

Package: aboveR (via r-universe)

June 3, 2026

Title 'LiDAR' Terrain Analysis and Change Detection from Above

Version 1.0.0

Description Terrain change detection, cut and fill volume estimation, terrain profiling, flood inundation analysis, slope and aspect computation, hillshade generation, contour extraction, reclamation monitoring, erosion analysis, and engineering export (LandXML, STL) 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](https://doi.org/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 covr, rstac, httr2, cli, rgl, mapview, ggplot2, whitebox, withr, testthat (>= 3.0.0), knitr, rmarkdown

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/testthat/edition 3

VignetteBuilder knitr

Config/pak/sysreqs libabsl-dev cmake libfreetype6-dev libgdal-dev gdal-bin libgeos-dev libglu1-mesa-dev make texlive libpng-dev libuv1-dev libglib2.0-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev zlib1g-dev

Repository <https://chrislyonsky.r-universe.dev>
Date/Publication 2026-04-02 11:32:04 UTC
RemoteUrl <https://github.com/chrislyonsky/abover>
RemoteRef HEAD
RemoteSha 1eae78ec36cb63a4800bb89621b29d010f59ff68

Contents

bench_detection	3
boundary_terrain_profile	4
change_by_zone	5
change_colors	6
classify_highwall	6
contour_lines	7
detect_channels	8
estimate_volume	8
export_landxml	9
export_stl	10
flood_colors	11
flood_depth	12
flood_inundation	12
has_s3_access	13
height_above_drainage	13
hillshade	14
impoundment_curve	15
kfa_county_bbox	16
kfa_find_tiles	16
kfa_list_counties	17
kfa_read_dem	18
kfa_read_ortho	18
kfa_read_pointcloud	19
kfa_stac_search	20
kfa_tile_index	21
pond_sedimentation	21
reclamation_progress	22
slope_aspect	23
surface_roughness	24
terrain_change	24
terrain_colors	25
terrain_profile	26
zonal_stats	27

Index

28

bench_detection	<i>Detect Mining Benches from a DEM</i>
-----------------	-----------------------------------------

Description

Identifies flat bench areas between highwall faces in surface mining terrain. Benches are detected as relatively flat areas (low slope) bounded by steep terrain (high slope), using slope segmentation and area filtering.

Usage

```
bench_detection(dem, max_slope = 10, min_area = 100, highwall_slope = 40)
```

Arguments

dem	A terra::SpatRaster representing the terrain surface.
max_slope	Numeric. Maximum slope in degrees for a cell to be considered part of a bench. Default 10.
min_area	Numeric. Minimum contiguous area (in map units squared) for a bench. Smaller patches are removed. Default 100.
highwall_slope	Numeric. Minimum slope for adjacent highwall terrain. Default 40.

Value

An [sf:sf](#) data frame of bench polygons with columns:

- bench_id: sequential bench identifier
- area_m2: bench area in square metres
- mean_elev: mean elevation of the bench surface
- mean_slope: mean slope within the bench

Examples

```
dem <- terra::rast(system.file("extdata/dem_before.tif", package = "aboveR"))
benches <- bench_detection(dem, max_slope = 15, min_area = 0)
if (nrow(benches) > 0) plot(sf::st_geometry(benches))
```

`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

<code>dem</code>	A <code>terra::SpatRaster</code> representing the terrain surface.
<code>boundary</code>	An <code>sf::sf</code> polygon whose boundary (exterior ring) will be profiled.
<code>spacing</code>	Numeric. Distance between sample points, in CRS units of <code>dem</code> . Default <code>NULL</code> 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	<i>Summarise Terrain Change by Zone</i>
----------------	-----------------------------------------

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 (m³, positive)
- fill_volume: total volume of material added (m³, positive)
- net_volume: fill - cut (m³)
- 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")
```

change_colors	<i>Diverging Color Ramp for Change Maps</i>
---------------	---------------------------------------------

Description

Returns a blue-white-red color ramp centered at zero, suitable for terrain change visualisation. Cut (negative) is blue, fill (positive) is red.

Usage

```
change_colors(n = 100L)
```

Arguments

n Integer. Number of colors. Default 100.

Value

Character vector of hex color values.

Examples

```
cols <- change_colors(10)
image(matrix(1:10), col = cols)
```

classify_highwall	<i>Classify Highwall Areas from a DEM</i>
-------------------	-------------------------------------------

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)
```

`contour_lines`*Generate Contour Lines from a DEM*

Description

Extracts contour lines at a specified interval and returns them as an [sf::sf](#) object with an elevation attribute.

Usage

```
contour_lines(dem, interval = 5, levels = NULL)
```

Arguments

<code>dem</code>	A terra::SpatRaster representing the terrain surface.
<code>interval</code>	Numeric. Elevation interval between contour lines. Default 5.
<code>levels</code>	Numeric vector. Specific elevations to contour. If provided, <code>interval</code> is ignored.

Value

An [sf::sf](#) data frame with LINESTRING geometry and an elevation column.

Examples

```
dem <- terra::rast(system.file("extdata/dem_before.tif", package = "aboveR"))
cl <- contour_lines(dem, interval = 2)
plot(sf::st_geometry(cl))
```

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 <code>terra::SpatRaster</code> representing the actual surface.
reference	A <code>terra::SpatRaster</code> or single numeric value representing the reference elevation. If numeric, a constant reference plane at that elevation is used.
boundary	An optional <code>sf::sf</code> polygon to clip both surfaces to before computation.
method	Character. Volume integration method: "trapezoidal" (default) or "simpson". Trapezoidal sums $\text{abs}(\text{diff}) * \text{cell_area}$. 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")
```

export_landxml

Export a DEM to LandXML TIN Surface

Description

Converts a DEM raster to a LandXML file containing a TIN (Triangulated Irregular Network) surface. LandXML is widely supported by civil engineering software including Autodesk Civil 3D, Bentley OpenRoads, and Trimble Business Center.

Usage

```
export_landxml(
  dem,
  output,
  surface_name = "Ground",
  decimate = 1L,
  boundary = NULL
)
```

Arguments

dem	A terra::SpatRaster representing the terrain surface.
output	Character. Output file path (should end in .xml).
surface_name	Character. Name for the TIN surface in the XML. Default "Ground".
decimate	Integer. Keep every Nth point to reduce file size. Default 1 (keep all points). Use 2 to halve the point count, 5 to keep every 5th point, etc.
boundary	An optional sf::sf polygon to clip the DEM before export.

Value

The output file path (invisibly).

Examples

```
dem <- terra::rast(system.file("extdata/dem_before.tif", package = "aboveR"))
out <- file.path(tempdir(), "surface.xml")
export_landxml(dem, out, surface_name = "Existing")
file.exists(out)
```

export_stl

Export a DEM to STL for 3D Printing

Description

Converts a DEM raster to an STL (stereolithography) file suitable for 3D printing. The model includes a flat base and vertical exaggeration control.

Usage

```
export_stl(dem, output, exaggeration = 1, base_height = 1, decimate = 1L)
```

Arguments

dem	A terra::SpatRaster representing the terrain surface.
output	Character. Output file path (should end in .stl).
exaggeration	Numeric. Vertical exaggeration factor. Default 1 (no exaggeration). Use 2 for 2x vertical stretch.
base_height	Numeric. Thickness of the flat base below the terrain minimum, in DEM units. Default 1.
decimate	Integer. Keep every Nth cell. Default 1.

Value

The output file path (invisibly).

Examples

```
dem <- terra::rast(system.file("extdata/dem_before.tif", package = "aboveR"))
out <- file.path(tempdir(), "terrain.stl")
export_stl(dem, out, exaggeration = 2)
file.exists(out)
```

flood_colors	<i>Flood Depth Color Ramp</i>
--------------	-------------------------------

Description

Light-to-dark blue color ramp for flood depth visualisation.

Usage

```
flood_colors(n = 100L)
```

Arguments

n	Integer. Number of colors. Default 100.
---	-----------------------------------------

Value

Character vector of hex color values.

Examples

```
cols <- flood_colors(10)
image(matrix(1:10), col = cols)
```

flood_depth	<i>Compute Flood Depth at a Given Water Level</i>
-------------	---------------------------------------------------

Description

Calculates the depth of water at each cell for a given water surface elevation. Cells above the water level receive NA.

Usage

```
flood_depth(dem, water_level, boundary = NULL)
```

Arguments

dem	A terra::SpatRaster representing the terrain surface.
water_level	Numeric. Water surface elevation.
boundary	An optional sf::sf polygon to restrict analysis to.

Value

A [terra::SpatRaster](#) of water depth values (positive, in DEM elevation units). Cells above the water level are NA.

Examples

```
dem <- terra::rast(system.file("extdata/dem_before.tif", package = "aboveR"))
depth <- flood_depth(dem, water_level = 305)
terra::plot(depth, main = "Flood Depth (m)")
```

flood_inundation	<i>Generate a Flood Inundation Mask</i>
------------------	-----------------------------------------

Description

Creates a binary raster showing which cells would be inundated at a given water surface elevation. Cells below the water level are marked as flooded (1), cells above are NA.

Usage

```
flood_inundation(dem, water_level, boundary = NULL)
```

Arguments

dem	A terra::SpatRaster representing the terrain surface.
water_level	Numeric. Water surface elevation in the same units as the DEM.
boundary	An optional sf::sf polygon to restrict analysis to (e.g., a floodplain boundary).

Value

A `terra::SpatRaster` with values 1 (inundated) and NA (dry).

Examples

```
dem <- terra::rast(system.file("extdata/dem_before.tif", package = "aboveR"))
inundated <- flood_inundation(dem, water_level = 305)
terra::plot(inundated, main = "Flood Inundation at 305m")
```

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()
```

height_above_drainage *Compute Height Above Nearest Drainage (HAND)*

Description

Calculates a relative elevation model representing each cell's height above the nearest drainage channel. This is a key input for flood susceptibility mapping. Channels are identified using Topographic Position Index (TPI).

Usage

```
height_above_drainage(dem, channel_threshold = 0.1, window = 15L)
```

Arguments

dem	A terra::SpatRaster representing the terrain surface.
channel_threshold	Numeric. TPI percentile threshold for channel identification. Lower values select deeper channels. Default 0.1 (10th percentile).
window	Integer. Focal window size for TPI computation. Must be odd. Default 15.

Value

A [terra::SpatRaster](#) of height-above-nearest-drainage values. Lower values indicate greater flood susceptibility.

Examples

```
dem <- terra::rast(system.file("extdata/dem_before.tif", package = "aboveR"))
hand <- height_above_drainage(dem, window = 7)
terra::plot(hand, main = "Height Above Nearest Drainage")
```

hillshade

Generate a Hillshade from a DEM

Description

Computes an analytical hillshade (shaded relief) from a DEM using specified sun azimuth and altitude angles. Useful for terrain visualisation and map backgrounds.

Usage

```
hillshade(dem, azimuth = 315, altitude = 45)
```

Arguments

dem	A terra::SpatRaster representing the terrain surface.
azimuth	Numeric. Sun azimuth in degrees (compass bearing). Default 315 (northwest).
altitude	Numeric. Sun altitude angle in degrees above the horizon. Default 45.

Value

A [terra::SpatRaster](#) with hillshade values (0–255).

Examples

```
dem <- terra::rast(system.file("extdata/dem_before.tif", package = "aboveR"))
hs <- hillshade(dem)
terra::plot(hs, col = grey.colors(256))
```

impoundment_curve	<i>Generate an Impoundment Capacity Curve</i>
-------------------	-----------------------------------------------

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 <code>terra::SpatRaster</code> representing the terrain surface.
boundary	An <code>sf::sf</code> 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 <code>n_steps</code> increments.
n_steps	Integer. Number of elevation increments when <code>elevations</code> 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_county_bbox	<i>Get Bounding Box for a Kentucky County</i>
-----------------	-----------------------------------------------

Description

Returns a bounding box (in EPSG:4326) for a Kentucky county by name. Useful for quickly querying KyFromAbove tiles without constructing an AOI manually.

Usage

```
kfa_county_bbox(county)
```

Arguments

county Character. County name (case-insensitive). Partial matching is supported.

Value

A numeric vector `c(xmin, ymin, xmax, ymax)` in EPSG:4326.

Examples

```
# Get bounding box for Fayette County
bbox <- kfa_county_bbox("Fayette")
print(bbox)

# Use directly with kfa_find_tiles
tiles <- kfa_find_tiles(kfa_county_bbox("Fayette"), product = "dem")
```

kfa_find_tiles	<i>Find KyFromAbove Tiles Covering an Area of Interest</i>
----------------	------------------------------------------------------------

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 <code>sf::sf</code> object or numeric bbox (<code>c(xmin, ymin, xmax, ymax)</code>) 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
)
```

<code>kfa_list_counties</code>	<i>List Available Kentucky Counties</i>
--------------------------------	-----------------------------------------

Description

Returns a character vector of all 120 Kentucky county names.

Usage

```
kfa_list_counties()
```

Value

Character vector of county names sorted alphabetically.

Examples

```
counties <- kfa_list_counties()
head(counties)
```

kfa_read_dem	<i>Read KyFromAbove DEMs for an Area of Interest</i>
--------------	------------------------------------------------------

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 <code>sf::sf</code> object or numeric bbox.
phase	Integer. 1 (5ft), 2 (2ft), or 3 (2ft).
merge	Logical. Mosaic multiple tiles into one <code>SpatRaster</code> ? 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	<i>Read KyFromAbove Orthoimagery for an Area of Interest</i>
----------------	--------------------------------------------------------------

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	<i>Search KyFromAbove STAC Catalog</i>
-----------------	----------------------------------------

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

<code>aoi</code>	An <code>sf::sf</code> object or numeric bbox.
<code>collection</code>	Character. STAC collection ID (e.g., "dem_phase2").
<code>datetime</code>	Character. ISO 8601 datetime or range (e.g., "2023-01-01/2023-12-31").

Details

The KyFromAbove STAC catalog is under active development by Ian Horn (github.com/ianhorn/kyfromabove-stac). This function will be activated when the catalog is live. In the meantime, use `kfa_find_tiles()` with `method = "index"`.

Value

An `sf::sf` data frame of STAC items with asset URLs, or an error if the STAC catalog is not yet available.

Examples

```
# STAC catalog not yet available - use kfa_find_tiles() instead
tiles <- kfa_find_tiles(
  aoi = c(-84.55, 37.95, -84.45, 38.05),
  product = "dem", phase = 2
)
```

kfa_tile_index	<i>Load and Cache a KyFromAbove Tile Index</i>
----------------	------------------------------------------------

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

<code>product</code>	Character. One of "dem", "pointcloud", "ortho".
<code>phase</code>	Integer. KyFromAbove acquisition phase: 1, 2, or 3.
<code>max_age_days</code>	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	<i>Estimate Pond Sedimentation from Multi-Temporal DEMs</i>
--------------------	-------------------------------------------------------------

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

<code>before</code>	A <code>terra::SpatRaster</code> of the pond before sedimentation.
<code>after</code>	A <code>terra::SpatRaster</code> of the pond after sedimentation.
<code>pond_boundary</code>	An <code>sf::sf</code> 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

<code>current</code>	A terra::SpatRaster of the current terrain surface.
<code>target</code>	A terra::SpatRaster of the target / design grade surface.
<code>boundary</code>	An optional sf::sf polygon to restrict analysis to.
<code>tolerance</code>	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")
```

slope_aspect	<i>Compute Slope and Aspect from a DEM</i>
--------------	--------------------------------------------

Description

Calculates slope (in degrees or percent) and aspect (compass bearing) from a DEM in a single call.

Usage

```
slope_aspect(dem, units = c("degrees", "percent"))
```

Arguments

dem A [terra::SpatRaster](#) representing the terrain surface.
units Character. Slope units: "degrees" (default) or "percent".

Value

A two-layer [terra::SpatRaster](#):

- slope: terrain slope in the requested units
- aspect: terrain aspect in degrees (0 = North, 90 = East, 180 = South, 270 = West; flat areas = -1)

Examples

```
dem <- terra::rast(system.file("extdata/dem_before.tif", package = "aboveR"))
sa <- slope_aspect(dem)
terra::plot(sa)
```

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_colors	<i>Terrain Color Ramp</i>
----------------	---------------------------

Description

Green-brown-white hypsometric color ramp for elevation maps.

Usage

```
terrain_colors(n = 100L)
```

Arguments

n Integer. Number of colors. Default 100.

Value

Character vector of hex color values.

Examples

```
cols <- terrain_colors(10)
image(matrix(1:10), col = cols)
```

terrain_profile	<i>Extract a Terrain Profile Along a Line</i>
-----------------	-----------------------------------------------

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")
```

zonal_stats	<i>Compute General Zonal Statistics from a Raster</i>
-------------	-------------------------------------------------------

Description

Extracts summary statistics for each polygon zone, including min, max, mean, median, standard deviation, and cell count.

Usage

```
zonal_stats(r, zones, id_field)
```

Arguments

r	A terra::SpatRaster .
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
- min, max, mean, median, sd: summary statistics
- cell_count: number of non-NA cells
- area_m2: zone area

Examples

```
dem <- terra::rast(system.file("extdata/dem_before.tif", package = "aboveR"))
zones <- sf::st_read(
  system.file("extdata/zones.gpkg", package = "aboveR"),
  quiet = TRUE
)
zs <- zonal_stats(dem, zones, id_field = "zone_id")
print(zs)
```

Index

bench_detection, 3
boundary_terrain_profile, 4

change_by_zone, 5
change_colors, 6
classify_highwall, 6
contour_lines, 7

detect_channels, 8

estimate_volume, 8
export_landxml, 9
export_stl, 10

flood_colors, 11
flood_depth, 12
flood_inundation, 12

has_s3_access, 13
height_above_drainage, 13
hillshade, 14

impoundment_curve, 15

kfa_county_bbox, 16
kfa_find_tiles, 16
kfa_find_tiles(), 20
kfa_list_counties, 17
kfa_read_dem, 18
kfa_read_ortho, 18
kfa_read_pointcloud, 19
kfa_stac_search, 20
kfa_tile_index, 21

lidR::LAS, 19
lidR::readLAS(), 19

pond_sedimentation, 21

reclamation_progress, 22

sf::sf, 3–5, 7–10, 12, 15, 17–22, 26, 27

slope_aspect, 23
surface_roughness, 24

terra::SpatRaster, 3–15, 18, 19, 21–27
terrain_change, 24
terrain_change(), 5
terrain_colors, 25
terrain_profile, 26

zonal_stats, 27