Package: rFDS (via r-universe)

May 20, 2026

Type Package

Title Fire Dynamics Simulation from LiDAR Data

Version 0.0.1

Description rFDS provides an interface between LiDAR-based vegetation structure data and the Fire Dynamics Simulator (FDS) for modeling forest and vegetation fire behavior. The package enables the conversion of classified LiDAR point clouds into voxelized fuel structures, generation of FDS input files, and automated simulation workflows for fire dynamics and fuel consumption analysis.

License MIT + file LICENSE

Encoding UTF-8

Imports lidR, grDevices, stats, utils, rgl

RoxygenNote 7.3.3

Config/pak/sysreqs libabsl-dev cmake libfreetype6-dev libgdal-dev gdal-bin libgeos-dev libglu1-mesa-dev make texlive libpng-dev libuv1-dev libgl1-mesa-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev zlib1g-dev

Repository https://carlos-alberto-silva.r-universe.dev

Date/Publication 2025-10-14 15:32:58 UTC

RemoteUrl https://github.com/carlos-alberto-silva/rFDS

RemoteRef HEAD

RemoteSha af3d7c720d3d23fc551fb25aa088d4597a64241d

Contents

`bndf_fds` *FDS '&BNDF' line*

Description

FDS '&BNDF' line

Usage

```
bndf_fds(QUANTITY, ...)
```

Arguments

`QUANTITY`	Character scalar. Quantity to output on boundaries.
`...`	Additional named parameters.

Value

Character scalar containing a single '&BNDF ... /' line.

build_fds *Concatenate FDS lines and ensure a trailing '&TAIL'*

Description

Concatenate FDS lines and ensure a trailing '&TAIL'

Usage

```
build_fds(..., tail = TRUE)
```

Arguments

...	Character vectors with FDS lines (in order).
tail	Logical scalar. If 'TRUE', append '&TAIL /' when missing (default 'TRUE').

Value

Character vector of FDS lines prepared for writing.

devc_fds *Create FDS &DEVC lines (point devices)*

Description

Build one or more Fire Dynamics Simulator (FDS) &DEVC entries. Devices are point sensors used for temperature, heat flux, etc.

Usage

```
devc_fds(...)

devc_fds(...)
```

Arguments

...	Named key/value pairs.
XYZ	Numeric vector (length 3) or matrix (n × 3) giving device coordinates. Each row is interpreted as (X, Y, Z).
ID	Optional character vector of device names. If missing, auto-numbers as DEVC_1, DEVC_2, ...
QUANTITY	FDS quantity to record (e.g. "TEMPERATURE", "HEAT FLUX"). A single string applies to all devices.
IOR	Optional integer 0–6 (direction for normal-oriented sensors). Omit for isotropic quantities (e.g., temperature).

Value

Character vector of formatted &DEVC lines.

Character scalar containing a single '&DEVC ... /' line.

Examples

```
devc_fds(c(10, 5, 2), ID = "TC1", QUANTITY = "TEMPERATURE")

mat <- rbind(c(0,0,0), c(1,1,1))
devc_fds(mat, QUANTITY = "HEAT FLUX")
```

---

domain_from_lidar *Derive domain bounds (XB) from LAS or voxel centers*

Description

Derive domain bounds (XB) from LAS or voxel centers

Usage

```
domain_from_lidar(las_or_vox, pad_xyz = 0, centers = FALSE)
```

Arguments

las_or_vox	Either a 'lidR::LAS' object or a voxel 'data.frame' from [laz_to_voxels()] (expects 'vxc, vyc, vzc, dx, dy, dz').
pad_xyz	Numeric length-1 or length-3. Padding added to min/max of each axis.
centers	Logical. If 'TRUE', expand voxel centers by half-cell to outer faces.

Value

Numeric length-6 vector: 'c(xmin, xmax, ymin, ymax, zmin, zmax)'.

dump_fds *FDS '&DUMP' line*

Description

FDS '&DUMP' line

Usage

```
dump_fds(...)
```

Arguments

...	Named key/value pairs (e.g., 'DT_HRR=1.', 'DT_DEVC=1.').

Value

Character scalar containing a single '&DUMP ... /' line.

head_fds *FDS '&HEAD' line*

Description

FDS '&HEAD' line

Usage

```
head_fds(CHID, TITLE = NULL)
```

Arguments

CHID	Character scalar. Unique case identifier used for file basenames.
TITLE	Character scalar or 'NULL'. Human-readable title for the run.

Value

Character scalar containing a single '&HEAD ... /' line.

init_fds *FDS '&INIT' line*

Description

FDS '&INIT' line

Usage

```
init_fds(...)
```

Arguments

...	Named key/value pairs (e.g., 'PART_ID='grass'', 'BULK_DENSITY_FILE='bdf/...'').

Value

Character scalar containing a single '&INIT ... /' line.

install_fds *Install FDS/SMV (interactive)*

Description

Downloads the official FDS/SMV bundle from NIST (GitHub releases) and launches the platform-specific installer.

Usage

```
install_fds(
  version = "6.10.1",
  dest_dir = tempdir(),
  sudo = interactive(),
  quiet_download = FALSE
)
```

Arguments

version	Character. Release version tag, e.g. "6.10.1". Defaults to a stable known version. You can change later.
dest_dir	Where to download the installer. Defaults to tempdir().
sudo	Logical. On macOS/Linux, run installer with sudo (default = interactive()).
quiet_download	Logical. Suppress download progress (FALSE by default).

Value

(Invisibly) a list with `installer` (path) and `status` (exit code or NA if detached).

Examples

```
## Not run:
install_fds()

## End(Not run)
```

`laz_to_voxels` *Voxelize a LAS/LAZ point cloud and assign a majority class per voxel*

Description

Builds a regular cubic grid and aggregates points into voxels of size ‘grid_res‘. Each occupied voxel is assigned the majority class from a point attribute column (e.g., ‘Classification‘) and returned with center coordinates, cell size, a ‘bulk_density‘ placeholder column, and basic statistics.

Usage

```
laz_to_voxels(
  las_input,
  grid_res = 1,
  class_col,
  default_bulk_density = 0.5,
  local = FALSE,
  filter_invalid_returns = FALSE
)
```

Arguments

`las_input`	Either a [‘lidR::LAS‘] object or a path to a ‘.las‘/‘.laz‘ file.
`grid_res`	Numeric scalar. Voxel edge length (same units as LAS; usually meters).
`class_col`	Character scalar. Name of the LAS attribute column containing classes.
`default_bulk_density`	Numeric scalar. Initial value written to the ‘bulk_density‘ column for every voxel (default ‘0.5‘).
`filter_invalid_returns`	Logical scalar. If ‘TRUE‘, removes returns with ‘ReturnNumber==0‘ or ‘NumberOfReturns==0‘ when those fields exist (default ‘FALSE‘).
`normalize_z`	Logical scalar. If ‘TRUE‘, shifts Z so ‘min(Z) = 0‘ before voxelization (default ‘FALSE‘).

Value

A ‘data.frame‘ with columns: - ‘vxc, vyc, vzc‘: voxel center coordinates - ‘dx, dy, dz‘: voxel size (each equal to ‘grid_res‘) - ‘bulk_density‘: numeric column initialized to ‘default_bulk_density‘ - ‘class_id‘: majority class ID per voxel - ‘n_points, maj_count, maj_prop‘: aggregation stats per voxel

---

matl_fds *FDS ‘&MATL‘ line*

Description

FDS ‘&MATL‘ line

Usage

```
matl_fds(ID, ...)
```

Arguments

ID	Character scalar. Material ID.
...	Named key/value pairs (e.g., ‘DENSITY=500.‘, ‘SPECIFIC_HEAT_RAMP='c_v'‘).

Value

Character scalar containing a single ‘&MATL ... /‘ line.

---

mesh_fds *FDS ‘&MESH‘ line with strict formatting*

Description

Ensures IJK are integers and XB is printed with fixed decimals.

Usage

```
mesh_fds(IJK, XB)
```

Arguments

IJK	Integer/numeric vector of length 3: number of cells in X,Y,Z.
XB	Numeric vector of length 6: ‘[xmin, xmax, ymin, ymax, zmin, zmax]‘.

Value

Character scalar containing a single ‘&MESH ... /‘ line.

`meshes_to_fds` *Convert a mesh tile list to ‘&MESH‘ lines*

Description

Convert a mesh tile list to ‘&MESH‘ lines

Usage

```
meshes_to_fds(mesh_list)
```

Arguments

`mesh_list`	List returned by [tile_meshes_from_extent()].

Value

Character vector of FDS ‘&MESH‘ lines.

`misc_fds_literal` *Template-safe literal ‘&MISC‘ line*

Usage

```
misc_fds_literal(TMPA = 14, HUMIDITY = 10.5, VERBOSE = "TRUE")
```

Arguments

`TMPA`	Numeric scalar. Ambient temperature (K).
`HUMIDITY`	Numeric scalar. Relative humidity (
	\itemVERBOSECharacter scalar (‘"TRUE"‘/‘"FALSE"‘). FDS verbosity flag.

Character scalar containing a single ‘&MISC ... /‘ line.

Template-safe literal ‘&MISC‘ line

obst_fds *FDS '&OBST' line (e.g., thin ignition patches)*

Description

FDS '&OBST' line (e.g., thin ignition patches)

Usage

```
obst_fds(XB, SURF_ID = NULL)
```

Arguments

XB	Numeric length-6 vector: obstacle bounds '[xmin, xmax, ymin, ymax, zmin, zmax]'.
SURF_ID	Optional character scalar surface to apply.

Value

Character scalar containing a single '&OBST ... /' line.

open_boundary_vents *Open-box vents on outer boundaries*

Description

Open-box vents on outer boundaries

Usage

```
open_boundary_vents(
  sides = c("XMIN", "XMAX", "YMIN", "YMAX", "ZMAX"),
  surf_id = "OPEN"
)
```

Arguments

sides	Character vector subset of 'c("XMIN","XMAX","YMIN","YMAX","ZMIN","ZMAX")'. Default opens five sides 'c("XMIN","XMAX","YMIN","YMAX","ZMAX")', leaving the ground ('ZMIN') closed.
surf_id	Character scalar surface ID to assign (default '"OPEN"').

Value

Character vector of FDS '&VENT' lines (one per side).

open_smv *Open a Smokeview (.smv) File*

Description

This function launches the **Smokeview** graphical interface to visualize the results of a completed **Fire Dynamics Simulator (FDS)** case. It works across Windows, macOS, and Linux systems, provided that the Smokeview executable is available (either specified manually or accessible through the system PATH).

Usage

```
open_smv(smv_file, smokeview_exe = NULL)
```

Arguments

smv_file	Character string. Full path to the '.smv' file, or to the FDS case name without the '.smv' extension. The function will automatically append '.smv' if missing.
smokeview_exe	Optional. Full path to the Smokeview executable ('Smokeview.exe' on Windows or 'smokeview' on Linux/macOS). If not provided, the function assumes that 'smokeview' is available in the system PATH.

Details

On **Windows**, this function uses 'cmd /c start' to open Smokeview in a detached process, allowing the GUI to run independently from R. On **macOS** and **Linux**, it executes the Smokeview command directly.

The working directory is temporarily set to the directory containing the '.smv' file, ensuring that all associated data files are found correctly by Smokeview.

Value

Opens the Smokeview GUI window for the specified simulation case. The function is called for its side effect and returns 'invisible(NULL)'.

Examples

```
## Not run:
# Windows example (explicit path to Smokeview)
open_smv(
  smv_file = "C:/Users/c.silva/Documents/rFDS/tests/fds_output/TEST_IGN_1_GROUND.smv",
  smokeview_exe = "C:/Program Files/FireModels/SMV6/Smokeview.exe"
)

# macOS/Linux example (Smokeview in PATH)
open_smv("~/fds_cases/TEST_IGN_1_GROUND.smv")

## End(Not run)
```

part_fds *FDS ‘&PART‘ line*

Description

FDS ‘&PART‘ line

Usage

```
part_fds(ID, ...)
```

Arguments

ID	Character scalar. Particle ID.
...	Named key/value pairs (e.g., ‘SURF_ID=’leaf surface’‘, ‘STATIC=.TRUE.‘).

Value

Character scalar containing a single ‘&PART ... /‘ line.

plot_voxels *Plot 3D Voxels (with optional FDS ignition overlays)*

Description

Visualize a voxelized point cloud in 3D using **rgl**.

Usage

```
plot_voxels(
  vox_df,
  style = c("boxes", "points"),
  color_by = c("class", "value"),
  value_col = "bulk_density",
  class_col = "class_id",
  alpha = 0.9,
  to_local = TRUE,
  origin = NULL,
  scale = c(1, 1, 1),
  bg = "gray95",
  point_size = 6,
  palette = "viridis",
  ignition = NULL,
  devices = NULL,
  device_color = "black",
  device_radius = 0.3,
  label_devices = TRUE
)
```

Arguments

vox_df	Data frame with columns: • vxc, vyc, vzc — voxel center coordinates (numeric) • dx, dy, dz — voxel side lengths (optional; not used by "points" style) • class_id — class label for color_by = "class" (or use class_col) • plus any numeric column used by value_col when color_by = "value"
style	One of c("boxes","points"). Default "boxes".
color_by	One of c("class","value"). Default "class".
value_col	Name of numeric column to map colors when color_by = "value". Default "bulk_density".
class_col	Name of categorical column to map colors when color_by = "class". Default "class_id".
alpha	Transparency in $[0, 1]$. Applies to points/cubes. Default 0.9.
to_local	Logical; if TRUE, translate coordinates so the minima become 0, then apply scale. Default TRUE.
origin	Optional numeric length-3 vector c(x0, y0, z0) used when to_local = TRUE. If NULL, uses the minima of vxc, vyc, vzc.
scale	Numeric length-3 (or scalar) scale factor(s). Default c(1,1,1).
bg	Background color. Default "gray95".
point_size	Point size for style = "points". Default 6.
palette	Name passed to hcl.colors. Default "viridis".
ignition	**Optional.** Character vector of FDS &OBST lines (each containing an XB=... six-number extent). Each line is rendered as a red ignition line at the slab's Y/Z midpoint, spanning X. Multiple lines are supported. See parse_ignition_lines.

Details

Styles

- style = "boxes": render each voxel as a cube colored by class_col or value_col.
- style = "points": draw voxel centers as colored points.

Coloring

- color_by = "class" uses the categorical column given by class_col.
- color_by = "value" uses the numeric column given by value_col (binned to a palette).

Ignition overlays If ignition is provided, the function parses one or many FDS &OBST lines that contain XB=... (axis-aligned boxes) and draws a red ignition *line* through the center of each slab (midpoints of Y and Z, spanning X from x1 to x2). Lines are transformed by to_local, origin, and scale in the same way as voxel coordinates.

Requires **rgl**. Ignition lines are parsed via parse_ignition_lines() (must be available in scope). If to_local = TRUE, the same origin/scale are applied to ignition lines.

Value

Invisibly returns the (possibly transformed) `vox_df`. Opens an interactive rgl window.

See Also

`parse_ignition_lines` for extracting `XB` from FDS `&OBST` lines.

Examples

```
## Not run:
# Minimal points view, class coloring:
plot_voxels(vox, style = "points", color_by = "class", class_col = "class_id")

# Value coloring:
plot_voxels(vox, style = "points", color_by = "value", value_col = "bulk_density")

# With multiple ignition slabs (FDS &OBST lines):
ign <- c(
  "&OBST XB=764402.874000, 764460.374000, 6580844.443000, 6580844.943000, 37.670200, 37.680200, SURF_ID='IGN FIRE'
  "&OBST XB=764402.874000, 764460.374000, 6580854.443000, 6580844.943000, 37.670200, 37.680200, SURF_ID='IGN FIRE'
)
plot_voxels(vox_df, style = "points", color_by = "class", to_local = FALSE, ignition = ign)

## End(Not run)
```

`radi_fds` *FDS '&RADI' line*

Description

FDS '&RADI' line

Usage

```
radi_fds(NUMBER_RADIATION_ANGLES = NULL, RADIATION_ITERATIONS = NULL)
```

Arguments

`NUMBER_RADIATION_ANGLES`	Optional integer scalar (e.g., 64).
`RADIATION_ITERATIONS`	Optional integer scalar (e.g., 2).

Value

Character scalar containing a single '&RADI ... /' line.

ramp_fds *FDS '&RAMP' line*

Description

FDS '&RAMP' line

Usage

```
ramp_fds(ID, T, F)
```

Arguments

ID	Character scalar. Ramp/curve ID.
T	Numeric scalar. Time (s).
F	Numeric scalar. Fraction/value at time T.

Value

Character scalar containing a single '&RAMP ... /' line.

reac_fds *FDS '&REAC' line*

Description

FDS '&REAC' line

Usage

```
reac_fds(...)
```

Arguments

...	Named key/value pairs (e.g., 'FUEL='CELLULOSE'', 'SOOT_YIELD=0.01').

Value

Character scalar containing a single '&REAC ... /' line.

run_fds — *Run an FDS case from the folder that contains the .fds and bdf/ subfolder*

Description

This runner sets the working directory to `dirname(fds_path)`, so that relative `BULK_DENSITY_FILE='bdf/....bdf'` references resolve. If `repair_bdf_paths=TRUE`, it will rewrite any `BULK_DENSITY_FILE` entries in the .fds to use `bdf/<basename>.bdf` when a matching file exists in `./bdf`.

Usage

```
run_fds(fds_path, fds_exe = NULL, np = 1L, repair_bdf_paths = TRUE)
```

Arguments

`fds_path`	Character. Full path to the `.fds` file.
`fds_exe`	Optional path to `fds_local` (e.g., `"C:/Program Files/FireModels/FDS6/bin/fds_local.bat"`). If NULL, common locations and the PATH are searched.
`np`	Integer number of MPI ranks (default 1).
`repair_bdf_paths`	Logical. If TRUE (default), scan and fix `BULK_DENSITY_FILE` references to ensure they point at `bdf/<filename>.bdf` when those files exist under `./bdf`.

Value

(Invisibly) the process exit status (0 means success).

slcf_fds — *FDS '&SLCF' line*

Description

FDS '&SLCF' line

Usage

```
slcf_fds(...)
```

Arguments

...	Named key/value pairs (e.g., 'PBZ=1.0', 'QUANTITY='TEMPERATURE'').

Value

Character scalar containing a single '&SLCF ... /' line.

spec_fds *FDS '&SPEC' line*

Description

FDS '&SPEC' line

Usage

```
spec_fds(ID, FORMULA = NULL)
```

Arguments

ID	Character scalar. Species ID.
FORMULA	Optional character scalar chemical formula.

Value

Character scalar containing a single '&SPEC ... /' line.

surf_fds *FDS '&SURF' line*

Description

FDS '&SURF' line

Usage

```
surf_fds(ID, ...)
```

Arguments

ID	Character scalar. Surface ID name.
...	Named key/value pairs to include (e.g., 'HRRPUA=800', 'GEOMETRY='CYLINDRICAL'').

Value

Character scalar containing a single '&SURF ... /' line.

tail_fds *FDS '&TAIL' line*

Description

FDS '&TAIL' line

Usage

```
tail_fds()
```

Value

Character scalar '"&TAIL /"'.

tile_meshes_from_extent *Tile a domain into multiple '&MESH'es given target cell sizes*

Description

Tile a domain into multiple '&MESH'es given target cell sizes

Usage

```
tile_meshes_from_extent(
  XB,
  cell_size = c(0.5, 0.5, 0.5),
  max_ijk_per_mesh = 120
)
```

Arguments

XB	Numeric length-6 domain bounds '[xmin, xmax, ymin, ymax, zmin, zmax]'.
cell_size	Length-3 numeric: target grid spacing '(dx, dy, dz)' in meters.
max_ijk_per_mesh	Integer scalar. If 'I' or 'J' exceeds this threshold, tiles are split evenly along X and/or Y so per-mesh counts stay reasonable. Z is not split.

Value

A list of tiles, each being 'list(IJK=c(I,J,K), XB=c(...))'.

time_fds *FDS ‘&TIME‘ line*

Description

FDS ‘&TIME‘ line

Usage

```
time_fds(T_END, DT = NULL, DT_OUTPUT = NULL)
```

Arguments

T_END	Numeric scalar. Simulation end time (s).
DT	Optional numeric scalar. Time step (s).
DT_OUTPUT	Optional numeric scalar. Output interval (s).

Value

Character scalar containing a single ‘&TIME ... /‘ line.

vent_fds *FDS ‘&VENT‘ line (MB-based safe outer boundary vents)*

Description

FDS ‘&VENT‘ line (MB-based safe outer boundary vents)

Usage

```
vent_fds(MB = NULL, SURF_ID = NULL)
```

Arguments

MB	Character scalar: one of ‘'"XMIN","XMAX","YMIN","YMAX","ZMIN","ZMAX"‘.
SURF_ID	Character scalar. Surface ID (e.g., ‘'OPEN'‘).

Value

Character scalar containing a single ‘&VENT ... /‘ line.

`voxel_to_bdf` *Write one BDF file per class from a voxel table*

Description

Splits a voxel table by ‘class_col‘ and writes a ‘.bdf‘ per class using the naming pattern ‘<bdf_base_path>_class_<id>.bdf‘. The file header stores the overall extent (‘VXMIN..VZMAX‘), grid spacing (‘dx,dy,dz‘), and record count. Each record contains a voxel center ‘(vxc, vyc, vzc)‘ and the value from ‘value_col‘ (e.g., ‘bulk_density‘).

Usage

```
voxel_to_bdf(
  vox_df,
  bdf_base_path,
  class_col = "class_id",
  value_col = "bulk_density",
  include_na = FALSE,
  normalize_xyz = FALSE
)
```

Arguments

`vox_df`	A ‘data.frame‘ from [laz_to_voxels()] containing ‘vxc, vyc, vzc, dx, dy, dz‘ and ‘value_col‘.
`bdf_base_path`	Character path to the **base** BDF path (with or without ‘.bdf‘). Outputs are named ‘<base>_class_<id>.bdf‘.
`class_col`	Character scalar. Voxel class column name (default ‘"class_id"‘).
`value_col`	Character scalar. Column to write as the BDF value (default ‘"bulk_density"‘).
`include_na`	Logical scalar. If ‘TRUE‘, also writes a file for ‘NA‘ class (default ‘FALSE‘).
`normalize_xyz`	Logical scalar. If ‘TRUE‘, shift centers so mins are zero before writing.

Value

Invisibly, a named character vector of the paths of written BDF files (one per class ID).

wind_fds *FDS ‘&WIND‘ line*

Description

FDS ‘&WIND‘ line

Usage

```
wind_fds(SPEED = NULL, DIRECTION = NULL, L = NULL, Z_0 = NULL)
```

Arguments

SPEED	Optional numeric scalar. Reference wind speed (m/s).
DIRECTION	Optional numeric scalar. Wind direction (deg, meteorological).
L	Optional numeric scalar. Monin-Obukhov length (m).
Z_0	Optional numeric scalar. Surface roughness length (m).

Value

Character scalar containing a single ‘&WIND ... /‘ line.

write_fds *Write an FDS input file to disk (truncate after ‘&TAIL‘)*

Description

Write an FDS input file to disk (truncate after ‘&TAIL‘)

Usage

```
write_fds(path, ...)
```

Arguments

path	Character scalar. Output ‘.fds‘ path.
...	Character vectors of lines to pass to [build_fds()].

Value

Invisibly, the output path ‘path‘.

Index