no frame bridge and no k-means. This is a thin adapter over scenario_from_population() (Tier B) — it maps the cohort table onto the population contract and delegates, so strata are derived by the same deterministic geometric height-gap segmentation and a cleanly layered set of cohorts reduces to scenario_from_community() bit-for-bit (the plan §3.2 equivalence contract). It introduces no new physics.

Usage

```
scenario_from_plant(
  site,
  cohorts,
  strata,
  traits,
  overlaps = NULL,
  units = c("si", "frame"),
  packing = list(live_moisture = 1, openness = 3, clump = 0.1)
)
```

Arguments

site	As in scenario_from_community().
cohorts	A data frame with one row per plant cohort: name, crown base, top, width, and abundance (relative weight / density, mapped to composition). Optional he, ht (crown edge/top heights; default to base/top) and optional per-cohort live_moisture, openness, clump.
strata	A data frame of the levels present, one row each: level (name) and plant_separation (m); the count must equal the number of vertical modes the cohorts resolve into (see scenario_from_population()).
traits	As in scenario_from_community().
overlaps	As in scenario_from_community().
units	As in scenario_from_community().
packing	A named list of fallback packing/moisture values — live_moisture, openness, clump — used for any cohort that does not supply them as a column.

Details

Cohorts carry crown geometry and relative abundance; the packing / moisture terms plant does not itself produce (live_moisture, openness, clump) are supplied per cohort (as columns) or, failing that, from packing. The packing defaults follow FRaME's buildTraitsP conventions (openness = gap:clump 3, clump = clump:canopy 0.1) and should be reviewed for the community at hand — framecpp does not infer them.

Value

A fire_scenario.

See Also

scenario_from_population(), scenario_from_community(), fire_scenario(), run_fire_model()

---

scenario_from_population *Build a fire-model scenario from a plant population (Tier B)*

Description

The continuous / non-layered front door: input is a *population* of plant elements (individuals or cohorts) with continuous crown geometry and leaf traits, and **no stratum labels**. Strata are derived internally from the vertical foliage profile, then the same fire_scenario is built on top of the shared species/site assembly used by scenario_from_community().

Usage

```
scenario_from_population(
  site,
  plants,
  strata,
  traits,
  overlaps = NULL,
  units = c("si", "frame")
)
```

Arguments

site	As in scenario_from_community().
plants	A data frame with one row per plant element (no stratum): name, crown base, he, ht, top, width, live_moisture, openness, clump, and composition (relative abundance / weight).
strata	A data frame of the levels present, one row each: level (name) and plant_separation (m). The count must equal the number of vertical modes the population resolves into.
traits	As in scenario_from_community().
overlaps	As in scenario_from_community().
units	As in scenario_from_community().

Details

Stratification is a deterministic geometric derivation (no k-means, no RNG): a stratum is a maximal set of plants whose crown vertical extents (`base`..`top`) form a connected band; a height gap with no foliage — a zero of the vertical foliage profile — separates strata. The resulting modes are ordered by height and mapped to the supplied `strata` levels by rank. On a cleanly layered population this recovers the classic strata, so the result reduces to `scenario_from_community()` bit-for-bit (the plan §3.2 equivalence contract). A profile whose number of modes does not match the supplied levels is *non-layered*: that is a research case (plan §6 P3) and is rejected here rather than resolved arbitrarily.

Plant elements sharing a derived stratum and species name are aggregated — composition sums, crown/packing values are composition-weighted means — so a population with one element per stratum x species passes through unchanged.

Stratum-level properties (`plant_separation`, `overlaps`) are *supplied* per level, exactly as in Tier A; this builder introduces no new physics, only a geometric replacement for the external stratification step.

Value

A fire_scenario.

See Also

`scenario_from_community()`, `fire_scenario()`, `run_fire_model()`

---

species_derived *Derived species / leaf-trait quantities*

Description

The species-level quantities FRaME derives from raw crown geometry and leaf traits: live proportion, leaf area, leaf moisture, ignitability coefficient, leaves per clump and leaf area index.

Usage

```
species_derived(
  hc,
  he,
  ht,
  hp,
  w,
  live_moisture,
  dead_moisture,
  prop_dead,
  ignition_temp,
  leaf_form,
  leaf_thickness,
```

```
  leaf_width,
  leaf_length,
  leaf_separation,
  stem_order,
  clump_diameter,
  clump_separation
)
```

Arguments

hc, he, ht, hp	Crown heights (m): base, height of the lower edge of maximum width, height of the upper edge of maximum width, and top.
w	Crown width (m).
live_moisture, dead_moisture	Live and dead leaf moisture (proportion).
prop_dead	Proportion of dead foliage.
ignition_temp	Ignition temperature (deg C).
leaf_form	Leaf form, "Flat" or "Round".
leaf_thickness, leaf_width, leaf_length	Leaf dimensions (m).
leaf_separation	Mean separation between leaves (m).
stem_order	Stem (branching) order.
clump_diameter, clump_separation	Foliage clump diameter and separation (m).

Value

A list of derived species quantities (propLive, leafArea, leafMoisture, ignitabilityCoef, leavesPerClump, leafAreaIndex).

---

surface_fire *Surface fire behaviour*

Description

Head rate of spread and flame length of the surface fire from the site's slope, dead-fuel moisture, fuel load, litter geometry and surface wind.

Usage

```
surface_fire(
  slope,
  dead_fuel_moisture,
  fuel_load,
  mean_fuel_diameter,
  mean_fineness_leaves,
  wind_speed
)
```

Arguments

`slope`	Ground slope (radians).
`dead_fuel_moisture`	Dead fuel moisture content (proportion).
`fuel_load`	Surface fuel load (kg/m2).
`mean_fuel_diameter`	Mean surface fuel diameter (m).
`mean_fineness_leaves`	Mean leaf fineness of the surface fuel (m).
`wind_speed`	Wind speed at the surface (m/s).

Value

A list of surface-fire attributes, including head rate of spread and flame length.

Examples

```
surface_fire(slope = 0, dead_fuel_moisture = 0.1, fuel_load = 0.5,
             mean_fuel_diameter = 0.005, mean_fineness_leaves = 0.0004,
             wind_speed = 3)
```

Index

crown_polygon, 2
crown_ray_intersection, 3

fire_result, *5, 6*
fire_scenario, 3, *6–11*
fire_scenario(), *6*, *8*, *10, 11*
flame_angle, 4
framecpp_version, 5

parse_fire_result, 5
parse_fire_result(), *6*, *8*

run_fire_model, 6
run_fire_model(), *4*, *6*, *8*, *10, 11*

scenario_from_community, 7
scenario_from_community(), *3, 4*, *9–11*
scenario_from_params, 8
scenario_from_params(), *3, 4*, *6*, *8*
scenario_from_plant, 9
scenario_from_population, 10
scenario_from_population(), *9, 10*
species_derived, 11
surface_fire, 12