Package: gloBFPr (via r-universe)

October 4, 2026

Type Package

Title Urban Environment Analysis Using Global Building Height Datasets

Version 2.0.1

Author Xiaohao Yang [aut, cre, cph]

Maintainer Xiaohao Yang <xiaohaoy111@gmail.com>

Description Automated access and analysis of multidimensional building profile in cities. The package provides tools to access, search, and download global 3D building footprint datasets. It includes functions to retrieve building height tiles, compute a range of 2D/2.5D building morphology metrics, assess environmental context (e.g., greenery accessibility), aggregate metrics to the city-block scale, support noise, shadow/radiation, and wind simulation, export study areas as 3D scenes (terrain, buildings, canopy) for 3D modeling software, such as Rhino3D and Blender.

License MIT + file LICENSE

URL https://github.com/billbillbilly/gloBFPr

BugReports https://github.com/billbillbilly/gloBFPr/issues

Encoding UTF-8

LazyData true

Language en-US

Depends R (>= 4.2)

Suggests testthat (>= 3.0.0), knitr, rmarkdown, dsmSearch, viewscape (>= 2.0.1), greenSD, ggplot2, osmdata, sfnetworks, tidygraph, tigris, duckdb, DBI

Imports Rcpp, grDevices, stats, igraph, sf, dplyr, httr2, terra, utils, rlang, cli, nominatimlite, parallelly, future, furrr, ecmwfr

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

VignetteBuilder knitr, rmarkdown

LinkingTo Rcpp

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libglpk-dev libsecret-1-dev libxml2-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

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

Date/Publication 2026-09-04 17:39:59 UTC

RemoteUrl https://github.com/billbillbilly/globfpr

RemoteRef HEAD

RemoteSha cc67d7c044cea13b4b7c2561fa8c6641d6f25aaa

Contents

add_flow_vectors . . . 3
aggregate_block . . . 4
generate_block . . . 5
get_3d_world . . . 6
get_era5_met . . . 12
get_fused_dsm . . . 14
get_metadata . . . 15
get_metrics . . . 16
get_noise_map . . . 22
get_shadows . . . 26
globfp_example . . . 31
globfp_example_canopy_height . . . 31
globfp_example_dem . . . 32
infer_osm_traffic . . . 33
install_noisemodelling . . . 34
osm_noise_traffic_defaults . . . 34
plot_bgvi_viewshed . . . 35
plot_foam_map . . . 36
plot_noise_map . . . 38
prepare_foam_case . . . 39
prepare_foam_geometry . . . 42
prepare_noisemodelling_inputs . . . 43
prepare_openfoam_inputs . . . 45
read_foam_pedestrian_slice . . . 48
run_openfoam_docker . . . 50
sample_foam_slice . . . 51
search_3dglobdf . . . 53

Index **55**

add_flow_vectors *Add wind / flow vector arrows to a foam map plot*

Description

Takes a ggplot object produced by plot_foam_map and overlays velocity arrows sampled on a regular sub-grid.

Usage

```
add_flow_vectors(
  p,
  r,
  spacing = 20,
  scale = 1,
  colour = "black",
  alpha = 0.6,
  linewidth = 0.25,
  arrow_size = 0.07,
  u_ref = NULL,
  quiet = FALSE
)
```

Arguments

p	A ggplot object (output of plot_foam_map).
r	A SpatRaster that contains layers named Ux and Uy.
spacing	Numeric. Arrow sub-grid spacing in the same units as the raster coordinates (usually metres). Default 20.
scale	Numeric. Arrow length multiplier. Default 1.
colour	Character. Arrow colour. Default "black".
alpha	Numeric. Arrow opacity (0-1). Default 0.6.
linewidth	Numeric. Line thickness in mm. Default 0.25 (thin).
arrow_size	Numeric. Arrowhead length in cm. Default 0.07 (tiny).
u_ref	Numeric. Speed (m/s) the longest arrow represents. The default, NULL, normalises to the field's own maximum, which makes a near-stagnant field look as vigorous as a strong one; pass a fixed value to put several maps on one scale.
quiet	Logical. Suppress the message reporting the arrow scale.

Value

The same ggplot object with arrows added.

aggregate_block *aggregate_block*

Description

Aggregate building-level metrics to block level using the output of `generate_block()`. Additive quantities (areas, volumes, population) are summed by default; per-building indices (shape metrics, elongation ratios, etc.) are averaged. Both defaults can be overridden per column via `.fns`. Two derived metrics are always added: `n_buildings` (count of buildings per block) and `coverage_ratio` (total building footprint area / block area).

Usage

```
aggregate_block(
  block_output,
  .fns = NULL,
  population = FALSE,
  population_year = 2025,
  residential = FALSE,
  residential_year = 2020,
  quiet = FALSE
)
```

Arguments

`block_output`	list. The named list returned by `generate_block()`, containing `$blocks` (sf polygons) and `$buildings` (sf with `block_id`).
`.fns`	named list. Optional overrides mapping column names to aggregation functions, e.g. `list(vol = max, Height = median)`. Any column not named here uses the built-in default (sum or mean as described above).
`population`	logical. If `TRUE`, fetch GHSL population at block level and add a `pop_total` column. Default `FALSE`.
`population_year`	integer. GHSL population year to use when `population = TRUE`. One of 1975, 1980, ..., 2025, 2030. Default `2025`.
`residential`	logical. If `TRUE`, fetch GHS built-up surface rasters at block level and compute `res_prop` (residential built-up surface fraction per block). Default `FALSE`. Note: per-building `res` flags are excluded from block-level aggregation - use this parameter for block-level residential proportion instead.
`residential_year`	integer. GHS built-up surface year to use when `residential = TRUE`. One of 1975, 1980, ..., 2020, 2025, 2030. Default `2020`.
`quiet`	logical. If `TRUE`, suppress cli messages. Default `FALSE`.

Value

The blocks sf object from block_output with one column per aggregated metric, plus n_buildings and coverage_ratio. When population = TRUE, also includes pop_total. When residential = TRUE, also includes res_prop.

---

generate_block *generate_block*

Description

Cluster given buildings into blocks based on street network. Uses a two-stage approach: (1) vector polygonization of the road network for well-formed areas, then (2) a raster fallback for buildings that fall in network gaps or dead-end pockets. Blocks with no buildings are dropped.

Before polygonization, dual carriageways (motorways and trunk roads represented as parallel lines) are simplified to single centrelines to prevent artificially narrow slivers between them from being misidentified as blocks. The simplification approach is conceptually adapted from UrbanWaterBlocks (Yin et al., 2025).

Usage

```
generate_block(
  x,
  network = NULL,
  network_source = c("overture", "osm"),
  overture_release = NULL,
  res = 2,
  min_block_area = 500,
  dc_highway_types = c("motorway", "trunk"),
  dc_overlap_threshold = 0.7,
  quiet = FALSE
)
```

Arguments

x	sf. Building footprint polygons, typically output from search_3dglobdf().
network	sf or character. Optional road network or path to one. When supplied, network_source is ignored.
network_source	character. Source for automatic network fetching when network = NULL. Either "overture" (Overture Maps via DuckDB parquet query, default, requires the duckdb and DBI packages) or "osm" (OpenStreetMap via the Overpass API, slower for large areas).
overture_release	character or NULL. Overture Maps release string used when network_source = "overture", e.g. "2025-03-19.0". The default NULL (or "auto") queries the public bucket for the newest release and caches it for the session. See https://github.com/OvertureMaps/data/releases for available releases.

`res` numeric. Raster resolution in metres for the fallback stage. Default `2`.

`min_block_area` numeric. Minimum block area in m^2 below which polygons are treated as slivers and merged into neighbours (raster stage) or dropped (polygonize stage). Default `500`.

`dc_highway_types` character vector. Highway class values treated as dual carriageway candidates (OSM `highway` tag or Overture `class` column). Default `c("motorway", "trunk")`.

`dc_overlap_threshold` numeric. Minimum overlap fraction (0-1) for a line to be considered a duplicate carriageway and removed. Default `0.7`.

`quiet` logical. If `TRUE`, suppress cli messages. Default `FALSE`.

Value

A named list with two elements:

`blocks` An `sf` polygon object, one row per block, with a `block_id` column. CRS matches the input `x`.

`buildings` The input `x` with an added `block_id` integer column linking each building to its block. Buildings that cannot be assigned to any block receive `NA`.

References

Yin, H., et al. (2025). UrbanWaterBlocks: A python tool for block-based urban water management. Sustainable Cities and Society.

`get_3d_world` *get_3d_world*

Description

Export a study area as a 3D scene (inspired by the arnis project): terrain surface, extruded building footprints, and canopy tree objects, written as Wavefront OBJ and/or binary STL files that load directly in Rhino3D and Blender.

The OBJ output contains named objects per layer (`terrain`, `buildings`, `canopy_trunks`, `canopy_crowns`) and one group per building (`building_<id>`), so individual buildings remain selectable after import. Ground surface classes (greenspace, roads, sidewalks, paths, water, sand) are part of the terrain object itself, as face groups with their own materials - the terrain is one continuous, detailed surface with differently colored regions, not a stack of separate ribbons. STL has no object concept, so one STL file is written per layer.

Usage

```
get_3d_world(
  x = NULL,
  bbox = NULL,
  place = NULL,
  terrain = TRUE,
  canopy = "metachm",
  key = NULL,
  dem = NULL,
  canopy_height = NULL,
  height_col = "Height",
  format = c("obj", "stl"),
  out_dir = "world3d",
  min_tree_height = 2,
  tree_window = 5,
  max_trees = 20000,
  simplify_terrain = 1,
  color_by = NULL,
  facade_palette = FALSE,
  roads = NULL,
  overture_release = NULL,
  bridges = TRUE,
  bridge_level_height = 6,
  bridge_ramp = 20,
  bridge_pillar_interval = 25,
  water = NULL,
  sand_buffer = 3,
  surface_res = 2,
  greenspace = NULL,
  greenspace_under_canopy = TRUE,
  greenspace_zoom = 17,
  greenspace_year = NULL,
  all_vox = FALSE,
  vox_size = 1,
  basemap = FALSE,
  local_origin = TRUE,
  crop = FALSE,
  data_source = "GBF",
  quiet = TRUE
)
```

Arguments

x	sf. Building footprint polygons, typically the output of `search_3dglobdf()` (must contain the column given by `height_col`). If `NULL`, buildings are fetched via `search_3dglobdf()` using bbox/place.
bbox	`sf`, `sfc`, or numeric vector (xmin, ymin, xmax, ymax) in WGS84. Ignored when `x` is provided.

`place` character (optional). Address or place name, passed to `search_3dglobdf()`. Ignored when `x` is provided.

`terrain` logical. If `TRUE` (default), download a DEM (OpenTopography via `dsmSearch`; requires `key`) and mesh the ground surface. Buildings and trees are then placed at their sampled ground elevation. If `FALSE`, the scene has a flat ground at z = 0 and no API key is needed.

`canopy` character or `NULL`. Canopy height source, `"metachm"` (default) or `"ethchm"`; `NULL` skips trees.

`key` character. OpenTopography API key, required when `terrain = TRUE` and no `dem` raster is supplied.

`dem` `SpatRaster` or `PackedSpatRaster` (optional). A pre-loaded digital elevation model (e.g. `globfp_example_dem`). When supplied with `terrain = TRUE`, no download or API key is needed. Ignored when `terrain = FALSE`.

`canopy_height` `SpatRaster` or `PackedSpatRaster` (optional). A pre-loaded canopy height model (e.g. `globfp_example_canopy_height`). When supplied, it is used for tree detection instead of downloading from the `canopy` source.

`height_col` character. Building height column. Default `"Height"`.

`format` character. Any of `"obj"`, `"stl"`. Default writes both.

`out_dir` character. Output directory (created if missing).

`min_tree_height` numeric. Minimum canopy height in metres. Default 2.

`tree_window` numeric. Local-maximum search window in **metres** for tree detection: at most one tree is placed per window. Larger values give fewer, more widely spaced trees; lower it (e.g. 3) if dense canopy looks too sparse in the model. Default 5.

`max_trees` integer. Cap on tree objects; the tallest trees are kept and a warning reports how many were detected. Default 20000.

`simplify_terrain` integer. Aggregation factor for the terrain mesh (1 = full DEM resolution). Default 1.

`color_by` character or `NULL`. Name of a numeric column in `x` mapped to per-building OBJ materials via a viridis ramp (OBJ only; STL carries no color). Default `NULL`.

`facade_palette` logical or character vector. If `TRUE`, buildings get varied muted facade colors (arnis-style) instead of uniform grey, assigned deterministically per building. Supply a character vector of R colors to use a custom palette. Ignored when `color_by` is set. Default `FALSE`.

`roads` `NULL`, `"overture"`, or an sf line layer. Paints road and sidewalk surfaces into the terrain using the arnis method (class-based widths; asphalt/concrete/dirt terrain colors). `"overture"` fetches transportation segments from Overture Maps via a native DuckDB parquet query (requires the suggested `duckdb` and `DBI` packages and internet, matching `generate_block()`'s road source). An sf layer should contain LINESTRING geometries with optional `class` and `subclass` columns (Overture/OSM highway classes); missing classes default to `residential`. Default `NULL` (no roads).

overture_release
: character or NULL. Overture Maps release string used when roads = "overture" or water = "overture", e.g. "2025-03-19.0". The default NULL (or "auto") queries the public bucket for the newest release and caches it for the session, so the code keeps working as Overture publishes new releases. See https://github.com/OvertureMaps/data/releases for the list.

bridges
: logical. If TRUE (default), road segments flagged as bridges are built as elevated structures instead of being painted on the ground: a deck raised level * bridge_level_height metres above the highest terrain along the span, ramps down to grade at each end, railings, and support pillars. Tunnel segments are always omitted from the ground surface. Requires bridge attributes in the road data (Overture supplies them; an sf layer may carry is_bridge, is_tunnel, and level).

bridge_level_height
: numeric. Metres of clearance per level, the arnis LAYER_HEIGHT_STEP. Default 6.

bridge_ramp
: numeric. Length in metres over which a deck ramps down to ground level at each end. Default 20.

bridge_pillar_interval
: numeric. Spacing in metres between support pillars. Default 25.

water
: NULL, "overture", or an sf (MULTI)POLYGON layer of water bodies. Rivers, lakes, and coastal water are handled with the arnis method: the terrain under each water body is flattened to just below its lowest bank and colored as water, and a sand fringe of sand_buffer metres is painted along the shore (riparian and coastal edges). "overture" fetches water polygons from the Overture Maps base theme via the native DuckDB query. Default NULL.

sand_buffer
: numeric. Width in metres of the sand fringe around water bodies. 0 disables the fringe. Default 3.

surface_res
: numeric. Grid resolution in metres for the classified terrain surface in the default (non-voxel) mode. Smaller values follow road edges more precisely at the cost of a denser mesh. Default 2.

greenspace
: NULL, TRUE, "esri", "sentinel2", or a SpatRaster. Paints ground-level greenery (lawns) into the terrain. Greenery is classified from map tiles by the suggested greenSD package: TRUE or "esri" uses Esri imagery, "sentinel2" uses Sentinel-2. Alternatively supply your own binary SpatRaster (green = 1). Cells where the canopy height model is at or above min_tree_height are excluded, so this surface class shows ground-level greenery only - tree canopy is already represented by the 3D tree objects. Default NULL.

greenspace_under_canopy
: logical. If TRUE (default), ground under tree canopy stays green - trees stand on grass, as in arnis. Set FALSE to cut canopy cells out of the greenspace surface, leaving bare ground beneath the crowns.

greenspace_zoom
: integer. Map-tile zoom level for the greenspace classification; higher zoom gives finer lawn edges. Default 17.

greenspace_year
: integer or NULL. Imagery year for greenspace = "sentinel2".

`all_vox` logical. If `TRUE`, voxelize the entire scene so the model reproduces the blocky world arnis builds, from this package's data inputs: terrain becomes stepped blocks (Minecraft-style heightmap), buildings become block columns on flattened quantized bases, and trees are already voxel objects. Building footprints are rasterized at `vox_size`, so exact footprint edges are traded for the voxel look. Default `FALSE`.

`vox_size` numeric. Block edge length in metres for `all_vox` mode (1 = Minecraft-like scale). Larger values give chunkier, lighter models. Default 1.

`basemap` logical. If `TRUE`, download an Esri World Imagery snapshot of the scene, save it next to the OBJ, and drape it on the ground via texture coordinates (OBJ only; requires internet). With `terrain = FALSE` a flat textured ground plane is added. On download failure the export falls back to the flat terrain color with a warning. Default `FALSE`.

`local_origin` logical. If `TRUE` (default), shift the scene so its minimum x/y is at (0, 0). The offset and CRS are stored in `world_metadata.json` and in the returned object, so results can be georeferenced back. Strongly recommended for Rhino/Blender, which lose float precision at UTM-scale coordinates.

`crop, data_source`
Passed to `search_3dglobdf()` when `x` is `NULL`.

`quiet` logical. Suppress progress messages. Default `TRUE`.

Details

All coordinates are in metres (scene UTM zone). Building walls extend 1 m below their sampled ground elevation when `terrain = TRUE` to avoid gaps on slopes. Trees are blocky voxel-style objects whose shapes are ported from the arnis project (trunk column, leaf columns, apex cap, and concentric canopy rings). Every tree's total height comes from the canopy height raster; the variant is chosen from that height (compact oaks below 8 m, standard/bushy oaks to 14 m, oak/spruce to 22 m, towering spruce above, with position-hashed variety within each class), then the block size is scaled so the tree matches its measured height exactly.

Ground surface classes are painted into the terrain, not built as separate geometry: a classification grid (`surface_res` metres in default mode, `vox_size` in voxel mode) assigns each terrain cell one of ground, greenspace, roads, sidewalks, paths, sand, or water, and the terrain mesh carries these as per-face material groups. Roads follow the arnis method: each segment gets a class-based half-width (motorway/trunk/primary 5 m, secondary 4 m, tertiary 3 m, residential/service 2 m, foot/cycle/path classes 1 m - the arnis `highway_block_range` defaults), is buffered into a ribbon, and painted as `roads` (asphalt), `sidewalks` (concrete; includes Overture `subclass = "sidewalk"`), or `paths` (dirt).

Bridges and elevated highways follow arnis's structural approach. Neither OSM nor Overture records deck elevations, so the height is inferred from the segment's `level`: the deck sits `level * bridge_level_height` metres (arnis uses 6 blocks per layer) above the highest terrain along the span, ramps linearly down to grade over `bridge_ramp` metres at each end, and is carried by pillars dropped to the ground every `bridge_pillar_interval` metres. Decks are solid slabs with railings, sized by the same class-based widths as surface roads. Bridge and tunnel segments are excluded from the terrain surface classification, so a bridge no longer leaves a road painted on the ground beneath it.

Riparian and coastal areas also follow arnis: the terrain under each water body is flattened to one level just below its lowest bank (0.2 m in default mode, one block in voxel mode), painted as water, and fringed with a `sand_buffer`-metre sand strip along the shoreline.

Vegetation is split across two levels so canopy and lawns are never conflated: tree canopy comes from the canopy height model and is rendered as 3D tree objects, while the `greenspace` terrain class carries the ground surface. Greenery comes from `greenSD`'s map-tile classification (Esri or Sentinel-2 imagery at `greenspace_zoom`). Ground under canopy stays grass by default, so a park reads as a continuous lawn with trees standing on it - exactly how arnis places tree objects on grass blocks. Use `greenspace_under_canopy = FALSE` if you instead want canopy cells cut out of the lawn.

With `all_vox = TRUE` the whole scene is voxelized the way arnis builds its Minecraft worlds: the DEM becomes a stepped block heightmap, each building footprint is rasterized at `vox_size` and raised as quantized block columns on a flattened base (extending one block below ground), and the voxel trees stand on the quantized ground. Heights still come from the input data (building `Height` column, CHM tree heights, DEM terrain) - only the geometry representation changes. Expect larger files than the default mode; increase `vox_size` to lighten them.

When `basemap = TRUE`, imagery is retrieved from the Esri World Imagery service; check the Esri terms of use and provide attribution (Esri, Maxar, Earthstar Geographics, and the GIS User Community) when publishing rendered scenes. Footprint holes (courtyards) are not carved in this version; the exterior ring is extruded and a warning is issued when holes are dropped.

Value

(invisibly) a list:

- `paths`: character vector of written files.
- `meshes`: named list of in-memory meshes (`vertices`/`faces`).
- `origin`: list with `x`, `y` offset and `epsg` of the scene CRS.
- `n_buildings`, `n_trees`: scene composition counts.
- `n_trees_detected`: trees found in the canopy height model before the `max_trees` cap - compare with `n_trees` to see how many were dropped.
- `n_bridges`: elevated bridge segments built.

Why some trees are missing

Three filters reduce canopy pixels to tree objects. `min_tree_height` (default 2 m) drops low vegetation; `tree_window` keeps only one tree per window, so closely spaced trees merge into their tallest neighbour; and `max_trees` caps the total, keeping the tallest. Compare `n_trees_detected` with `n_trees` in the returned object, and lower `tree_window` or raise `max_trees` if the model looks sparser than the canopy data. Note that `tree_window` cannot resolve trees closer together than the CHM resolution: on a coarse (e.g. aggregated) canopy raster, set `tree_window` at or below the cell size to place one tree per canopy cell.

Examples

```
example <- gloBFPr::globfp_example
world <- get_3d_world(
```

```
  x = example, terrain = FALSE, canopy = NULL,
  out_dir = tempfile("world3d")
)
```

`get_era5_met` *Fetch ERA5 conditions for OpenFOAM boundary conditions*

Description

Downloads one hour of ERA5 reanalysis data (10-m wind components, 2-m temperature, skin temperature) from the Copernicus Climate Data Store for a single location and time step. Returns the values pre-formatted for direct use as arguments to `prepare_foam_case()` for wind simulation.

ERA5 is a global reanalysis with 0.25 deg (~28 km) spatial resolution and hourly temporal resolution from 1940 to present. It is **not** a local measurement; treat it as a representative synoptic condition, not a site-specific reading.

Usage

```
get_era5_met(
  lon,
  lat,
  datetime,
  cds_key = Sys.getenv("CDS_API_KEY"),
  cache_dir = tempdir(),
  quiet = FALSE
)
```

Arguments

`lon`	Numeric. Longitude of the site in decimal degrees (WGS-84).
`lat`	Numeric. Latitude of the site in decimal degrees (WGS-84).
`datetime`	POSIXct or character `"YYYY-MM-DD HH:MM"` in UTC. ERA5 is available at full hours; the nearest hour is used automatically.
`cds_key`	Character. CDS personal access token. Defaults to the `CDS_API_KEY` environment variable.
`cache_dir`	Character. Directory for caching downloaded NetCDF files so repeated calls for the same location and time are instant. Default `tempdir()`.
`quiet`	Logical. Suppress progress messages. Default `FALSE`.

Value

A named list (invisibly) containing:

`inlet_velocity` `c(u, v, 0)` in m/s - direct input for `prepare_foam_case(inlet_velocity = ...)`. x = east, y = north, matching a UTM-projected domain.

`z_ref` Reference height for the wind measurement - always 10 m.

`T_ref` ERA5 2-m air temperature in K. In wind-only `prepare_foam_case()` runs, this is used as the uniform reference temperature that switches buoyancy off.

`T_skin` ERA5 skin (land-surface) temperature in K. Useful as a `T_ground` estimate; `NULL` if unavailable.

`wind_speed_ms` Scalar 10-m wind speed in m/s.

`wind_dir_deg` Meteorological wind direction in degrees (direction FROM which the wind blows; 0 = from North, 90 = from East).

`u10`, `v10` Raw ERA5 eastward and northward wind components (m/s).

`datetime` Rounded POSIXct of the ERA5 time step used.

`lon`, `lat` Site coordinates as supplied.

Authentication

You need a free Copernicus CDS account and a personal access token.

1. Register at https://cds.climate.copernicus.eu
2. Copy your personal access token from your user profile page.
3. Set it once per session: `Sys.setenv(CDS_API_KEY = "your-token-here")` or store it in `~/.Renviron` so it loads automatically.

The legacy CDS was retired in 2024, so `ecmwfr` >= 2.0.0 is required for current tokens. This function detects the installed `ecmwfr` API version and adapts automatically, but on older versions the request will be rejected by the server. Upgrade with `install.packages("ecmwfr")` if you hit auth errors.

You must also accept the dataset licence once, in the browser, at https://cds.climate.copernicus.eu/datasets/reanalysis-era5-single-levels (the "Terms of use" tab). Requests fail with a licence error until you do.

See Also

`prepare_foam_case()`

Examples

```
## Not run:
Sys.setenv(CDS_API_KEY = "your-token-here")

# Summer evening in Detroit
met <- get_era5_met(
  lon               = -83.05,
  lat               =  42.34,
```

```
  datetime          = "2023-07-15 22:00"
)

## ---- Wind simulation ------------------------------------------------
prepare_foam_case(
  case_dir       = "path/to/case",
  stl_file       = "path/to/buildings.stl",
  domain         = list(xmin = 0, xmax = 500, ymin = 0, ymax = 500,
                        zmin = 0, zmax = 200),
  inlet_velocity = met$inlet_velocity,
  z_ref          = met$z_ref,
  T_ref          = met$T_ref
)

## End(Not run)
```

`get_fused_dsm` *get_fused_dsm*

Description

Generate digital surface model using multiple datasets, including building height map, canopy height map, and terrain model. Each building is given a single flat roof elevation (its own base ground elevation, sampled at its centroid, plus its height) rather than following the terrain slope beneath it pixel by pixel, so buildings on sloped ground do not come out tilted or warped.

Usage

```
get_fused_dsm(
  x = NULL,
  datasource_canopy_height = "metachm",
  min_tree_height = 2,
  resolution = NULL,
  opentopo_key = NULL,
  key = NULL,
  quiet = TRUE
)
```

Arguments

`x`	sf. building footprint polygon, typically output from `search_3dglobdf()`
`datasource_canopy_height`	character or `NULL`. Canopy height source. Currently supports `"metachm"`, `"ethCHM"`, or `NULL`.
`min_tree_height`	numeric. Minimum canopy height threshold in meters.

resolution	numeric or NULL. Output raster resolution in meters. If NULL (default), the finest native resolution among the downloaded DEM and canopy height model is used, so the output is never silently degraded to the coarser of the two source rasters. Set explicitly (e.g. 1) to force a finer grid than the native source data (e.g. when the DEM falls back to coarse SRTM data), or a coarser one to speed up large areas.
opentopo_key	character. OpenTopography API key used to download DEM data.
key	Deprecated alias for opentopo_key.
quiet	logical. If TRUE, suppress cli messages and progress output. Default is TRUE.

Examples

```
## Not run:
 example <- gloBFPr::globfp_example
 dsm <- get_fused_dsm(x= example, opentopo_key = 'key')

## End(Not run)
```

---

get_metadata *get_metadata*

Description

Returns a spatial grid (as an sf object) containing metadata and download URLs for global 3D building footprint tiles (3D-GloBFP).

Usage

```
get_metadata(test = FALSE, quiet = TRUE)
```

Arguments

test	logic, Ignored during normal use; included for internal testing purposes. Defaults to FALSE.
quiet	logical. Accepted for consistency with other package functions; currently unused because this function does not emit cli messages.

Details

The metadata of 3D Global Building Footprints (3D-GloBFP) dataset is uploaded on zenodo. More detials about this dataset can to found here.

The data is detailed in the following article

Value

sf a spatial polygon grid with attributes: id, gridID, bounding box coordinates, and download_url.

References

Che, Y., Li, X., Liu, X., Wang, Y., Liao, W., Zheng, X., Zhang, X., Xu, X., Shi, Q., Zhu, J., Zhang, H., Yuan, H., & Dai, Y. (2025). 3D-GloBFP: the first global three-dimensional building footprint dataset. Zenodo. https://doi.org/10.5281/zenodo.15487037

Che Yangzi, Li Xuecao, Liu Xiaoping, Wang Yuhao, Liao Weilin, Zheng Xianwei, Zhang Xucai, Xu Xiaocong, Shi Qian, Zhu Jiajun, Zhang Honghui, Yuan Hua, & Dai Yongjiu (2024). 3D-GloBFP: the first global three-dimensional building footprint dataset. Earth Syst. Sci. Data, 16, 5357-5374

Examples

```
meta <- gloBFPr::get_metadata(test=TRUE)
```

`get_metrics` *get_metrics*

Description

`get_morphology`: Computes a set of morphological properties and geometric descriptors for each building footprint polygon. These include area, perimeter, surface area, volume, shape compactness, elongation, and accessibility measures. If `x` contains a `group_id` column, features with the same `group_id` are treated as one building for morphology metrics while the returned rows and geometries remain unchanged.

`get_neighbors`: Computes the number of neighboring buildings and centroid distance summaries, based on a fixed-radius buffer and Voronoi-based adjacency.

`get_bgvi`: Calculate the Building Green View Index (BGVI) for each building volume. Neighboring buildings in the internal DSM used for the viewshed are given a single flat roof elevation (base ground elevation at their centroid, plus height) rather than following the terrain slope pixel by pixel, matching `get_fused_dsm()`.

`get_dng`: Calculate the distance to the nearest greenspace (DNG) patch for each building volume

Usage

```
get_morphology(
  x = NULL,
 metrics = c("g_area", "pmeter", "v_surf", "t_surf", "vol", "obb_vol", "pa_ratio",
   "rec", "fra", "cbn", "hem", "cnv", "me_dist", "mp_dist", "vol_exch", "elo_x",
    "elo_y", "elo_z"),
  quiet = FALSE
)

get_neighbors(x = NULL, radius = 500, quiet = FALSE)

get_bgvi(
  x = NULL,
```

```
  datasource_canopy_height = "metachm",
  datasource_greenspace = NULL,
  min_tree_height = 2,
  zoom = 17,
  radius = 800,
  year = NULL,
  floor = FALSE,
  floor_step = 3,
  short_building_threshold = 6,
  field_of_view = 45,
  directions = NULL,
  workers = NULL,
  resolution = NULL,
  key = NULL,
  quiet = FALSE
)

get_dng(
  x = NULL,
  datasource = NULL,
  min_tree_height = 2,
  zoom = 17,
  radius = 800,
  min_area = 500,
  unit = c("m2", "ha", "km2"),
  year = NULL,
  network = NULL,
  overpass_url = "https://overpass-api.de/api/interpreter",
  timeout = 180,
  workers = NULL,
  quiet = FALSE
)
```

Arguments

x	sf. building footprint polygon, typically output from search_3dglobdf()
metrics	vector. A list of metrics to be computed. All metrics will be computed by default.
quiet	logical. If TRUE, suppress cli messages and progress bars.
radius	numeric. (only required for get_neighbors, get_gbvi, and get_dng) A numeric value specifying the buffer radius (in meters) used to define the proximity of a building footprint centroid. Default for get_neighbors, and get_gbvi and get_dng is respectfully 500 and 800.
datasource_canopy_height	character or NULL. Canopy height source for building the DSM and canopy green feature layer. Currently supports "metachm", "ethCHM", or NULL. If NULL, the DSM is built from buildings and DEM only.

datasource_greenspace
: character or NULL. Optional 2D greenspace map tile source for the visible-green feature layer. Supports "esri" and "sentinel2". If both canopy height and greenspace sources are supplied, the visible-green layer is the union of height-filtered canopy and 2D greenspace.

min_tree_height
: numeric. (only required for get_bgvi and get_dng) When datasource_canopy_height is a canopy height source, minimum height threshold (in meters) to classify vegetation as trees in the CHM. Default is 2.

zoom
: numeric. (only required for get_bgvi and get_dng) Zoom level of map tile when datasource_greenspace = "esri" or datasource_greenspace = "sentinel2". The default is 17. The higher level of zoom will lead to higher resolution of greenspace data for computing BGVI or DNG.

year
: numeric. The desired year for Sentinel-2 cloudless mosaic tiles. (This has to be specified when datasource_greenspace = "sentinel2")

floor
: logical. (only required for get_bgvi) Whether to compute Building Green View Index (BGVI) for each floor level based on estimated number of floors. Default is FALSE.

floor_step
: integer. (only required for get_bgvi when floor = TRUE) Compute GVI every floor_step floors. The top estimated floor is always included. Default is 1, meaning every floor.

short_building_threshold
: numeric. (only required for get_bgvi) Height threshold in meters for deciding whether a building should use only the bottom viewpoint in non-floor mode. Default is 6.

field_of_view
: numeric. (only required for directional get_bgvi) Angular field of view in degrees for direction-specific GVI. Used only when directions is not NULL. Default is 45.

directions
: character vector or NULL. Optional direction names for direction-specific GVI. Valid values are "southwest", "southeast", "northeast", "northwest", "north", "east", "west", and "south". If NULL, field_of_view is ignored and only non-directional GVI columns are returned.

workers
: integer. (only required for get_dng) Number of parallel workers for per-building distance computation. Defaults to one fewer than available cores. Use workers = 1 to run sequentially.

resolution
: numeric or NULL. (only used by get_bgvi) Output raster resolution in meters for the internal DSM used for viewshed computation. If NULL (default), the finest native resolution among the downloaded DEM and canopy height model is used, matching get_fused_dsm(). Set explicitly (e.g. 1) to force a finer grid.

key
: character. (only required for get_bgvi) API key of OpenTopography.

datasource
: character. (only required for get_bgvi and get_dng) Green/canopy data source. Supported values are "metachm", "esri", and "sentinel2" for get_dng.

min_area
: numeric. (only required for get_dng) The minimum area (the unit is defined by unit) of greenspace patches within the proximity of a building. For example, when min_area = 500, any greenspace patches with area less than 500 square meter will be excluded.

unit	character. (only required for get_dng) The unit for min_area: 'm2','ha', and 'km2'.
network	NULL, character, or sf. (only required for get_dng) Controls whether distances are measured along a real street network instead of in a straight line. Use NULL (the default) for straight-line distance, "osm" to download a walkable OpenStreetMap network for the study extent, or supply your own sf line layer of road/path centre lines. When routing is requested, the distance is the sum of the walk from the building centroid to the network, the shortest path along the network, and the walk from the network to the green-space pixel.
overpass_url	character. (only required for get_dng when network = "osm") Overpass API endpoint used to download the street network.
timeout	numeric. (only required for get_dng when network = "osm") Overpass query timeout in seconds. Default is 180.

Details

When floor = TRUE, get_bgvi() estimates the number of floors from building height and computes GVI from selected floor viewpoints. The floor_step argument controls how densely floors are sampled. If floor_step = 1, GVI is computed for every estimated floor. If floor_step > 1, GVI is computed every floor_step floors to reduce runtime; for example, floor_step = 3 samples floors 1, 4, 7, and so on. The top estimated floor is always included, even when it does not fall on the step sequence.

If directions is provided, get_bgvi() also computes direction-specific GVI by applying the requested field_of_view around each direction within each computed viewshed. The viewshed itself is computed once per viewpoint, then reused for each direction. Direction-specific columns are named with the direction suffix, such as gvi_bottom_south, gvi_top_south, mean_gvi_south, min_gvi_south, max_gvi_south, and sd_gvi_south.

Value

get_morphology returns an sf object identical to input x, with additional columns for:

- g_area: Ground area of footprint
- pmeter: Perimeter length of footprint
- v_surf: Vertical surface area (walls)
- t_surf: Total surface area (walls + roof)
- vol: Volume (area * height)
- obb_vol: Volume of oriented bounding box
- pa_ratio: Perimeter-area ratio
- rec: Rectangular compactness
- fra: Fractal dimension ratio
- cbn: Cuboidness index
- hem: Hemisphericality index
- cnv: Convexity index

- me_dist: Mean edge accessibility distance
- mp_dist: Mean pairwise distance within the footprint
- vol_exch: Volume exchange ratio
- elo_x/elo_y/elo_z: Elongation ratios (x/y/z axis)

get_neighbors returns an sf object identical to input x, with additional columns for:

- n_count: Number of adjacent buildings within the radius.
- m_ndist: Mean distance from the building centroid to neighboring building centroids.
- min_ndist: Minimum distance to neighboring building centroids.
- max_ndist: Maximum distance to neighboring building centroids.
- sd_ndist: Standard deviation of those distances.

get_bgvi returns an sf object identical to input x, with additional columns for:

- mean_gvi: Mean Building Green View Index value (0 to 1) from selected height(s).
- bottom_gvi: Building Green View Index from the bottom viewpoint, 1.7 m above ground.
- top_gvi: Building Green View Index from the top viewpoint. For short buildings, this is equal to bottom_gvi.
- bottom_green_area: Visible green area (m^2) from the bottom viewpoint.
- top_green_area: Visible green area (m^2) from the top viewpoint.
- mean_green_area: Mean visible green area (m^2) across viewpoints.
- min_gvi, max_gvi, sd_gvi: Minimum, maximum, and standard deviation of Green View Index (GVI) (if floor = TRUE).
- estimated_floors: Estimated number of floors based on building height (if floor = TRUE).

get_dng returns an sf object identical to input x, with additional columns: dng: Distance to nearest green space (tree canopy pixel) in meters, measured along the street network when network is supplied; and dng_method: either "network" or "euclidean", recording how each value was obtained. Buildings that cannot reach any green space through the network fall back to straight-line distance and are flagged "euclidean".

Note

x must include a unique id field. If x includes group_id, metrics that represent a whole building are computed on temporary grouped features and copied back to the original rows.

To request an OpenTopography API token, please visit: https://portal.opentopography.org/requestService?service=api

USGS 3DEP 1m/10m raster dataset is currently restricted to academic users. Academic users can request access to these data via the OpenTopography portal. Non-academic users can enquire about an enterprise API key by emailing info@opentopography.org. See OpenTopography Terms of Use for more information on appropriate use of the API.

References

Anna Labetski, Stelios Vitalis, Filip Biljecki, Ken Arroyo Ohori & Jantien Stoter (2023): 3D building metrics for urban morphology. International Journal of Geographical Information Science, 37(1): 36-67. DOI: 10.1080/13658816.2022.2103818

Rachid Hamaina, Thomas Leduc, Guillaume Moreau. Towards Urban Fabrics Characterization Based on Buildings Footprints. Jerome Gensel; Didier Josselin; Danny Vandenbroucke. Bridging the Geographic Information Sciences- International AGILE'2012 Conference, Avignon (France), April, 24-27, 2012, Springer Berlin Heidelberg, pp.327-346, 2012, 978-3-642-29062-6. 10.1007/978-3-642-290633_18. hal-01347299

Melchiorri, M., Freire, S., Schiavina, M. et al. The Multi-temporal and Multi-dimensional Global Urban Centre Database to Delineate and Analyse World Cities. Sci Data 11, 82 (2024). https://doi.org/10.1038/s41597-023-02691-1

Essential background in Pesaresi, M. et al. (2024) "Advances on the Global Human Settlement Layer by joint assessment of Earth Observation and population survey data", International Journal of Digital Earth, 17(1).

Qi, L., Hu, Y., Bu, R., Xiong, Z., Li, B., Zhang, C., ... & Li, C. (2024). Spatial-temporal patterns and influencing factors of the Building Green View Index: A new approach for quantifying 3D urban greenery visibility. Sustainable Cities and Society, 111, 105518.

Examples

```
library(gloBFPr)
data(globfp_example)
result <- gloBFPr::get_morphology(globfp_example[c(1:3),], quiet = TRUE)

result <- gloBFPr::get_neighbors(globfp_example[c(1:3),], radius = 100)
## Not run:
result <- gloBFPr::get_bgvi(globfp_example[c(1:3),],
                            datasource_canopy_height = "metachm",
                            datasource_greenspace = "esri",
                            key = "YOUR_opentopography_API_KEY")

## End(Not run)

result <- gloBFPr::get_dng(#globfp_example[c(1:3),],
                           datasource = "metachm",
                           unit = "m2")

# Measure along real road and path centre lines instead
result <- gloBFPr::get_dng(#globfp_example[c(1:3),],
                           datasource = "metachm",
                           unit = "m2",
                           network = "osm")
```

get_noise_map *Create a screening-level urban road-noise workflow object*

Description

Convenience wrapper around `prepare_noisemodelling_inputs()`. With `run = FALSE`, it returns reproducible input layers. With `run = TRUE`, it runs the official headless NoiseModelling WPS scripts and reads the `RECEIVERS_LEVEL` output table back into R. The returned `noise_map` element joins the calculated levels to receiver geometries for point mapping, and `isophones` contains NoiseModelling's `CONTOURING_NOISE_MAP` polygons.

Usage

```
get_noise_map(
  x = NULL,
  height_field = "Height",
  datasource_canopy_height = NULL,
  datasource_greenspace = NULL,
  greenspace_year = NULL,
  greenspace_zoom = 17,
  opentopo_key = NULL,
  roads = NULL,
  greenspace = NULL,
  canopy_height = NULL,
  dem = NULL,
  population = FALSE,
  population_field = NULL,
  population_year = 2025,
  min_tree_height = 2,
  receiver = c("grid", "none"),
  resolution = 10,
  out_dir = NULL,
  write = FALSE,
  run = FALSE,
  nm_path = NULL,
  nm_version = "5.0.1",
  download_nm = TRUE,
  osm_file = NULL,
  osm_remove_tunnels = TRUE,
  osm_eliminate_no_traffic_roads = TRUE,
  java = NULL,
  keep_files = FALSE,
  wall_alpha = 0.1,
  reflection_order = 0,
  max_src_distance = 150,
  max_reflection_distance = 50,
  thread_count = 0,
```

```
  diffraction_vertical = FALSE,
  diffraction_horizontal = FALSE,
  export_source_id = FALSE,
  humidity = NULL,
  temperature = NULL,
  favourable_occurrences = NULL,
  rays_name = NULL,
  max_error = NULL,
  frequency_field_prepend = "HZ",
  noise_wps_args = NULL,
  delaunay_max_area = NULL,
  road_width = 2,
  building_buffer = 2,
  iso_levels = c(35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 200),
  iso_field = "LAEQ",
  iso_smooth = 0.5,
  quiet = TRUE
)
```

Arguments

x	sf polygon object with building footprints and a height field.
height_field	Building height column name. Defaults to "Height".
datasource_canopy_height	Character or NULL. Canopy height source to retrieve internally when canopy_height is not supplied. Currently supports "metachm" and "ethCHM".
datasource_greenspace	character or NULL. Optional 2D greenspace map tile source for the visible-green feature layer. Supports "esri" and "sentinel2". If both canopy height and greenspace sources are supplied, the visible-green layer is the union of height-filtered canopy and 2D greenspace.
greenspace_year	numeric. The desired year for Sentinel-2 cloudless mosaic tiles. (This has to be specified when datasource_greenspace = "sentinel2")
greenspace_zoom	numeric. Zoom level of map tile when datasource_greenspace = "esri" or "sentinel2".
opentopo_key	OpenTopography API key used to retrieve DEM data internally when dem is not supplied.
roads	Optional sf line object, usually from OSM. If NULL, OSM roads are downloaded from the bounding box of x. If inferred traffic columns are absent, infer_osm_traffic() is applied.
greenspace	Optional sf polygons or terra::SpatRaster marking green areas. Green areas are translated to ground absorption G = 1.
canopy_height	Optional terra::SpatRaster canopy height map. Cells greater than or equal to min_tree_height are treated as green ground.
dem	Optional terra::SpatRaster terrain layer.

`population` Logical. If `TRUE`, assign GHSL population to buildings with the package population helper (or `get_pop()` when available) before writing NoiseModelling `BUILDINGS`. Existing `POP` or `population_field` values are used when available.

`population_field` Optional column containing building population. The value is copied to NoiseModelling's `POP` field.

`population_year` GHSL population year passed to the package population function when `population = TRUE` and no usable population field is already present.

`min_tree_height` Minimum canopy height treated as green cover.

`receiver` One of `"grid"` or `"none"`.

`resolution` Receiver grid resolution in map units.

`out_dir` Optional output directory for a GeoPackage.

`write` Logical. If `TRUE`, write layers to `noise_inputs.gpkg`.

`run` Logical. If `TRUE`, run NoiseModelling after preparing inputs.

`nm_path` Optional path to a `NoiseModelling_without_gui-*` directory. If `NULL`, the package uses `options(gloBFPr.noisemodelling.path)`, `NOISEMODELLING_HOME`, or a cached download.

`nm_version` NoiseModelling version to download when needed.

`download_nm` Logical. If `TRUE`, download the headless NoiseModelling runner when `nm_path` is not available.

`osm_file` Optional path to a local `.osm`, `.osm.gz`, or `.osm.pbf` extract. When supplied with `roads = NULL` and `run = TRUE`, NoiseModelling's `Import_OSM.groovy` creates the `ROADS` table internally using its OSM road defaults. This is different from the default R workflow, which downloads roads from the building bounding box.

`osm_remove_tunnels` Logical passed to NoiseModelling `Import_OSM` when `osm_file` is used. If `TRUE`, OSM roads tagged `tunnel=yes` are removed.

`osm_eliminate_no_traffic_roads` Logical passed to NoiseModelling `Import_OSM` when `osm_file` is used. If `TRUE`, keeps only road classes that NoiseModelling treats as traffic-bearing roads.

`java` Optional Java executable, Java home directory, or installed Java major version such as `17`. If `NULL`, the function uses `JAVA_HOME` or `java` on `PATH`. NoiseModelling 5.0.1 works with Java 11-21; Java 17 is recommended.

`keep_files` Logical. If `TRUE`, keep the temporary NoiseModelling work directory and include it in the result.

`wall_alpha` Wall absorption coefficient passed to NoiseModelling.

`reflection_order` Reflection order passed to NoiseModelling.

`max_src_distance` Maximum source-receiver distance in meters.

`max_reflection_distance`
: Maximum reflection distance in meters.

`thread_count`
: Number of NoiseModelling worker threads. `0` lets NoiseModelling choose.

`diffraction_vertical`
: Logical. Passes `confDiffVertical`; enables diffraction around vertical edges. NoiseModelling notes that CNOSSOS-EU uses this mainly for rail and industrial sources.

`diffraction_horizontal`
: Logical. Passes `confDiffHorizontal`; enables diffraction over horizontal building/terrain edges.

`export_source_id`
: Logical. Passes `confExportSourceId`; if `TRUE`, receiver levels are kept by source identifier instead of merged across all sources. This is useful for source-contribution diagnostics and can greatly enlarge the output.

`humidity`
: Relative humidity percentage for atmospheric absorption. Passes `confHumidity`. Defaults to NoiseModelling's script default of `70`.

`temperature`
: Air temperature in degrees Celsius for atmospheric absorption. Passes `confTemperature`. Defaults to NoiseModelling's script default of `15`.

`favourable_occurrences`
: Probability of favourable propagation conditions by 16 wind-direction sectors, clockwise, where the north sector is the last value. Supply one value to recycle to all sectors or 16 values. The NoiseModelling default is sixteen `0.5` values.

`rays_name`
: Optional table name or file URL passed as `confRaysName`. When supplied, NoiseModelling exports propagation rays/attenuation details for advanced diagnostics. This can create very large outputs.

`max_error`
: Maximum allowed error in dB for pruning negligible source contributions. Passes `confMaxError`; NoiseModelling's default is `0.1`.

`frequency_field_prepend`
: Prefix for source spectral columns, passed as `frequencyFieldPrepend`. Defaults to `"HZ"` for columns such as `HZ1000`.

`noise_wps_args`
: Optional named list of additional raw arguments passed to `Noise_level_from_source.groovy`. Use this only for advanced NoiseModelling options not yet exposed directly.

`delaunay_max_area`
: Maximum Delaunay triangle area in square map units. Defaults to `resolution^2`. Smaller values create denser receiver/contour meshes and slower runs.

`road_width`
: Receiver exclusion distance around roads in meters for the NoiseModelling Delaunay grid.

`building_buffer`
: Receiver exclusion distance around buildings in meters for the NoiseModelling Delaunay grid.

`iso_levels`
: Numeric vector of isosurface breakpoints in dB passed to `Create_Isosurface.groovy`.

`iso_field`
: Result field used to build isosurfaces. Defaults to `"LAEQ"`.

`iso_smooth`
: Smoothing coefficient passed to `Create_Isosurface.groovy`.

`quiet`
: Logical. If `TRUE`, suppress informational messages.

Value

Prepared noise input layers when `run = FALSE`; otherwise a list with prepared inputs, receiver-level results, output file paths, and logs.

`get_shadows` *Building shadow and radiation calculations*

Description

`svf()` computes a Sky View Factor raster from building and optional canopy obstacles.

`get_shadow_footprint()` computes ground shadow footprints for extruded building polygons.

`get_shadow_height()` computes shadow height at points or across a `terra` surface. If `shadow_locations` is omitted, a `terra` template is generated around the buildings.

`get_radiation()` estimates direct, diffuse, and total radiation load on roofs and facades represented by a 3D `sf` surface grid.

Usage

```
svf(
  x = NULL,
  height_field = "Height",
  min_tree_height = 2,
  datasource_canopy_height = NULL,
  key = NULL,
  canopy_height = NULL,
  dem = NULL,
  raster_buffer = NULL,
  grid_res = 2,
  extent_buffer = NULL,
  res_angle = 5,
  observer_height = 1.7,
  max_distance = NULL,
  plot = FALSE,
  scalebar = TRUE,
  scalebar_unit = c("auto", "km", "m"),
  scalebar_cex = 0.7,
  north_arrow = TRUE,
  quiet = TRUE
)

get_shadow_footprint(
  x = NULL,
  solar_time = NULL,
  time_zone = NULL,
  azimuth = NULL,
```

```
  elevation = NULL,
  height_field = "Height",
  min_tree_height = 2,
  datasource_canopy_height = NULL,
  key = NULL,
  canopy_height = NULL,
  dem = NULL,
  raster_buffer = NULL,
  b = 0.01,
  overlap_shadow = FALSE,
  plot = FALSE,
  plot_overlap_gradient = FALSE,
  scalebar = TRUE,
  scalebar_unit = c("auto", "km", "m"),
  scalebar_cex = 0.7,
  north_arrow = TRUE,
  quiet = TRUE
)

get_shadow_height(
  x = NULL,
  shadow_locations = NULL,
  solar_time = NULL,
  time_zone = NULL,
  azimuth = NULL,
  elevation = NULL,
  height_field = "Height",
  min_tree_height = 2,
  datasource_canopy_height = NULL,
  key = NULL,
  raster_buffer = NULL,
  canopy_height = NULL,
  dem = NULL,
  cell_size = 2,
  extent_buffer = NULL,
  b = 0.01,
  filter_footprint = FALSE,
  quiet = TRUE
)

get_radiation(
  x = NULL,
  grid = NULL,
  solar_time = NULL,
  time_zone = NULL,
  azimuth = NULL,
  elevation = NULL,
  solar_normal,
```

```
  solar_diffuse,
  height_field = "Height",
  min_tree_height = 2,
  datasource_canopy_height = NULL,
  key = NULL,
  raster_buffer = NULL,
  canopy_transmissivity = 0.15,
  canopy_height = NULL,
  dem = NULL,
  grid_res = 2,
  ground = FALSE,
  ground_res = NULL,
  offset = 0.01,
  radius = 500,
  svf_res_angle = 15,
  return_list = FALSE,
  plot = FALSE,
  plot_3d = FALSE,
  scalebar = TRUE,
  scalebar_unit = c("auto", "km", "m"),
  scalebar_cex = 0.7,
  north_arrow = TRUE,
  quiet = TRUE
)
```

Arguments

x	An sf polygon object with building footprints and a height field.
height_field	Character. Name of the building height column. Defaults to "Height", matching search_3dglobdf() output.
min_tree_height	Numeric. Minimum canopy height, in map units, used as a tree obstacle.
datasource_canopy_height	Character or NULL. Canopy height source to retrieve internally when canopy_height is not supplied. Currently supports "metachm", "ethCHM", or NULL.
key	Character or NULL. OpenTopography API key used to retrieve DEM internally when dem is not supplied.
canopy_height	Optional terra::SpatRaster canopy height map. Values are interpreted as height above ground.
dem	Optional terra::SpatRaster digital elevation model. When supplied, canopy and building shadows are compared in absolute elevation and shadow-height outputs are returned above local ground.
raster_buffer	Numeric or NULL. Buffer distance in CRS units around buildings used when retrieving CHM/DEM internally. If NULL, a buffer is estimated from building height and solar elevation.
grid_res	Numeric surface-grid resolution in CRS units.

`extent_buffer` Optional numeric buffer around `x` used when creating an automatic `terra::SpatRaster` template. If omitted, a buffer is estimated from building heights and solar elevation.

`res_angle` Numeric. Azimuth sampling interval in decimal degrees for `svf()`. Smaller values are slower and more detailed.

`observer_height` Numeric. Height above local ground for SVF query locations. Defaults to `1.7`, representing pedestrian eye level.

`max_distance` Numeric. Maximum obstacle search distance in CRS units for `svf()`.

`plot` Logical. For `get_shadow_footprint()`, draw a base R map of the building footprints and shadow polygons before returning the `sf` result. For `get_radiation()`, draw the default 2D base R radiation map colored by `total`. When ground samples are included, the 2D layout shows separate ground, facade, and roof maps with one shared legend. When canopy data are supplied, a second 2D map shows canopy impact as `canopy - no_canopy` total-radiation difference.

`scalebar` Logical. Draw a distance scale bar using the shared map layout. Defaults to `TRUE` for plotted maps.

`scalebar_unit` Scale bar unit: `"auto"` (default), `"km"`, or `"m"`.

`scalebar_cex` Scale bar label size. Defaults to `0.7`.

`north_arrow` Logical. Draw a north arrow in the map panel. Defaults to `TRUE`.

`quiet` Logical. If `FALSE`, emit progress messages.

`solar_time` Character vector or list of character strings. Local solar times such as `"2026-06-21 15:00:00"`. If `solar_time` and `time_zone` are supplied, `azimuth` and `elevation` are ignored and solar position is estimated from time and the building-layer centroid.

`time_zone` Character. A single time zone used to interpret `solar_time`, for example `"America/Denver"` or `"UTC"`.

`azimuth` Numeric vector or list. Solar azimuth in decimal degrees, measured clockwise from north. Must have the same length as `elevation`.

`elevation` Numeric vector or list. Solar elevation in decimal degrees above the horizon. Must have the same length as `azimuth`.

`b` Numeric buffer tolerance used when cleaning footprint unions.

`overlap_shadow` Logical. For `get_shadow_footprint()`, if `TRUE`, dissolve overlapping shadows across all supplied solar positions by shadow source.

`plot_overlap_gradient` Logical. For `get_shadow_footprint()` plots with multiple `solar_time` values, if `TRUE`, draw all shadows in transparent gray so overlapping areas appear darker.

`shadow_locations` Optional query locations for shadow height, as an `sf` point layer or a `terra::SpatRaster`.

`cell_size` Numeric cell resolution in CRS units when `shadow_locations` is omitted.

`filter_footprint` Ignored. Shadow footprints are always used to limit height calculations.

grid	Optional 3D sf point surface grid. If omitted, it is created from building roofs and facades (and optionally the ground).
solar_normal	Direct Normal Irradiance vector, one value per solar position.
solar_diffuse	Diffuse Horizontal Irradiance vector, one value per solar position.
canopy_transmissivity	Numeric from 0 to 1. Fraction of direct irradiance transmitted through canopy shadows in get_radiation().
ground	Logical. If TRUE, add a regular grid of ground-level sample points over the study-area bounding box (excluding building footprints). Ground points have an upward normal and receive direct radiation whenever they are not in a building or canopy shadow, and diffuse radiation scaled by their Sky View Factor. Returned rows have surface = "ground" and building_id = NA.
ground_res	Numeric resolution for the ground sample grid in CRS units. If NULL, defaults to grid_res.
offset	Numeric vertical offset added to generated surface-grid points.
radius	Maximum obstacle search distance in CRS units for radiation Sky View Factor calculations. Defaults to 500. Obstacles beyond this distance contribute negligibly to Sky View Factor but dominate runtime, so a finite radius enables spatial culling and is typically many times faster. Use Inf to consider all obstacles regardless of distance.
svf_res_angle	Numeric. Azimuth sampling interval in decimal degrees used when estimating Sky View Factor inside get_radiation().
return_list	Logical. If TRUE, return per-timestep radiation matrices instead of a summed sf surface grid.
plot_3d	Logical. For get_radiation(), draw a base R 3D-style view with separate panels for direct, diffuse, and total radiation. This is opt-in; plot = TRUE uses the 2D map layout by default.

Details

These functions are implemented directly with sf and terra using a projected 2.5D building model.

Value

get_shadow_footprint() returns an sf polygon layer.

get_shadow_height() returns a terra::SpatRaster for terra locations or a numeric matrix for point locations.

get_radiation() returns an sf point layer with svf, direct, diffuse, and total columns, unless return_list = TRUE.

svf() returns a terra::SpatRaster with Sky View Factor values from 0 to 1.

References

Dorman, M. et al. shadow: Geometric Shadow Calculations. https://github.com/michaeldorman/shadow

globfp_example *Test 3D-GloBFP dataset*

Description

A sample dataset containing simplified 3D building footprint information for demonstration and testing purposes.

Usage

```
globfp_example
```

Format

A data frame with 369 rows and 3 variables:

id Numeric. Unique identifier for each building.

Height Numeric. Estimated height of the building in meters.

geometry sfc_POLYGON. The building footprint geometry in simple feature (sf) format.

Source

Che Yangzi, Li Xuecao, Liu Xiaoping, Wang Yuhao, Liao Weilin, Zheng Xianwei, Zhang Xucai, Xu Xiaocong, Shi Qian, Zhu Jiajun, Zhang Honghui, Yuan Hua, & Dai Yongjiu (2024). 3D-GloBFP: the first global three-dimensional building footprint dataset. Earth Syst. Sci. Data, 16, 5357-5374

Examples

```
data(globfp_example)
head(globfp_example)
```

globfp_example_canopy_height *Example canopy height raster for the 3D-GloBFP sample*

Description

A `terra::PackedSpatRaster` canopy height map cropped to the bounding box of `globfp_example` and aggregated from the source resolution for lightweight examples. Convert it with `terra::rast()` before analysis.

Usage

```
globfp_example_canopy_height
```

Format

A `terra::PackedSpatRaster` with one layer named `canopy_height`.

Source

metaCHM canopy height data via dsmSearch, downloaded for the `globfp_example` extent.

Examples

```
data(globfp_example_canopy_height)
canopy_height <- terra::rast(globfp_example_canopy_height)
canopy_height
```

---

globfp_example_dem *Example DEM for the 3D-GloBFP sample*

Description

A `terra::PackedSpatRaster` digital elevation model cropped to the bounding box of `globfp_example`. Convert it with `terra::rast()` before analysis.

Usage

```
globfp_example_dem
```

Format

A `terra::PackedSpatRaster` with one layer named `dem`.

Source

OpenTopography / dsmSearch elevation data, downloaded for the `globfp_example` extent.

Examples

```
data(globfp_example_dem)
dem <- terra::rast(globfp_example_dem)
dem
```

`infer_osm_traffic` *Infer screening-level road traffic inputs from OSM road attributes*

Description

Adds traffic volume and speed columns to an OSM road `sf` object based on the `highway` class and optional `maxspeed`, `lanes`, or `oneway` values. These values are intended for relative/scenario noise mapping when measured traffic counts are unavailable, not calibrated regulatory noise maps.

Usage

```
infer_osm_traffic(
  roads,
  defaults = osm_noise_traffic_defaults(),
  use_maxspeed = FALSE,
  use_lanes = FALSE,
  use_oneway = FALSE,
  quiet = TRUE
)
```

Arguments

`roads`	`sf` line object containing at least a `highway` column.
`defaults`	Data frame of road-class assumptions. Defaults to `osm_noise_traffic_defaults()`.
`use_maxspeed`	Logical. If `TRUE`, parse numeric speeds from `maxspeed` and use them when available. Defaults to `FALSE` to match NoiseModelling's OSM import defaults.
`use_lanes`	Logical. If `TRUE`, scale vehicle counts by lane count relative to a two-lane road. Defaults to `FALSE` to match NoiseModelling's OSM import defaults.
`use_oneway`	Logical. If `TRUE`, halve traffic counts on OSM one-way roads. Defaults to `FALSE`.
`quiet`	Logical. If `TRUE`, suppress informational messages.

Value

An `sf` object with inferred traffic columns, including `speed_kmh`, `light_veh_h`, `heavy_veh_h`, and NoiseModelling-style `LV_*`, `MV_*`, `HGV_*`, `WAV_*`, `WBV_*`, and speed columns for day/evening/night.

install_noisemodelling *Install the headless NoiseModelling runner*

Description

Downloads and unzips the official headless NoiseModelling release into an R user cache directory. This is used by `get_noise_map()` when `run = TRUE` and no `nm_path` is supplied.

Usage

```
install_noisemodelling(
  version = "5.0.1",
  destdir = noisemodelling_cache_dir(),
  quiet = TRUE
)
```

Arguments

`version`	NoiseModelling release version. Defaults to `"5.0.1"`.
`destdir`	Destination directory. Defaults to the `gloBFPr` user cache.
`quiet`	Logical. If `TRUE`, suppress download messages.

Value

Path to the installed NoiseModelling directory.

---

osm_noise_traffic_defaults *Default OSM traffic assumptions for screening-level road noise*

Description

Returns the built-in lookup table used by `infer_osm_traffic()` when observed traffic counts are unavailable. The defaults mirror the road-category assumptions embedded in NoiseModelling's `Import_OSM.groovy`, which cites the Good Practice Guide for Strategic Noise Mapping and the Production of Associated Data on Noise Exposure, version 2. They should still be replaced with local traffic counts whenever available.

Usage

```
osm_noise_traffic_defaults()
```

Value

A data frame with OSM highway class, speed, and hourly light/heavy vehicle assumptions.

plot_bgvi_viewshed *Visualize an Individual Building BGVI Viewshed*

Description

Computes and optionally plots the BGVI viewshed for one building and one viewpoint height. The function uses the same DSM, green feature layer, target footprint flattening, and flat-roof handling as `get_bgvi()`, but returns the diagnostic layers for a single building instead of summary columns for every building.

Usage

```
plot_bgvi_viewshed(
  x,
  building = 1,
  level = c("bottom", "top"),
  floor = NULL,
  height = NULL,
  orientation = NULL,
  field_of_view = 45,
  datasource_canopy_height = "metachm",
  datasource_greenspace = NULL,
  min_tree_height = 2,
  zoom = 17,
  radius = 800,
  year = NULL,
  resolution = NULL,
  key = NULL,
  plot = TRUE,
  scalebar = TRUE,
  scalebar_unit = c("auto", "km", "m"),
  scalebar_cex = 0.7,
  north_arrow = TRUE,
  quiet = FALSE,
  ...
)
```

Arguments

`x`	sf. Building footprint polygons, typically output from `search_3dglobdf()`. Must include a `Height` column.
`building`	Integer row number, or a value from the `id` column when `id` is present.
`level`	Character. One of `"bottom"` or `"top"`. Ignored when `floor` or `height` is supplied.
`floor`	Integer floor number to visualize. Floor 1 is 1.7 m above ground; higher floors add 3 m each.

height	Numeric observer offset above ground in metres. Overrides level and floor when supplied.
orientation	Optional sector orientation. Supply a bearing in degrees clockwise from north, or one of "north", "northeast", "east", "southeast", "south", "southwest", "west", or "northwest". If NULL, the full viewshed is used.
field_of_view	Numeric angular width in degrees for orientation.
datasource_canopy_height, datasource_greenspace, min_tree_height, zoom, radius, year, resolution, key, quiet	Passed to the BGVI raster preparation workflow; see get_bgvi().
plot	Logical. If TRUE, draw the viewshed map.
scalebar	Logical. If TRUE, add a scale bar using the package's shared map layout.
scalebar_unit	Character. Unit for the scale bar: "auto", "km", or "m".
scalebar_cex	Numeric text size for the scale bar and north arrow.
north_arrow	Logical. If TRUE, add a north arrow.
...	Additional arguments passed to the initial viewshed plot().

Value

A list containing the selected building, viewpoint, observer height, gvi, green_area, viewshed, viewshed_raster, viewshed_area, radius, visible_green, sector_mask, plot_raster, dsm, and binary_green.

Examples

```
## Not run:
result <- plot_bgvi_viewshed(
  globfp_example,
  building = 1,
  level = "top",
  orientation = "south",
  field_of_view = 60,
  datasource_canopy_height = "metachm",
  key = "YOUR_opentopography_API_KEY"
)

## End(Not run)
```

---

plot_foam_map *Plot an OpenFOAM pedestrian-level map*

Description

Convenience wrapper around ggplot2 for visualising a single layer from the raster returned by sample_foam_slice or read_foam_pedestrian_slice.

Usage

```
plot_foam_map(
  r,
  layer = "U_mag",
  title = NULL,
  palette = "YlOrRd",
  reverse = FALSE,
  buildings = NULL,
  canopy = NULL,
  legend_title = layer,
  max_u_ref = NULL,
  na_colour = "black",
  scalebar = TRUE,
  scalebar_unit = c("auto", "km", "m"),
  scalebar_cex = 0.7,
  north_arrow = TRUE
)
```

Arguments

r	A SpatRaster (single layer, or one layer will be selected via layer).
layer	Character. Layer name to plot. Default "U_mag".
title	Character. Plot title. Default auto-generated.
palette	Character. hcl.colors palette name. Default "YlOrRd".
reverse	Logical. Reverse palette direction. Default FALSE.
buildings	Optional sf object of building footprints to overlay (in the same local coordinate system as the raster).
canopy	Optional data frame of canopy cell centres (x, y) with a "res" attribute giving the cell size. Taken from the raster's "canopy" attribute when not supplied, which read_foam_pedestrian_slice attaches whenever the case has a canopy height raster.
legend_title	Character. Legend label. Default layer.
max_u_ref	Numeric. If plotting U_mag, annotate the colour scale as a wind speed ratio by dividing by this reference speed. Default NULL (no ratio).
na_colour	Colour for cells with no result. Inside the mapped area a void is an obstruction the flow never entered, so the default is the same solid black used for buildings and canopy.
scalebar	Logical. Draw a distance scale bar in the lower-left corner. Defaults to TRUE. Distances assume projected map units are metres; for geographic rasters they are approximated at the map's mid-latitude.
scalebar_unit	Scale bar unit: "km", "m", or "auto" to pick whichever keeps the label readable. Default "auto".
scalebar_cex	Scale bar label size multiplier. Defaults to 0.7.
north_arrow	Logical. Draw a north arrow above the scale bar. Defaults to TRUE.

Value

A ggplot object.

---

plot_noise_map *Plot a NoiseModelling-style road-noise map*

Description

Draws NoiseModelling isosurface polygons underneath roads and buildings when available. If x does not contain official isosurfaces, it falls back to an interpolated receiver surface.

Usage

```
plot_noise_map(
  x,
  period = "DEN",
  field = "LAEQ",
  resolution = NULL,
  breaks = c(35, 40, 45, 50, 55, 60, 65, 70, 75),
  nodata = -99,
  palette = noise_map_palette(),
  road_col = "white",
  road_alpha = 0.35,
  road_lwd = 1.4,
  building_col = "black",
  legend = TRUE,
  legend_width = 0.26,
  legend_cex = 0.85,
  scalebar = TRUE,
  scalebar_unit = c("auto", "km", "m"),
  scalebar_cex = 0.7,
  north_arrow = TRUE,
  mar = c(0.2, 0.2, 0.2, 0.2),
  add = FALSE,
  ...
)
```

Arguments

x	A result from get_noise_map(run = TRUE), a gloBFPr_noise_surface object, or an sf receiver noise map.
period	Noise period to plot. Defaults to "DEN".
field	Noise field to plot. Defaults to "LAEQ".
resolution	Optional raster resolution in map units when x is an sf receiver map.
breaks	Noise class breakpoints in dB.

nodata	Values at or below this threshold are treated as no-data. Defaults to -99, NoiseModelling's no-result sentinel value.
palette	Fill colors from quiet to loud.
road_col	Road overlay color. Defaults to white with transparency. Use NA to omit roads.
road_alpha	Road overlay alpha from 0 fully transparent to 1 opaque.
road_lwd	Road overlay line width.
building_col	Building fill color. Use NA to omit buildings.
legend	Logical. Draw the dB(A) class legend in a dedicated panel to the right of the map. Defaults to TRUE.
legend_width	Width of the legend panel relative to the map panel. Smaller values give the map more room. Defaults to 0.26.
legend_cex	Legend text size. Defaults to 0.85.
scalebar	Logical. Draw a distance scale bar just below the legend (or in the bottom-left of the map when legend = FALSE). Defaults to TRUE. Distances assume a projected CRS in metres; for geographic coordinates they are approximated at the map's mid-latitude.
scalebar_unit	Scale bar unit: "auto" (default), "km", or "m" to pick whichever keeps the label readable.
scalebar_cex	Scale bar label size. Defaults to 0.7.
north_arrow	Logical. Draw a north arrow in the lower-left map area. Defaults to TRUE.
mar	Margins (in lines) around the map panel. Defaults to a tight margin so the map fills the device.
add	Logical. If TRUE, add to the current plot.
...	Additional arguments passed to plot().

Value

Invisibly returns the isosurface sf object or fallback gloBFPr_noise_surface object used for plotting.

`prepare_foam_case` *Prepare a transient OpenFOAM case for urban wind*

Description

Generates a complete OpenFOAM wind case using buoyantBoussinesqPimpleFoam - a transient, Boussinesq, finite-volume Navier-Stokes solve with a kOmegaSST (URANS) closure - driven by an ABL log-law inlet.

The solver is a buoyant one running an isothermal problem: every surface sits at T_ref and the walls are adiabatic, so the Boussinesq body force is identically zero and the result is pure mechanical flow.

Terrain and canopy are used when supplied and skipped when not. Terrain becomes solid geometry; canopy becomes distributed drag via atmPlantCanopy* rather than solid blocks, because a crown passes and drags air instead of blocking it.

Usage

```
prepare_foam_case(
  case_dir,
  stl_file,
  domain,
  inlet_velocity = c(5, 0, 0),
  z_ref = 10,
  T_ref = 295,
  sim_hours = NULL,
  n_writes = 8L,
  terrain_stl = NULL,
  terrain_dem = NULL,
  canopy_stl = NULL,
  leaf_area_density = 0.4,
  canopy_heat_source = NULL,
  plant_cd = 0.2,
  z0 = 0.1,
  base_cell_size = 10,
  building_refinement = 2L,
  terrain_refinement = 0L,
  max_cells = 3000000L,
  max_co = NULL,
  n_outer_correctors = 2L,
  overwrite = FALSE,
  quiet = FALSE
)
```

Arguments

case_dir	Character. Case directory; must already contain constant/triSurface/<stl_file>.
stl_file	Character. Building STL path (host path).
domain	Named list with xmin/xmax/ymin/ymax/zmin/zmax (metres, local coordinates), from prepare_openfoam_inputs()$domain.
inlet_velocity	Numeric length-3 (Ux, Uy, Uz) in m/s at z_ref. Any horizontal direction is supported.
z_ref	Numeric. Reference height for inlet_velocity. Default 10.
T_ref	Numeric. Reference air temperature (K). Default 295. Every surface is held at this value, which is what switches buoyancy off.
sim_hours	Numeric. Physical hours to simulate. NULL (the default) computes three flow-through times from the domain length and the inlet speed, which is what a wind case needs to flush its transient.
n_writes	Integer. Output times over the run. Default 8.
terrain_stl	Path to a terrain STL (see prepare_foam_geometry), or NULL for a flat floor.
terrain_dem	Optional SpatRaster / path of the bare-earth DEM. Used to set the domain floor. Strongly recommended whenever terrain_stl is supplied.

`canopy_stl` Path to a canopy volume STL, or `NULL`.

`leaf_area_density`
: Numeric. LAD (1/m) inside the canopy. Default 0.4.

`canopy_heat_source`
: Numeric or `NULL`. Enables `atmPlantCanopyTSource` and writes `0/qPlant` with this uniform value. `NULL` (default) leaves the source out entirely.

  Off by default deliberately: `qPlant` is a canopy energy flux that cannot be derived from a canopy height model, and enabling the source with `qPlant = 0` would be inert while still adding a way for the run to abort. Check the units against the `atmPlantCanopyTSource` documentation for your OpenFOAM version before relying on a value.

`plant_cd` Numeric. Canopy drag coefficient. Default 0.2.

`z0` Numeric. Aerodynamic roughness length (m). Default 0.1.

`base_cell_size` Numeric. Background cell size (m). Default 10.

`building_refinement`
: Integer. snappyHexMesh level for buildings. Default 2.

`terrain_refinement`
: Integer. snappyHexMesh level for terrain. Default 0 (the background cell size already resolves gentle slope).

`max_cells` Integer. Global cell-count cap. Default 3e6.

`max_co` Numeric. Maximum Courant number. `NULL` (the default) gives 20: PIMPLE re-converges momentum and pressure within each step, so Courant well above 1 is stable when marching to a steady field.

`n_outer_correctors`
: Integer. PIMPLE outer correctors. Default 2.

`overwrite`, `quiet`
: Logical.

Value

Invisibly, a list with `case_dir`, `files` and `params`.

Oblique wind

`blockMesh` emits four separately named lateral patches (`xMin`, `xMax`, `yMin`, `yMax`) and each is assigned an inlet / outlet / lateral role from the sign of `dot(flowDir, outward_normal)`. Any wind direction works without rotating the domain - unlike the previous generator, which pinned the inlet to the x-min face so that a north wind injected velocity through a face whose normal was -x and effectively nothing entered.

See Also

`prepare_openfoam_inputs`, `prepare_foam_geometry`, `read_foam_pedestrian_slice`

`prepare_foam_geometry` *Build terrain, canopy and terrain-based building geometry for a case*

Description

Turns the rasters written by `prepare_openfoam_inputs` into the STL geometry `prepare_foam_case` needs. Everything is optional: supply what you have and the rest is skipped.

Terrain is recovered from the fused DSM rather than used directly, because the fused DSM is terrain + buildings + canopy and meshing it as ground under a separate building STL would double-count every building. Given how `get_fused_dsm()` builds it, bare earth comes back exactly as `fused_dsm - max(building_height, canopy_height, 0)`.

Usage

```
prepare_foam_geometry(
  case_dir,
  fused_dsm = NULL,
  building_height = NULL,
  canopy_height = NULL,
  buildings = NULL,
  height_col = NULL,
  domain = NULL,
  terrain_res = NULL,
  canopy_res = 5,
  min_tree_height = 2,
  base_cell_size = 10,
  skirt = 30,
  quiet = FALSE
)
```

Arguments

`case_dir`	Character. OpenFOAM case directory.
`fused_dsm, building_height, canopy_height`	SpatRaster or path or NULL. Typically `foam_inputs$files$fused_dsm`, `...$building_height_raste ...$canopy_height_raster`.
`buildings`	sf polygons in local coordinates, or NULL to leave the existing building STL alone. Usually `readRDS(foam_inputs$files$buildings_rds)`.
`height_col`	Character. Building height column.
`domain`	Named list with xmin/xmax/ymin/ymax/zmin/zmax; used to crop.
`terrain_res`	Numeric. Resolution (m) to resample terrain to before triangulating. One vertex per cell, so a 1 m DEM over 1 km2 is a million vertices; default `max(5, base_cell_size / 2)`.
`canopy_res`	Numeric. Resolution (m) for the canopy box cloud. Default 5.

`min_tree_height`
: Numeric. Canopy cells below this are ignored. Default 2.

`base_cell_size`
: Numeric. Background mesh cell size, used only to pick `terrain_res`. Default 10.

`skirt`
: Numeric. Metres the terrain solid extends below its minimum. Default 30.

`quiet`
: Logical.

Value

Invisibly, a list with `terrain_stl`, `canopy_stl`, `building_stl`, `dem` (SpatRaster) and `dem_file` - feed these straight to `prepare_foam_case`.

`prepare_noisemodelling_inputs`
Prepare NoiseModelling-style input layers

Description

Builds a compact set of layers for an integrated or external NoiseModelling workflow: `BUILDINGS`, `ROADS`, `GROUND`, `RECEIVERS`, and optional `DEM`. The returned layers can be inspected directly in R and optionally written to a GeoPackage.

Usage

```
prepare_noisemodelling_inputs(
  x = NULL,
  height_field = "Height",
  datasource_canopy_height = NULL,
  datasource_greenspace = NULL,
  greenspace_year = NULL,
  greenspace_zoom = 17,
  opentopo_key = NULL,
  canopy_height = NULL,
  roads = NULL,
  download_roads = TRUE,
  greenspace = NULL,
  dem = NULL,
  population = FALSE,
  population_field = NULL,
  population_year = 2025,
  min_tree_height = 2,
  receiver = c("grid", "none"),
  resolution = 10,
  ground_default = 0,
  green_ground = 1,
  out_dir = NULL,
```

```
  write = FALSE,
  quiet = TRUE
)
```

Arguments

x	sf polygon object with building footprints and a height field.
height_field	Building height column name. Defaults to "Height".
datasource_canopy_height	Character or NULL. Canopy height source to retrieve internally when canopy_height is not supplied. Currently supports "metachm" and "ethCHM".
datasource_greenspace	character or NULL. Optional 2D greenspace map tile source for the visible-green feature layer. Supports "esri" and "sentinel2". If both canopy height and greenspace sources are supplied, the visible-green layer is the union of height-filtered canopy and 2D greenspace.
greenspace_year	numeric. The desired year for Sentinel-2 cloudless mosaic tiles. (This has to be specified when datasource_greenspace = "sentinel2")
greenspace_zoom	numeric. Zoom level of map tile when datasource_greenspace = "esri" or "sentinel2".
opentopo_key	OpenTopography API key used to retrieve DEM data internally when dem is not supplied.
canopy_height	Optional terra::SpatRaster canopy height map. Cells greater than or equal to min_tree_height are treated as green ground.
roads	Optional sf line object, usually from OSM. If NULL, OSM roads are downloaded from the bounding box of x. If inferred traffic columns are absent, infer_osm_traffic() is applied.
download_roads	Logical. If TRUE and roads = NULL, download roads from OSM using the building extent. Set to FALSE only when a later workflow step supplies roads, for example get_noise_map(osm_file = ...).
greenspace	Optional sf polygons or terra::SpatRaster marking green areas. Green areas are translated to ground absorption G = 1.
dem	Optional terra::SpatRaster terrain layer.
population	Logical. If TRUE, assign GHSL population to buildings with the package population helper (or get_pop() when available) before writing NoiseModelling BUILDINGS. Existing POP or population_field values are used when available.
population_field	Optional column containing building population. The value is copied to NoiseModelling's POP field.
population_year	GHSL population year passed to the package population function when population = TRUE and no usable population field is already present.

min_tree_height	Minimum canopy height treated as green cover.
receiver	One of "grid" or "none".
resolution	Receiver grid resolution in map units.
ground_default	Default ground absorption for the analysis extent.
green_ground	Ground absorption assigned to greenspace/canopy polygons.
out_dir	Optional output directory for a GeoPackage.
write	Logical. If TRUE, write layers to noise_inputs.gpkg.
quiet	Logical. If TRUE, suppress informational messages.

Value

A list with prepared sf/terra layers and optional GeoPackage path.

---

prepare_openfoam_inputs

Prepare gloBFPr spatial outputs for an OpenFOAM Docker workflow

---

Description

Collects building, terrain, canopy, and ground-cover data from gloBFPr and writes them into a structured OpenFOAM case folder ready for external CFD simulation (via Docker).

Output layers and their intended role in OpenFOAM:

building STL Solid geometry for snappyHexMesh

building height raster Auxiliary mesh-generation reference

binary building raster Building mask / validation

fused DSM Optional ground STL source (terrain + buildings + canopy surface)

canopy height raster Standalone CHM for defining porous-zone extents and drag coefficients in fvOptions / topoSetDict

ground roughness raster (z0) Per-cell aerodynamic roughness length derived from ESA WorldCover land cover, for nutURoughWallFunction. Building footprint cells and tree-cover cells are set to NA because they are handled by solid geometry and porous zones respectively.

Usage

```
prepare_openfoam_inputs(
  case_dir,
  bbox = NULL,
  place = NULL,
  buildings_list = NULL,
  data_source = "GBF",
  cell_size = 1,
  crop = FALSE,
```

```
  mask = TRUE,
  include_buildings = TRUE,
  include_fused_dsm = TRUE,
  include_tree_canopy = TRUE,
  canopy_source = "metachm",
  min_tree_height = 2,
  include_morphology = TRUE,
  include_neighbors = TRUE,
  include_greenspace = TRUE,
  mask_tree_cover = TRUE,
  landcover_source = c("esa", "esri"),
  landcover_year = 2021,
  include_bgvi = FALSE,
  include_shadow = FALSE,
  include_radiation = FALSE,
  opentopo_key = NULL,
  target_crs = NULL,
  height_col = NULL,
  default_height = 10,
  min_height = 2,
  domain_buffer = 100,
  zmax_buffer = 50,
  stl_name = "buildings.stl",
  overwrite = FALSE,
  quiet = FALSE
)
```

Arguments

case_dir	Character. OpenFOAM case directory.
bbox	Numeric vector c(xmin, ymin, xmax, ymax) in WGS-84 lon/lat.
place	Optional place name passed to search_3dglobdf().
buildings_list	Optional precomputed output from search_3dglobdf(..., out_type = "all").
data_source	Character. Building source passed to search_3dglobdf(). Default "GBF".
cell_size	Numeric. Raster resolution in metres for building rasters.
crop	Logical. Whether to crop buildings to bbox.
mask	Logical. Whether to mask height raster by building footprints.
include_buildings	Logical. Prepare building vector/raster/STL.
include_fused_dsm	Logical. Prepare fused DSM.
include_tree_canopy	Logical. Extract standalone canopy height raster (CHM) and include canopy in fused DSM. The CHM is written as a separate file so OpenFOAM porous-zone definitions can reference it directly.
canopy_source	Character or NULL. "metachm", "ethCHM", or NULL.

`min_tree_height`
: Numeric. Minimum tree canopy height in metres.

`include_morphology`
: Logical. Add morphology metrics.

`include_neighbors`
: Logical. Add neighbour metrics.

`include_greenspace`
: Logical. Produce two greenspace outputs: (1) a ground roughness raster (z0, metres) from land-cover data for nutURoughWallFunction, and (2) distance-to-greenspace as a building-level attribute for post-processing / context.

`mask_tree_cover`
: Logical. When building the roughness raster, set tree-cover cells to NA so porous-zone drag is not counted twice. Default TRUE.

`landcover_source`
: Character. Land-cover dataset for the ground roughness raster. `"esa"` (default) uses ESA WorldCover (years 2020-2021); `"esri"` uses the Sentinel-2 10 m ESRI LULC Time Series (years 2017-2025), which provides more recent and historically consistent annual maps.

`landcover_year` Integer. Year of the land-cover product. For `"esa"`: 2020 or 2021. For `"esri"`: 2017-2025. Default 2021.

`include_bgvi` Logical. Add building green visibility index.

`include_shadow` Logical. Add shadow outputs.

`include_radiation`
: Logical. Add radiation outputs.

`opentopo_key` Character. OpenTopography API key for get_fused_dsm().

`target_crs` Optional projected target CRS. Usually leave as `NULL`: when the building data are in longitude/latitude, the local UTM zone is derived automatically from the data centroid, whichever of `bbox`, `place`, or `buildings_list` was supplied. Set this only to force a specific projection (e.g. a national grid). OpenFOAM domains must be metric, so a geographic CRS is rejected.

`height_col` Optional height column name. If NULL, guessed automatically.

`default_height` Numeric. Default building height if height column is missing (metres).

`min_height` Numeric. Minimum building height (metres).

`domain_buffer` Numeric. Buffer in metres around data extent.

`zmax_buffer` Numeric. Buffer in metres above maximum surface height.

`stl_name` Character. STL file name.

`overwrite` Logical. Whether to overwrite prepared files.

`quiet` Logical. Suppress messages.

Value

A list with:

case_dir Absolute path to the OpenFOAM case directory.

data_dir Absolute path to the gloBFPr data sub-directory.

files Named list of output file paths (NULL when not produced).

data Named list of in-memory spatial objects.

domain Named list with xmin/xmax/ymin/ymax/zmin/zmax for blockMeshDict.

origin Named numeric vector (x, y, z) of the local coordinate system origin in the input CRS.

crs CRS of the building data.

height_col Name of the height column used.

n_buildings Number of building polygons.

max_building_height Maximum building height in metres.

max_surface_height Maximum surface height (buildings + DSM).

include_tree_canopy Logical flag as supplied.

canopy_source Canopy data source as supplied.

min_tree_height Minimum tree height as supplied.

landcover_source Land-cover dataset used (`"esa"` or `"esri"`).

landcover_year Land-cover year used.

`read_foam_pedestrian_slice` *Read the pedestrian-level slice and compute wind maps*

Description

Reads the pedestrian-level surface sample written by the `pedestrianSlice` function object and returns a multi-layer `SpatRaster`. The current OpenFOAM workflow is wind-only, so the velocity layers are the primary result; temperature-derived layers should be approximately zero for neutral wind cases:

T_air Air temperature (K)

U_mag Wind speed (m/s)

T_cool Cooling relative to ambient, $T_{ref} - T_{air}$ (K)

T_cool_flux $\max(T_{cool}, 0) \times |U|$ - cool-air transport

Ux, Uy Horizontal velocity components (m/s)

The case is transient, so `time_step` selects a physical time in seconds since sunset; `"latest"` gives the end of the run.

Usage

```
read_foam_pedestrian_slice(
  case_dir,
  T_ref = NULL,
  time_step = "latest",
  resolution = 2,
  base_cell_size = 10,
  agl_tol = 5,
  trim = "auto",
  canopy = NULL,
  min_canopy_height = 2,
  crs = NA,
  buildings = NULL,
  quiet = FALSE
)
```

Arguments

case_dir	Character. OpenFOAM case directory.
T_ref	Numeric. Reference temperature (K) for T_cool, which is T_ref - T. It has to match the temperature the case was initialised at, or every cooling number is offset by the difference and T_cool can come out negative everywhere. The default, NULL, reads TRef from the case's constant/transportProperties and only falls back to 295 K when that file cannot be read.
time_step	"latest" (default) or a number of seconds.
resolution	Numeric. Output cell size (m). Default 2.
base_cell_size	Numeric. Background mesh cell size used when the case was generated (foam$params$base_cell_size) sets the gap-fill window. Default 10.
agl_tol	Numeric. Samples more than this many metres above the local ground surface are discarded. Guards against the stray near-vertical sheet a distanceSurface can generate around a closed terrain STL, which otherwise folds upper-level wind into the pedestrian map. Default 5.
trim	Crop the flow-adjustment zone off the returned raster. "auto" (default) removes, per side, the building-free apron plus an adjustment fetch; "buildings" removes only the apron; a number removes that many metres from every side; 0 returns the full domain. Trimming is on by default because the outer band is not a result. The domain is larger than the built area, and even where buildings reach the boundary the ABL inlet delivers an undisturbed profile that only slows as it works into the roughness. Measured on a real case, the outer 100 m averaged 1.86 m/s against 0.74 m/s in the interior - a bright rim that is an artifact of the domain, not a feature of the city. Excluding it is standard practice in urban CFD. Each side is trimmed by max(apron, fetch) - not their sum. The fetch is measured from the domain boundary and the apron is its first part, so they overlap rather than stack.

The fetch is ~5x the median building height (100 m when the layer has no height column), clamped to 50-150 m and to 10% of the shorter domain span. That is deliberately less than the ~15x an internal boundary layer needs to fully equilibrate: on a measured case the edge excess was concentrated in the outer 100 m, and beyond that the band-to-band variation was genuine urban structure rather than an edge effect. The amount removed on each side is reported.

`canopy` Canopy height raster (path or SpatRaster) used to build a canopy overlay for `plot_foam_map`. Auto-detected from `constant/gloBFPr/rasters/canopy_height.tif` when `NULL`; pass `FALSE` to skip it. The result is attached as the `"canopy"` attribute, mirroring `"buildings"`.

`min_canopy_height` Numeric. Canopy cells below this height are ignored. Default 2 m, matching `min_tree_height` in `prepare_openfoam_inputs`.

`crs` CRS to assign (e.g. `32617`). Default `NA`.

`buildings` Optional `sf` footprints in local coordinates; auto-detected from the case directory when `NULL`.

`quiet` Logical. Default `FALSE`.

Value

A `terra::SpatRaster` with six layers.

See Also

`prepare_foam_case`, `run_openfoam_docker`

`run_openfoam_docker` *Run an OpenFOAM case via Docker*

Description

Mounts the case directory into an OpenFOAM Docker container and executes the `Allrun` script produced by `prepare_foam_case()`. The generated wind case chains `blockMesh` -> `snappyHexMesh` -> `buoyantBoussinesqPimpleFoam` and writes log files in `case_dir`.

Docker Desktop must be running and the `case_dir` path must be under a directory shared with Docker (Docker Desktop -> Settings -> Resources -> File Sharing).

Usage

```
run_openfoam_docker(
  case_dir,
  image = "opencfd/openfoam-run:2506",
  ncpus = foam_default_ncpus(),
  wait = TRUE,
  quiet = FALSE
)
```

Arguments

case_dir	Character. Absolute path to the OpenFOAM case directory. Must contain an Allrun script (written by prepare_foam_case()).
image	Character. Docker image tag. Default "opencfd/openfoam-run:2506". Use the tag you pulled, e.g. "opencfd/openfoam-run:2406" for an older version.
ncpus	Integer. Number of CPU cores to use. Defaults to all physical cores detected on the host via parallel::detectCores(logical = FALSE). When ncpus > 1 the solver runs in MPI parallel: decomposePar splits the mesh, mpirun -np ncpus solver -parallel runs it, and reconstructPar reassembles. Typical speed-up is 3-6x on a 4-core machine. Pass ncpus = 1 to force single-core.
wait	Logical. If TRUE (default), R blocks until the simulation finishes. If FALSE, the container is launched in the background and the function returns immediately.
quiet	Logical. Suppress messages. Default FALSE.

Value

Invisibly returns a list with case_dir, image, and the exit status (0 = success; only meaningful when wait = TRUE).

See Also

prepare_foam_case

---

sample_foam_slice *Sample OpenFOAM results at pedestrian level and return a raster*

Description

Uses OpenFOAM's postProcess -func surfaces utility to cut a horizontal plane at z = 1.5 m (or any height) through the latest time-step, then reads the resulting .raw files into R and returns a multi-layer SpatRaster.

For the current wind workflow, prepare_foam_case() already writes a pedestrian slice during the solver run via the pedestrianSlice function object; use read_foam_pedestrian_slice for that default 1.5 m output. Use this helper to sample other heights or fields after a run.

Usage

```
sample_foam_slice(
  case_dir,
  fields = c("U", "p"),
  z = 1.5,
  image = "opencfd/openfoam-run:2506",
  resolution = 5,
  time_step = "latestTime",
  interpolate = c("smooth", "focal", "none"),
```

```
  idw_power = 2,
  idw_maxdist = NULL,
  buildings = NULL,
  quiet = FALSE
)
```

Arguments

case_dir	Character. OpenFOAM case directory.
fields	Character vector of field names to sample. Default c("U", "p").
z	Numeric. Height above ground in metres. Default 1.5.
image	Character. Docker image tag. Default "opencfd/openfoam-run:2506".
resolution	Numeric. Output raster cell size in metres. Default 5.
time_step	Character or numeric. "latestTime" (default) or a specific time-step number.
interpolate	Character. Post-rasterization smoothing method. "smooth" (default) fills NA gaps then applies a Gaussian focal filter to all cells, producing continuous gradients similar to ParaView; "focal" fills NA gaps only (faster, but blocky in open areas); "none" returns the raw rasterization without any smoothing.
idw_power	Numeric. Controls the Gaussian sigma when interpolate = "smooth": sigma = resolution * idw_power metres. Default 2 (10 m sigma at 5 m resolution). Increase for a wider blur; decrease for sharper transitions near buildings.
idw_maxdist	Ignored (reserved for future use).
buildings	Optional sf polygon layer of building footprints in the domain's local coordinate system. When NULL (default) the footprints saved by prepare_openfoam_inputs are loaded automatically from <case_dir>/constant/gloBFPr/metadata/buildings_openfoam.rds. Building interiors are masked to NA, and the layer is attached to the returned raster so plot_foam_map overlays it without any extra argument. Pass buildings = sf::st_sf(...) to override, or note that masking is skipped when no footprints can be found.
quiet	Logical. Default FALSE.

Value

A terra::SpatRaster with layers named after the sampled fields. Velocity U produces three extra layers: U_mag (speed), Ux, Uy. Building footprints, when available, are attached as attr(x, "buildings").

See Also

plot_foam_map, read_foam_pedestrian_slice

search_3dglobdf *search_3dglobdf*

Description

Search and retrieve 3D building footprint data from 3D-GloBFP or GlobalBuildingAtlas that intersect a given bounding box or area of interest (a city), with options to return vector or raster outputs including building polygons, binary presence rasters, and height-coded rasters.

Usage

```
search_3dglobdf(
  bbox = NULL,
  place = NULL,
  crop = FALSE,
  data_source = "GBF",
  keep_source_id = FALSE,
  out_type = "poly",
  mask = FALSE,
  cell_size = 1,
  quiet = TRUE
)
```

Arguments

bbox	sf, sfc, or a numeric vector (xmin, ymin, xmax, ymax) defining the area of interest. This can be ignored if place is specified.
place	vector (optional). A single line address, e.g. ("1600 Pennsylvania Ave NW, Washington") or a vector of addresses (c("Madrid", "Barcelona")).
crop	logical. If TRUE, the resulting building footprint geometries will be cropped to the input bbox. Default is FALSE.
data_source	character. Building data source to query. Use "GBF" for 3D-GloBFP (default) or "GBA"/"gba" for GlobalBuildingAtlas.
keep_source_id	logical. If TRUE, keep the original source feature identifier as source_id when it is available. Default is FALSE.
out_type	character. Default is 'poly'. Output type(s) to return. Options include: • "poly": building footprints as an sf polygon object. • "binary_rast": binary terra raster where buildings = 1. • "graduated_rast": terra raster encoding building height values. • "rast": a named list with both binary and graduated rasters. • "all": a named list including the polygon layer and both raster layers.
mask	logical (optional). Default is FALSE. If TRUE, masks the graduated raster using the building footprint layer. Only used when out_type is "graduated_rast", "rast", or "all".

cell_size	numeric (optional). Default is 1. Only used when out_type is "graduated_rast", "rast", or "all".
quiet	logical. If TRUE, suppress cli messages and progress output. Default is TRUE.

Value

Varies based on out_type:

- If "poly": an sf object of building footprints. MULTIPOLYGON geometries are converted to POLYGON geometries while preserving one row per source feature. Polygons that touch or intersect share a group_id, which can be used to treat fragmented rows as one building group.
- If "binary_rast": a binary SpatRaster (terra) indicating building presence.
- If "graduated_rast": a quantitative SpatRaster of building heights.
- If "rast": a named list with two SpatRaster objects: binary and graduated.
- If "all": a named list with poly (sf), binary, and graduated rasters.

Note

The downloading process may take some time, depending on the number and size of building footprint tiles.

This implementation for gloBFP-3D relies on the current structure of the dataset as hosted on Figshare. It may break if the dataset owner changes the file organization or metadata format.

The server of GlobalBuildingAtlas may have issues sometimes, so users may need to switch over to gloBFP-3D, which means using data_source="GBF"

When using data_source = "GBA", the GlobalBuildingAtlas dataset does not provide unique identifiers for individual building parcels. As a result, the group_id assigned to overlapping or fragmented polygons may not accurately reflect true building boundaries in all cases. This limitation may affect morphological analyses at the individual building level (e.g., footprint area, perimeter, building-level height statistics). However, the data remains suitable for environmental simulation purposes such as noise mapping, solar radiation analysis, and wind flow modelling, where parcel-level identity is less critical.

References

Che Yangzi, Li Xuecao, Liu Xiaoping, Wang Yuhao, Liao Weilin, Zheng Xianwei, Zhang Xucai, Xu Xiaocong, Shi Qian, Zhu Jiajun, Zhang Honghui, Yuan Hua, & Dai Yongjiu (2024). 3D-GloBFP: the first global three-dimensional building footprint dataset. Earth Syst. Sci. Data, 16, 5357-5374

Zhu X. X., Chen S., Zhang F., Shi Y., & Wang Y. (2025). GlobalBuildingAtlas: an open global and complete dataset of building polygons, heights and LoD1 3D models. Earth Syst. Sci. Data, 17, 6647-6668.

Examples

```
## Not run:
buildings <- gloBFPr::search_3dglobdf(bbox=c(-84.485519,45.636118,-84.462774,45.650639))

## End(Not run)
```

Index

∗ **datasets**
    globfp_example, 31
    globfp_example_canopy_height, 31
    globfp_example_dem, 32

add_flow_vectors, 3
aggregate_block, 4

generate_block, 5
generate_block(), *4*, *8*
get_3d_world, 6
get_bgvi *(*get_metrics*)*, 16
get_bgvi(), *35, 36*
get_dng *(*get_metrics*)*, 16
get_era5_met, 12
get_fused_dsm, 14
get_fused_dsm(), *16*, *18*
get_metadata, 15
get_metrics, 16
get_morphology *(*get_metrics*)*, 16
get_neighbors *(*get_metrics*)*, 16
get_noise_map, 22
get_noise_map(), *34*
get_radiation *(*get_shadows*)*, 26
get_shadow_footprint *(*get_shadows*)*, 26
get_shadow_height *(*get_shadows*)*, 26
get_shadows, 26
globfp_example, 31
globfp_example_canopy_height, 31
globfp_example_dem, 32

infer_osm_traffic, 33
install_noisemodelling, 34

osm_noise_traffic_defaults, 34

plot_bgvi_viewshed, 35
plot_foam_map, *3*, 36, *50*, *52*
plot_noise_map, 38
prepare_foam_case, 39, *42, 43*, *50, 51*
prepare_foam_case(), *12, 13*
prepare_foam_geometry, *40, 41*, 42
prepare_noisemodelling_inputs, 43
prepare_openfoam_inputs, *41, 42*, 45, *50*, *52*

read_foam_pedestrian_slice, *36, 37*, *41*, 48, *51, 52*
run_openfoam_docker, 50, *50*

sample_foam_slice, *36*, 51
search_3dglobdf, 53
search_3dglobdf(), *5*, *7, 8*, *10*, *14*, *17*, *28*, *35*
svf *(*get_shadows*)*, 26