Package 'aboveR'

March 19, 2026

Title 'LiDAR' Terrain Analysis and Change Detection from Above

Version 0.1.0

Description Terrain change detection, cut and fill volume estimation, terrain profiling, reclamation monitoring, erosion analysis, and flood risk assessment from 'LiDAR' (Light Detection and Ranging) point clouds and digital elevation models ('DEMs'). Applications include surface mine reclamation monitoring, sediment pond capacity tracking, highwall safety classification, and erosion channel detection. Built on 'lidR' for point cloud I/O and 'terra' for raster operations. Includes access utilities for 'KyFromAbove' cloud-native elevation data on Amazon Web Services ('AWS') <`https://kyfromabove.ky.gov/`>. Methods for terrain change detection and volume estimation follow Li and others (2005) <doi:10.1016/j.geomorph.2004.10.007>.

License MIT + file LICENSE

URL `https://github.com/chrislyonsKY/aboveR`

BugReports `https://github.com/chrislyonsKY/aboveR/issues`

Depends R (>= 4.1.0)

Imports lidR, terra, sf

Suggests rstac, httr2, cli, rgl, mapview, ggplot2, whitebox, testthat (>= 3.0.0), knitr, rmarkdown

Encoding UTF-8

RoxygenNote 7.3.3

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Chris Lyons [aut, cre]

Maintainer Chris Lyons <`chris.lyons@ky.gov`>

Repository CRAN

Date/Publication 2026-03-19 13:10:10 UTC

Contents

`boundary_terrain_profile`

Extract Terrain Profile Along a Polygon Boundary

Description

Converts a polygon boundary to a closed linestring and samples elevation values around the perimeter. Useful for inspecting highwall edges, dam crests, or property boundaries.

Usage

```
boundary_terrain_profile(dem, boundary, spacing = NULL)
```

Arguments

`dem`	A terra::SpatRaster representing the terrain surface.
`boundary`	An sf::sf polygon whose boundary (exterior ring) will be profiled.
`spacing`	Numeric. Distance between sample points, in CRS units of `dem`. Default `NULL` uses 1 cell width.

Value

A data frame with columns:

- `distance`: distance along the boundary perimeter
- `elevation`: sampled elevation value
- `x`, `y`: coordinates of each sample point

Examples

```
dem <- terra::rast(system.file("extdata/dem_before.tif", package = "aboveR"))
boundary <- sf::st_read(
  system.file("extdata/boundary.gpkg", package = "aboveR"),
  quiet = TRUE
)
bprof <- boundary_terrain_profile(dem, boundary)
plot(bprof$distance, bprof$elevation, type = "l",
     xlab = "Perimeter Distance", ylab = "Elevation")
```

`change_by_zone` *Summarise Terrain Change by Zone*

Description

Extracts change statistics for each polygon in a zone layer, computing cut volume, fill volume, net change, and descriptive statistics per zone.

Usage

```
change_by_zone(change_raster, zones, id_field)
```

Arguments

`change_raster`	A terra::SpatRaster as returned by `terrain_change()` (uses the `"change"` layer).
`zones`	An sf::sf data frame of polygons defining analysis zones.
`id_field`	Character. Column name in `zones` to use as zone identifier.

Value

An sf::sf data frame with columns:

- zone identifier
- `cut_volume`: total volume of material removed (m^3, positive)
- `fill_volume`: total volume of material added (m^3, positive)
- `net_volume`: fill - cut (m^3)
- `area_m2`: zone area
- `mean_change`: mean elevation change
- `max_cut`: deepest cut (most negative value)
- `max_fill`: highest fill (most positive value)

Examples

```
before <- terra::rast(system.file("extdata/dem_before.tif", package = "aboveR"))
after  <- terra::rast(system.file("extdata/dem_after.tif", package = "aboveR"))
chg <- terrain_change(before, after)
zones <- sf::st_read(
  system.file("extdata/zones.gpkg", package = "aboveR"),
  quiet = TRUE
)
change_by_zone(chg, zones, id_field = "zone_id")
```

`classify_highwall` *Classify Highwall Areas from a DEM*

Description

Identifies steep terrain faces (highwalls) typical of surface mining operations by computing slope from a DEM and classifying cells that exceed a slope threshold. Returns a binary raster and optionally vectorised polygons of highwall zones.

Usage

```
classify_highwall(dem, slope_threshold = 60, min_area = 0, as_polygons = FALSE)
```

Arguments

`dem`	A terra::SpatRaster representing the terrain surface.
`slope_threshold`	Numeric. Minimum slope in degrees to classify as highwall. Default `60`.
`min_area`	Numeric. Minimum contiguous area (in map units squared) for a highwall zone. Smaller patches are removed. Default `0` (keep all).
`as_polygons`	Logical. Return vectorised polygons instead of a raster? Default `FALSE`.

Value

If `as_polygons = FALSE`, a terra::SpatRaster with values 1 (highwall) and `NA` (non-highwall). If `as_polygons = TRUE`, an sf::sf data frame of highwall polygons with an `area_m2` column.

Examples

```
dem <- terra::rast(system.file("extdata/dem_before.tif", package = "aboveR"))
hw <- classify_highwall(dem, slope_threshold = 5)
terra::plot(hw)
```

`detect_channels` *Detect Erosion Channels from a DEM*

Description

Identifies potential erosion channels (rills, gullies) by computing flow accumulation and filtering cells where accumulated flow exceeds a threshold. Optionally returns vectorised channel lines.

Usage

```
detect_channels(dem, threshold = 100, as_lines = FALSE)
```

Arguments

`dem`	A terra::SpatRaster.
`threshold`	Numeric. Minimum flow accumulation value to classify as a channel. Default `100`.
`as_lines`	Logical. Return channel centrelines as sf::sf lines? Default `FALSE` (returns raster).

Value

If `as_lines = FALSE`, a terra::SpatRaster with channel cells = 1, other = `NA`. If `as_lines = TRUE`, an sf::sf LINESTRING object of detected channels.

Examples

```
dem <- terra::rast(system.file("extdata/dem_before.tif", package = "aboveR"))
channels <- detect_channels(dem, threshold = 50)
terra::plot(channels)
```

`estimate_volume` *Estimate Cut or Fill Volume Between Two Surfaces*

Description

Computes the volume of material between a `surface` DEM and a `reference` surface (e.g., a design grade or pre-mining DEM), optionally clipped to a `boundary` polygon. Reports cut and fill volumes separately and documents the integration method used.

Usage

```
estimate_volume(
  surface,
  reference,
  boundary = NULL,
  method = c("trapezoidal", "simpson")
)
```

Arguments

`surface`	A terra::SpatRaster representing the actual surface.
`reference`	A terra::SpatRaster or single numeric value representing the reference elevation. If numeric, a constant reference plane at that elevation is used.
`boundary`	An optional sf::sf polygon to clip both surfaces to before computation.
`method`	Character. Volume integration method: `"trapezoidal"` (default) or `"simpson"`. Trapezoidal sums `abs(diff) * cell_area`. Simpson uses Simpson's 1/3 rule for higher accuracy.

Value

A list with components:

- `cut_volume_m3`: volume of material removed (positive)
- `fill_volume_m3`: volume of material added (positive)
- `net_volume_m3`: fill minus cut
- `area_m2`: total analysed area
- `mean_depth_m`: mean absolute difference
- `max_cut_m`: deepest cut
- `max_fill_m`: highest fill
- `method`: integration method used

Examples

```
surface <- terra::rast(system.file("extdata/dem_after.tif", package = "aboveR"))
reference <- terra::rast(system.file("extdata/dem_before.tif", package = "aboveR"))
boundary <- sf::st_read(
  system.file("extdata/boundary.gpkg", package = "aboveR"),
  quiet = TRUE
)
vol <- estimate_volume(surface, reference, boundary)
cat("Net volume:", vol$net_volume_m3, "m3\n")
```

`has_s3_access` *Check S3 Access to KyFromAbove Bucket*

Description

Returns `TRUE` only when the `ABOVER_KFA_TEST` environment variable is set, ensuring KyFromAbove examples never run on CRAN or in environments without verified S3 access.

Usage

```
has_s3_access()
```

Value

Logical scalar indicating whether the KyFromAbove test environment variable is set.

Examples

```
has_s3_access()
```

`impoundment_curve` *Generate an Impoundment Capacity Curve*

Description

Calculates storage volume at a series of water surface elevations for a terrain depression (e.g., a pond, reservoir, or sediment basin). The result is an elevation-area-volume curve.

Usage

```
impoundment_curve(dem, boundary = NULL, elevations = NULL, n_steps = 20L)
```

Arguments

`dem`	A terra::SpatRaster representing the terrain surface.
`boundary`	An sf::sf polygon defining the impoundment boundary (e.g., dam crest outline). If `NULL`, the full DEM extent is used.
`elevations`	Numeric vector of water surface elevations to evaluate. If `NULL`, a sequence from the minimum to maximum DEM value within the boundary is generated with `n_steps` increments.
`n_steps`	Integer. Number of elevation increments when `elevations` is `NULL`. Default 20.

Value

A data frame with columns:

- `elevation`: water surface elevation
- `area_m2`: inundated area at this elevation
- `volume_m3`: cumulative storage volume below this elevation

Examples

```
dem <- terra::rast(system.file("extdata/dem_before.tif", package = "aboveR"))
boundary <- sf::st_read(
  system.file("extdata/boundary.gpkg", package = "aboveR"),
  quiet = TRUE
)
curve <- impoundment_curve(dem, boundary, n_steps = 10)
plot(curve$elevation, curve$volume_m3, type = "l",
     xlab = "Elevation", ylab = "Volume (m3)")
```

kfa_find_tiles *Find KyFromAbove Tiles Covering an Area of Interest*

Description

Queries KyFromAbove tile indexes to find elevation, point cloud, or imagery tiles that intersect a given area of interest. Tries the STAC catalog first (if available), then falls back to the tile index GeoPackage on S3.

Usage

```
kfa_find_tiles(
  aoi,
  product = "dem",
  phase = 2L,
  method = c("auto", "stac", "index")
)
```

Arguments

`aoi`	An sf::sf object or numeric bbox (`c(xmin, ymin, xmax, ymax)`) defining the area of interest. Any CRS is accepted; reprojection to EPSG:3089 (Kentucky Single Zone) is handled internally.
`product`	Character. One of `"dem"`, `"pointcloud"`, `"ortho"`, `"contours"`, `"oblique"`.
`phase`	Integer. KyFromAbove acquisition phase: 1, 2, or 3. Phase 1 DEMs are 5ft resolution; Phase 2/3 are 2ft.
`method`	Character. `"auto"` (default) tries STAC first, then tile index. `"stac"` uses STAC only. `"index"` uses tile index only.

Value

An sf::sf data frame with columns: `tilename`, `s3_url`, `phase`, `product`, and `geometry`.

Examples

```
# Find Phase 2 DEM tiles for a bounding box in Fayette County
tiles <- kfa_find_tiles(
  aoi = c(-84.55, 37.95, -84.45, 38.05),
  product = "dem",
  phase = 2
)
```

`kfa_read_dem` *Read KyFromAbove DEMs for an Area of Interest*

Description

Finds DEM tiles covering the AOI, reads them as Cloud-Optimized GeoTIFFs via `/vsicurl/`, optionally merges into a single raster, and crops to the AOI extent.

Usage

```
kfa_read_dem(aoi, phase = 2L, merge = TRUE, crop = TRUE, cache = FALSE)
```

Arguments

`aoi`	An sf::sf object or numeric bbox.
`phase`	Integer. 1 (5ft), 2 (2ft), or 3 (2ft).
`merge`	Logical. Mosaic multiple tiles into one SpatRaster? Default `TRUE`.
`crop`	Logical. Crop result to AOI extent? Default `TRUE`.
`cache`	Logical. Cache downloaded tiles locally? Default `FALSE`.

Value

A terra::SpatRaster object.

Examples

```
dem <- kfa_read_dem(
  aoi = c(-84.55, 37.95, -84.45, 38.05),
  phase = 2
)
```

`kfa_read_ortho` *Read KyFromAbove Orthoimagery for an Area of Interest*

Description

Finds ortho (nadir) or oblique imagery tiles covering the AOI and reads them as RGB SpatRaster.

Usage

```
kfa_read_ortho(aoi, phase = 3L, type = c("nadir", "oblique"))
```

Arguments

`aoi`	An sf::sf object or numeric bbox.
`phase`	Integer. 1, 2, or 3.
`type`	Character. `"nadir"` (default) or `"oblique"` (Phase 3 only).

Value

A terra::SpatRaster object with 3 bands (RGB).

Examples

```
ortho <- kfa_read_ortho(
  aoi = c(-84.55, 37.95, -84.54, 37.96),
  phase = 3
)
```

`kfa_read_pointcloud` *Read KyFromAbove Point Cloud for an Area of Interest*

Description

Finds point cloud tiles (LAZ for Phase 1, COPC for Phase 2/3) covering the AOI and reads them via `lidR::readLAS()`.

Usage

```
kfa_read_pointcloud(aoi, phase = 2L)
```

Arguments

`aoi`	An sf::sf object or numeric bbox.
`phase`	Integer. 1, 2, or 3.

Value

A lidR::LAS object.

Examples

```
las <- kfa_read_pointcloud(
  aoi = c(-84.55, 37.95, -84.54, 37.96),
  phase = 2
)
```

`kfa_stac_search` *Search KyFromAbove STAC Catalog*

Description

Queries the KyFromAbove STAC catalog for items matching an area of interest and product type. Requires the 'rstac' package.

Usage

```
kfa_stac_search(aoi, collection = NULL, datetime = NULL)
```

Arguments

`aoi`	An sf::sf object or numeric bbox.
`collection`	Character. STAC collection ID.
`datetime`	Character. ISO 8601 datetime or range.

Value

A tibble of STAC items with asset URLs.

`kfa_tile_index` *Load and Cache a KyFromAbove Tile Index*

Description

Downloads a tile index GeoPackage from the KyFromAbove S3 bucket and caches it locally. Subsequent calls use the cached copy unless it is older than `max_age_days`.

Usage

```
kfa_tile_index(product = "dem", phase = 2L, max_age_days = 30L)
```

Arguments

`product`	Character. One of `"dem"`, `"pointcloud"`, `"ortho"`.
`phase`	Integer. KyFromAbove acquisition phase: 1, 2, or 3.
`max_age_days`	Integer. Re-download if cache is older than this. Default 30.

Value

An sf::sf data frame representing the tile index grid.

Examples

```
idx <- kfa_tile_index(product = "dem", phase = 2)
head(idx)
```

`pond_sedimentation` *Estimate Pond Sedimentation from Multi-Temporal DEMs*

Description

Compares two DEMs of a pond or sediment basin to estimate the volume of accumulated sediment. The pond area is defined by a boundary polygon, and sedimentation is computed as fill (positive change) within that area.

Usage

```
pond_sedimentation(before, after, pond_boundary)
```

Arguments

`before`	A terra::SpatRaster of the pond before sedimentation.
`after`	A terra::SpatRaster of the pond after sedimentation.
`pond_boundary`	An sf::sf polygon defining the pond extent.

Value

A list with:

- `sediment_volume_m3`: estimated sediment volume (positive fill)
- `mean_depth_change_m`: mean elevation change within the pond
- `max_accumulation_m`: maximum sedimentation depth
- `pond_area_m2`: area of the pond boundary
- `change_raster`: terra::SpatRaster of elevation change within the pond

Examples

```
before <- terra::rast(system.file("extdata/dem_before.tif", package = "aboveR"))
after  <- terra::rast(system.file("extdata/dem_after.tif", package = "aboveR"))
boundary <- sf::st_read(
  system.file("extdata/boundary.gpkg", package = "aboveR"),
  quiet = TRUE
)
sed <- pond_sedimentation(before, after, boundary)
cat("Sediment volume:", sed$sediment_volume_m3, "m3\n")
```

`reclamation_progress` *Assess Reclamation Progress Between Time Steps*

Description

Compares a current DEM against a target (design grade) surface to quantify how much of a reclamation area has been restored to the desired elevation. Returns per-cell deviations and summary statistics.

Usage

```
reclamation_progress(current, target, boundary = NULL, tolerance = 0.3)
```

Arguments

`current`	A terra::SpatRaster of the current terrain surface.
`target`	A terra::SpatRaster of the target / design grade surface.
`boundary`	An optional sf::sf polygon to restrict analysis to.
`tolerance`	Numeric. Cells within +/- `tolerance` of the target are considered "on grade". Default `0.3` (metres).

Value

A list with:

- `deviation`: terra::SpatRaster of current - target values
- `on_grade_pct`: percentage of cells within tolerance
- `above_grade_pct`: percentage of cells above tolerance
- `below_grade_pct`: percentage of cells below tolerance
- `mean_deviation`: mean signed deviation
- `rmse`: root mean square error

Examples

```
current <- terra::rast(system.file("extdata/dem_after.tif", package = "aboveR"))
target  <- terra::rast(system.file("extdata/dem_before.tif", package = "aboveR"))
prog <- reclamation_progress(current, target, tolerance = 1)
cat("On grade:", prog$on_grade_pct, "%\n")
```

`surface_roughness` *Compute Surface Roughness of a DEM*

Description

Calculates surface roughness as the standard deviation of elevation within a moving window. Rougher surfaces indicate unreclaimed terrain, active construction, or natural heterogeneity.

Usage

```
surface_roughness(dem, window = 5L)
```

Arguments

`dem`	A terra::SpatRaster.
`window`	Integer. Size of the moving window (number of cells). Must be odd. Default `5`.

Value

A terra::SpatRaster of local roughness values (standard deviation of elevation).

Examples

```
dem <- terra::rast(system.file("extdata/dem_before.tif", package = "aboveR"))
rough <- surface_roughness(dem, window = 5)
terra::plot(rough)
```

`terrain_change` *Compute Terrain Change Between Two DEMs*

Description

Calculates the elevation difference between a *before* and *after* DEM, returning both the continuous change values and a classified layer (cut / stable / fill). Rasters are aligned automatically if their extents or resolutions differ (same CRS required).

Usage

```
terrain_change(before, after, tolerance = 0.1)
```

Arguments

`before`	A terra::SpatRaster representing the earlier DEM.
`after`	A terra::SpatRaster representing the later DEM.
`tolerance`	Numeric. Changes within +/- `tolerance` are classified as stable. Default `0.1` (metres or native units).

Value

A two-layer terra::SpatRaster:

- `change`: continuous elevation difference (after - before)
- `class`: integer classification (1 = cut, 2 = stable, 3 = fill)

Examples

```
before <- terra::rast(system.file("extdata/dem_before.tif", package = "aboveR"))
after  <- terra::rast(system.file("extdata/dem_after.tif", package = "aboveR"))
result <- terrain_change(before, after)
terra::plot(result[["change"]])
```

`terrain_profile` *Extract a Terrain Profile Along a Line*

Description

Samples elevation values from a DEM at regular intervals along a transect line and returns a data frame of distance vs. elevation.

Usage

```
terrain_profile(dem, line, spacing = NULL)
```

Arguments

`dem`	A terra::SpatRaster representing the terrain surface.
`line`	An sf::sf object containing a single LINESTRING geometry defining the transect. Or a path to a GeoPackage/shapefile.
`spacing`	Numeric. Distance between sample points along the line, in the CRS units of `dem`. Default `NULL` uses 1 cell width.

Value

A data frame with columns:

- `distance`: distance along the profile from the start point
- `elevation`: sampled elevation value
- `x`, `y`: coordinates of each sample point

Examples

```
dem <- terra::rast(system.file("extdata/dem_before.tif", package = "aboveR"))
line <- sf::st_read(
  system.file("extdata/profile_line.gpkg", package = "aboveR"),
  quiet = TRUE
)
prof <- terrain_profile(dem, line)
plot(prof$distance, prof$elevation, type = "l",
     xlab = "Distance", ylab = "Elevation")
```

Index