Package: blueterra (via r-universe)

July 20, 2026

Title Process-Oriented Geomorphometry for Submerged Terrain

Version 0.2.0

Description Derives, organizes, summarizes, and visualizes terrain metrics from bathymetric and elevation rasters for submerged-terrain geomorphometry. Tools support terra-based raster preparation, slope and aspect decomposition, terrain position, rugosity, a four-neighbor Laplacian-style index, depth-band summaries, transect extraction, isobath-corridor analysis, table preparation, visualization, and aligned user-defined metric layers. Methodological context for geomorphometric terrain analysis is provided by Lindsay (2016) <doi:10.1016/j.cageo.2016.07.003>.

License MIT + file LICENSE

URL https://el-cordero.github.io/blueterra/, https://github.com/el-cordero/blueterra

BugReports https://github.com/el-cordero/blueterra/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.1)

Imports cli, dplyr, rlang, stats, terra, tibble, utils

Suggests exactextractr, ggplot2, knitr, lintr, pkgdown, rmarkdown, sf, testthat (>= 3.0.0), units, xml2

VignetteBuilder knitr

Config/testthat/edition 3

LazyData true

Config/pak/sysreqs libgdal-dev gdal-bin libgeos-dev libproj-dev libsqlite3-dev

Repository https://el-cordero.r-universe.dev

Date/Publication 2026-07-19 23:05:38 UTC

RemoteUrl https://github.com/el-cordero/blueterra

RemoteRef HEAD

RemoteSha 2b58393be091751c42890af52f1b741143f86bf8

Contents

`add_metric_layers` *Add metric layers to an existing raster stack*

Description

Combines precomputed metric rasters with an existing metric stack after checking grid geometry.

Usage

```
add_metric_layers(
  metrics,
  ...,
  names = NULL,
  overwrite = FALSE,
  check_geometry = TRUE
)
```

Arguments

`metrics`	A `terra::SpatRaster` metric stack or raster input accepted by `as_bathy()`.
`...`	One or more `terra::SpatRaster` objects or local raster paths to add.
`names`	Optional names for the added layers.
`overwrite`	Logical. Allow added layers to replace layers with matching names in `metrics`.
`check_geometry`	Logical. Require matching CRS, extent, resolution, dimensions, and origin.

Details

Custom metrics must be on the same raster grid as the stack they extend. Geometry checks are enabled by default because summaries, PCA tables, and process-group assignments assume cell-aligned layers.

Value

A combined `terra::SpatRaster`.

See Also

derive_custom_metric(), create_metric_catalog()

Examples

```
bathy <- read_bathy(blueterra_example("bathy"))
metrics <- derive_terrain(bathy, metrics = c("slope", "tri"))
index <- derive_custom_metric(metrics, "slope_tri_index", expression = quote(slope_deg * tri))
add_metric_layers(metrics, index)
```

---

as_bathy *Coerce common raster inputs to a bathymetry raster*

Description

Accepts a terra::SpatRaster or a local raster file path and returns a terra::SpatRaster.

Usage

```
as_bathy(x, ..., check = TRUE)
```

Arguments

x	A terra::SpatRaster or local raster path.
...	Additional arguments passed to read_bathy() when x is a path.
check	Logical. If TRUE, validate the raster before returning it.

Details

as_bathy() preserves raster values and metadata when possible. It does not flip signs, project CRS, or filter depths unless another function explicitly requests that behavior.

Value

A terra::SpatRaster.

See Also

read_bathy(), validate_bathy()

Examples

```
as_bathy(blueterra_example("bathy"))
```

`assign_process_groups` *Assign metrics to process groups*

Description

Matches raster layer names or character metric names to the metric catalog.

Usage

```
assign_process_groups(
  x,
  catalog = metric_catalog(),
  groups = NULL,
  unmatched = "unassigned"
)
```

Arguments

`x`	A `terra::SpatRaster`, character vector, or data frame with metric columns.
`catalog`	Optional catalog table. Defaults to `metric_catalog()`.
`groups`	Optional named character vector mapping metric names to process groups. Supplied mappings override catalog matches for those metrics.
`unmatched`	Character value assigned to unmatched metrics.

Details

Matching uses standardized lower-case metric names. Unmatched metrics are returned with `process_group = unmatched` so users can inspect custom layers.

Value

A tibble with one row per supplied metric.

See Also

`metric_catalog()`, `summarize_process_groups()`

Examples

```
terrain <- derive_terrain(read_bathy(blueterra_example("bathy")))
assign_process_groups(terrain)
```

balance_samples *Balance samples across groups*

Description

Down-samples or samples with replacement so each group has the same number of rows.

Usage

```
balance_samples(data, group, n = NULL, replace = FALSE, seed = NULL)
```

Arguments

data	A data frame.
group	Character name of the grouping column.
n	Optional number of rows per group. Defaults to the smallest group size when replace = FALSE.
replace	Logical. Sample with replacement.
seed	Optional random seed.

Value

A tibble.

See Also

prepare_model_matrix()

Examples

```
df <- data.frame(group = rep(c("a", "b"), c(2, 5)), value = seq_len(7))
balance_samples(df, group = "group")
```

bathy_info *Summarize a bathymetric or elevation raster*

Description

Returns a compact table describing raster dimensions, extent, CRS, resolution, layer names, and value range.

Usage

```
bathy_info(x)
```

Arguments

x	A raster-like object accepted by as_bathy().

Value

A tibble with one row per raster layer.

See Also

validate_bathy(), check_bathy_crs()

Examples

```
bathy_info(blueterra_example("bathy"))
```

---

blueterra_example *Locate package example files*

Description

Returns the path to a small file installed with blueterra.

Usage

```
blueterra_example(
  name = c("hitw", "hoyo", "slope", "sampling_rectangles", "bathy", "zones", "sites",
    "synthetic_bathy", "synthetic_zones")
)

blueterra_examples()

blueterra_extdata(file = NULL)
```

Arguments

name	Example name. Use "hitw", "hoyo", or "slope" for reduced bathymetry rasters from the southwest Puerto Rico shelf margin near La Parguera; "sampling_rectangles" for the accompanying vector layer; "bathy" and "zones" as short aliases; or "synthetic_bathy" and "synthetic_zones" for test fixtures.
file	File name under inst/extdata.

Details

The primary examples are reduced analysis rasters and sampling rectangles from the southwest Puerto Rico shelf margin near La Parguera. The synthetic files are retained for numerical tests where a simple known surface is useful.

Value

A normalized local file path.

`blueterra_examples()` returns a tibble describing installed example files.

See Also

`read_bathy()`

Examples

```
hitw <- blueterra_example("hitw")
rectangles <- blueterra_example("sampling_rectangles")
file.exists(c(hitw, rectangles))
blueterra_examples()
```

---

blueterra_options *Configure blueterra runtime options*

Description

Returns or sets package options used by examples and helper functions.

Usage

```
blueterra_options(...)
```

Arguments

`...`	Named option values. Currently supported options are `blueterra.progress` and `blueterra.max_plot_cells`.

Details

Options affect only local package behavior and do not write outside paths provided by the user.

Value

A named list with current option values, invisibly when setting.

Examples

```
blueterra_options()
old <- blueterra_options(blueterra.progress = FALSE)
blueterra_options(blueterra.progress = old$blueterra.progress)
```

check_bathy_crs *Check raster CRS for geomorphometry*

Description

Reports whether a bathymetric or elevation raster has a CRS and whether that CRS appears to be geographic longitude/latitude.

Usage

```
check_bathy_crs(x, require_projected = FALSE, warn_lonlat = TRUE)
```

Arguments

x	A raster-like object accepted by as_bathy().
require_projected	Logical. If TRUE, error when the CRS is missing or longitude/latitude.
warn_lonlat	Logical. If TRUE, warn when the raster is lon/lat.

Details

Many terrain metrics can be calculated on any numeric grid, but metric interpretation is usually strongest in projected coordinate systems with linear units. Buffers, distances, slope, surface-area ratios, and focal windows are all scale-sensitive.

Value

A tibble with CRS status fields.

See Also

prepare_bathy(), project_bathy()

Examples

```
bathy <- read_bathy(blueterra_example("bathy"))
check_bathy_crs(bathy)
```

`check_bathy_units` *Check bathymetry value conventions*

Description

Summarizes raster value ranges and records the intended depth convention when supplied by the user.

Usage

```
check_bathy_units(x, units = NULL, positive_depth = NULL)
```

Arguments

`x`	A raster-like object accepted by `as_bathy()`.
`units`	Optional character label for vertical units, such as `"m"`.
`positive_depth`	Optional logical. Use `TRUE` when larger positive values mean deeper water, `FALSE` when bathymetry is stored as negative elevation, or `NULL` when unknown.

Details

This function does not infer or alter scientific meaning. It reports the observed range and the user-supplied convention so downstream workflows can be explicit.

Value

A tibble with value range and convention fields.

See Also

`set_depth_positive()`, `set_depth_negative()`

Examples

```
check_bathy_units(blueterra_example("bathy"), units = "m", positive_depth = FALSE)
```

`create_metric_catalog` *Create and validate metric catalog rows*

Description

Builds catalog rows for custom metrics and checks that metric catalogs follow the schema used by `metric_catalog()`.

Usage

```
create_metric_catalog(
  metric,
  label = metric,
  process_group,
  description = NA_character_,
  units = NA_character_,
  source_function = NA_character_,
  requires_optional_dependency = FALSE,
  scale_sensitive = TRUE,
  interpretation_notes = NA_character_
)

extend_metric_catalog(catalog = metric_catalog(), ...)

validate_metric_catalog(catalog)
```

Arguments

`metric`	Metric layer name.
`label`	Human-readable metric label.
`process_group`	Process-group label.
`description`	Metric description.
`units`	Metric units.
`source_function`	Function or workflow that produced the metric.
`requires_optional_dependency`	Logical. Whether the metric requires an optional package.
`scale_sensitive`	Logical. Whether the metric is sensitive to grid resolution or focal scale.
`interpretation_notes`	Notes on interpretation.
`catalog`	Existing metric catalog.
`...`	One or more catalog rows or tibbles to append.

Details

Custom process groups are user-defined terrain-form categories. The catalog records how custom layers should be grouped and interpreted without changing raster values.

Value

A tibble with the same columns as `metric_catalog()`.

See Also

`assign_process_groups()`, `summarize_process_groups()`

Examples

```
row <- create_metric_catalog(
  metric = "slope_tri_index",
  process_group = "custom_relief",
  description = "Product of local slope and terrain ruggedness index."
)
validate_metric_catalog(row)
extend_metric_catalog(metric_catalog(), row)
```

---

crop_bathy *Crop, mask, resample, and project bathymetry*

Description

Small wrappers around `terra` raster-preparation operations with bathymetry input validation.

Usage

```
crop_bathy(x, extent, filename = "", overwrite = FALSE)

mask_bathy(x, mask, filename = "", overwrite = FALSE)

resample_bathy(x, y, method = "bilinear", filename = "", overwrite = FALSE)

project_bathy(x, crs, method = "bilinear", filename = "", overwrite = FALSE)
```

Arguments

`x`	A raster-like object accepted by `as_bathy()`.
`extent`	Crop extent.
`filename`	Optional output raster path.
`overwrite`	Logical. Allow overwriting `filename`.
`mask`	Polygon/vector mask.

y	Template raster used for resampling.
method	Interpolation method passed to terra.
crs	Target CRS.

Details

These helpers preserve depth sign and raster values except for interpolation effects introduced by resampling or projection.

Value

A terra::SpatRaster.

See Also

prepare_bathy()

Examples

```
bathy <- read_bathy(blueterra_example("bathy"))
zones <- terra::vect(blueterra_example("zones"))
crop_bathy(bathy, terra::ext(zones[zones$site_id == "slope", ]))
```

---

depth_filter *Filter or convert bathymetric depth values*

Description

Keeps raster values within a depth or elevation range, or explicitly changes sign convention.

Usage

```
depth_filter(
  x,
  depth_range,
  positive_depth = NULL,
  filename = "",
  overwrite = FALSE
)

invert_depth(x, filename = "", overwrite = FALSE)

set_depth_positive(x, filename = "", overwrite = FALSE)

set_depth_negative(x, filename = "", overwrite = FALSE)
```

Arguments

x	A raster-like object accepted by as_bathy().
depth_range	Numeric length-two retained range.
positive_depth	Optional logical. If TRUE, filtering is applied to absolute depth values. If FALSE or NULL, filtering is applied to the stored raster values.
filename	Optional output raster path.
overwrite	Logical. Allow overwriting filename.

Details

depth_filter() does not flip signs. Use set_depth_positive(), set_depth_negative(), or invert_depth() for explicit sign changes.

Value

A terra::SpatRaster.

See Also

check_bathy_units()

Examples

```
bathy <- read_bathy(blueterra_example("bathy"))
depth_filter(bathy, c(-80, -30))
set_depth_positive(bathy)
```

---

derive_bpi *Derive bathymetric position index*

Description

Computes a local bathymetric position index from the difference between each cell and its focal-neighborhood mean.

Usage

```
derive_bpi(
  x,
  inner_radius = NULL,
  outer_radius = NULL,
  window = NULL,
  scale = c("fine", "broad", "custom"),
  normalize = FALSE,
  filename = "",
```

```
  overwrite = FALSE,
  ...
)

derive_multiscale_bpi(
  x,
  windows = c(3, 11),
  normalize = FALSE,
  filename = "",
  overwrite = FALSE,
  ...
)
```

Arguments

x	A raster-like object accepted by as_bathy().
inner_radius	Optional inner radius for an annulus window in map units.
outer_radius	Optional outer radius for an annulus window in map units.
window	Optional odd integer square window size in cells.
scale	Preset scale when window and outer_radius are not supplied.
normalize	Logical. If TRUE, divide BPI by local focal standard deviation. Zero-variance or unavailable focal neighborhoods return NA.
filename	Optional output raster path.
overwrite	Logical. Allow overwriting filename.
...	Reserved for future extensions.
windows	Integer vector of odd square window sizes.

Details

The calculation is cell value - focal mean. Positive values therefore mean higher-than-neighborhood values when the raster is elevation-like. For positive-depth rasters, interpretation is reversed unless users convert the sign convention first. Square windows are measured in cells and include the focal cell. Annular windows are measured in map units, require a projected CRS with linear units, and use separate x- and y-cell resolutions when cells are not square. At raster edges and next to missing cells, BPI uses the available cells in its partial focal support; a missing focal value remains missing.

Value

A single-layer terra::SpatRaster.

See Also

derive_multiscale_bpi(), derive_tpi()

Examples

```
bathy <- read_bathy(blueterra_example("bathy"))
derive_bpi(bathy, window = 5)
```

`derive_custom_metric` *Derive a custom metric from raster layers*

Description

Creates a single custom metric layer from an expression or a user-supplied R function.

Usage

```
derive_custom_metric(
  metrics,
  name,
  expression = NULL,
  fun = NULL,
  ...,
  overwrite = FALSE
)
```

Arguments

`metrics`	A `terra::SpatRaster` metric stack.
`name`	Output layer name.
`expression`	Optional quoted R expression evaluated with raster layers available by name.
`fun`	Optional function called as `fun(metrics, ...)`. The function must return a single-layer `terra::SpatRaster`.
`...`	Additional arguments passed to `fun`.
`overwrite`	Logical. Allow `name` to match an existing layer in `metrics`.

Details

Expressions must be quoted, for example `quote(slope_deg * rugosity_vrm_3x3)`. Character strings are not evaluated. This keeps the workflow explicit and avoids treating text as code.

Value

A single-layer `terra::SpatRaster` named `name`.

See Also

`add_metric_layers()`

Examples

```
bathy <- read_bathy(blueterra_example("bathy"))
metrics <- derive_terrain(bathy, metrics = c("slope", "tri", "bpi"))
derive_custom_metric(metrics, "slope_tri_index", expression = quote(slope_deg * tri))

relief <- derive_custom_metric(metrics, "relief_index", fun = function(r) {
  out <- r[["tri"]] + abs(r[["bpi_3x3"]])
  names(out) <- "relief_index"
  out
})
relief
```

`derive_metric_stack` *Build a stack of terrain metrics*

Description

Computes selected metrics and returns them as a named raster stack.

Usage

```
derive_metric_stack(
  x,
  metrics = "default",
  scales = NULL,
  units = c("degrees", "radians"),
  neighbors = 8,
  positive_depth = NULL,
  filename = "",
  overwrite = FALSE,
  progress = TRUE
)
```

Arguments

`x`	A raster-like object accepted by `as_bathy()`.
`metrics`	Character vector of metrics, or `"default"`.
`scales`	Optional BPI/TPI window sizes used when multiscale BPI is requested.
`units`	Slope and aspect units, `"degrees"` or `"radians"`.
`neighbors`	Neighborhood passed to `terra::terrain()`.
`positive_depth`	Optional logical documenting the input sign convention. Metric values are not sign-flipped unless a specific function says so.
`filename`	Optional output raster path.
`overwrite`	Logical. Allow overwriting `filename`.
`progress`	Logical. Reserved for long-running workflows.

Details

"default" currently expands to bathymetry, slope, aspect, northness, eastness, hillshade, roughness, TRI, TPI, fine BPI, broad BPI, curvature, and surface-area ratio. All metrics are derived locally from the supplied raster.

Value

A terra::SpatRaster.

See Also

derive_terrain()

Examples

```
bathy <- read_bathy(blueterra_example("bathy"))
derive_metric_stack(bathy, metrics = "default")
```

---

derive_slope *Derive individual terrain metrics*

Description

Convenience wrappers around terra terrain functions and lightweight geomorphometry formulas.

Usage

```
derive_slope(
  x,
  units = c("degrees", "radians"),
  neighbors = 8,
  filename = "",
  overwrite = FALSE
)

derive_aspect(
  x,
  units = c("degrees", "radians"),
  neighbors = 8,
  filename = "",
  overwrite = FALSE
)

derive_northness(x, neighbors = 8, filename = "", overwrite = FALSE)

derive_eastness(x, neighbors = 8, filename = "", overwrite = FALSE)
```

```
derive_hillshade(
  x,
  angle = 45,
  direction = 315,
  neighbors = 8,
  filename = "",
  overwrite = FALSE
)

derive_roughness(x, filename = "", overwrite = FALSE)

derive_tri(x, filename = "", overwrite = FALSE)

derive_tpi(x, filename = "", overwrite = FALSE)

derive_rugosity(x, window = 3, neighbors = 8, filename = "", overwrite = FALSE)

derive_curvature(x, filename = "", overwrite = FALSE)

derive_surface_area_ratio(x, neighbors = 8, filename = "", overwrite = FALSE)
```

Arguments

`x`	A raster-like object accepted by `as_bathy()`.
`units`	Units for slope or aspect: `"degrees"` or `"radians"`.
`neighbors`	Neighborhood size passed to `terra::terrain()`.
`filename`	Optional output raster path.
`overwrite`	Logical. Allow overwriting `filename`.
`angle`	Illumination angle for hillshade.
`direction`	Illumination direction for hillshade.
`window`	Odd integer focal-window size for local metrics.

Details

Functions preserve the input raster sign. For example, slope depends on the magnitude of local gradients, while BPI/TPI interpretation depends on whether the raster stores elevation-like values or positive depth.

`derive_rugosity()` computes a vector-ruggedness-measure-style index from local slope and aspect vectors. Its focal means use available derivative cells, including partial focal support adjacent to a derivative boundary or missing data, but the outermost cells can remain missing because slope and aspect themselves require neighbouring elevation values.

`derive_curvature()` computes a four-neighbor Laplacian-style index. It is not plan, profile, mean, or Gaussian curvature, is not scaled by cell dimensions, and is consequently strongly resolution-dependent. `derive_surface_area_ratio()` is a slope-secant approximation, `1 / cos(slope)`, rather than a triangulated or directly measured benthic surface area. It clamps near-zero cosine values to avoid pathological ratios at extreme slopes.

Value

A single-layer `terra::SpatRaster`, except where documented otherwise.

See Also

`derive_terrain()`, `derive_bpi()`

Examples

```
bathy <- read_bathy(blueterra_example("bathy"))
derive_slope(bathy)
derive_northness(bathy)
```

---

derive_terrain *Derive bathymetric terrain metrics*

Description

Computes one or more terrain metrics from a user-supplied bathymetric or elevation raster.

Usage

```
derive_terrain(
  x,
  metrics = "default",
  scales = NULL,
  units = c("degrees", "radians"),
  neighbors = 8,
  positive_depth = NULL,
  filename = "",
  overwrite = FALSE,
  progress = TRUE
)
```

Arguments

`x`	A raster-like object accepted by `as_bathy()`.
`metrics`	Character vector of metrics, or `"default"`.
`scales`	Optional BPI/TPI window sizes used when multiscale BPI is requested.
`units`	Slope and aspect units, `"degrees"` or `"radians"`.
`neighbors`	Neighborhood passed to `terra::terrain()`.
`positive_depth`	Optional logical documenting the input sign convention. Metric values are not sign-flipped unless a specific function says so.
`filename`	Optional output raster path.
`overwrite`	Logical. Allow overwriting `filename`.
`progress`	Logical. Reserved for long-running workflows.

Details

Terrain metrics are scale-sensitive. Slope, curvature, rugosity, TPI, BPI, roughness, and surface-area style metrics depend on grid resolution, smoothing, and focal-window size. Use projected coordinate systems when distances or slopes are interpreted in real linear units.

Value

A `terra::SpatRaster` containing named metric layers.

See Also

`derive_metric_stack()`, `derive_bpi()`, `metric_catalog()`

Examples

```
bathy <- read_bathy(blueterra_example("bathy"))
terrain <- derive_terrain(bathy, metrics = c("slope", "aspect", "bpi"))
names(terrain)
```

---

estimate_surface_orientation

Estimate mean surface orientation from a bathymetric raster

Description

Estimates the mean aspect direction of a raster surface and converts that compass bearing to the line-angle convention used by `make_transects()`.

Usage

```
estimate_surface_orientation(
  bathy,
  area = NULL,
  orientation_weight = c("slope", "none"),
  min_slope = 0,
  return = c("transect_angle", "bearing", "both")
)
```

Arguments

`bathy`	A single-layer bathymetric or elevation raster, or a raster input accepted by `as_bathy()`.
`area`	Optional polygon area used to crop and mask `bathy` before estimating orientation.
`orientation_weight`	Weighting method. `"slope"` weights aspect components by slope magnitude; `"none"` averages finite aspect cells equally.

min_slope	Minimum slope in degrees retained when orientation_weight = "slope".
return	Return type: "transect_angle" for the mathematical line angle used by make_transects(), "bearing" for the mean compass aspect, or "both" for a tibble containing both values and the number of cells used.

Details

Aspect is treated as a compass bearing where northness is cos(aspect) and eastness is sin(aspect). Mean circular components are converted to a compass bearing with atan2(eastness, northness). Transect lines are undirected, so the line angle is normalized to [0, 180). A south-facing mean aspect near 180 degrees therefore produces a transect angle near 90 degrees.

Value

A numeric angle for "transect_angle" or "bearing", or a tibble with bearing_deg, transect_angle_deg, orientation_weight, min_slope, and n_orientation_cells when return = "both".

See Also

make_transects(), derive_aspect(), derive_slope()

Examples

```
bathy <- read_bathy(blueterra_example("hitw"))
zones <- terra::vect(blueterra_example("zones"))
hitw <- zones[zones$site_id == "hitw", ]
estimate_surface_orientation(bathy, hitw)
```

---

extract_isobath_corridors

Extract terrain cells in isobath corridors

Description

Extracts raster cell values under corridor polygons.

Usage

```
extract_isobath_corridors(metrics, corridors, ...)
```

Arguments

metrics	A metric raster stack.
corridors	Corridor polygons from make_isobath_corridors() or another polygon source.
...	Additional arguments passed to terra::extract().

Value

A tibble with corridor identifiers and extracted raster values.

See Also

summarize_isobath_terrain(), summarize_terrain()

Examples

```
bathy <- read_bathy(blueterra_example("bathy"))
terrain <- derive_terrain(bathy, metrics = c("slope", "bpi"))
corridors <- make_isobath_corridors(bathy, depths = -50, width = 5)
extract_isobath_corridors(terrain, corridors)
```

---

extract_isobaths *Extract isobaths from a raster*

Description

Converts raster contour levels to line features.

Usage

```
extract_isobaths(
  x,
  depths,
  positive_depth = NULL,
  as = c("SpatVector", "sf"),
  ...
)
```

Arguments

x	A raster-like object accepted by as_bathy().
depths	Numeric contour levels.
positive_depth	Optional logical. If TRUE, depths are converted to positive values. If FALSE, they are converted to negative values. If NULL, depths are used exactly as supplied.
as	Output type: "SpatVector" or "sf".
...	Additional arguments reserved for future extensions.

Details

Depth convention is explicit. For rasters stored as negative elevation, either pass negative depths or set positive_depth = FALSE when passing positive depth labels.

Value

Isobath line features as `terra::SpatVector` by default.

See Also

`make_isobath_corridors()`

Examples

```
bathy <- read_bathy(blueterra_example("bathy"))
extract_isobaths(bathy, depths = c(-40, -60))
```

`extract_terrain_points`

Extract terrain values at points

Description

Extracts raster values at point locations.

Usage

```
extract_terrain_points(metrics, points, method = "bilinear", ...)
```

Arguments

`metrics`	A metric raster stack.
`points`	Points as `sf`, `terra::SpatVector`, or a local vector path.
`method`	Extraction method passed to `terra::extract()`.
`...`	Additional arguments passed to `terra::extract()`.

Value

A tibble with point attributes and raster values.

See Also

`sample_terrain_cells()`

Examples

```
bathy <- read_bathy(blueterra_example("bathy"))
zones <- terra::vect(blueterra_example("zones"))
pts <- terra::centroids(zones)
extract_terrain_points(bathy, pts)
```

`make_isobath_corridors` *Build isobath corridors*

Description

Buffers isobath contour lines to create depth-following corridor polygons.

Usage

```
make_isobath_corridors(
  x,
  depths,
  width,
  smooth = FALSE,
  as = c("SpatVector", "sf"),
  positive_depth = NULL,
  ...
)
```

Arguments

`x`	A raster-like object accepted by `as_bathy()`.
`depths`	Numeric contour levels.
`width`	One-sided buffer distance in map units. The nominal full corridor width is twice this value.
`smooth`	Logical. If `TRUE`, apply a zero-width buffer after buffering to clean polygon topology.
`as`	Output type: `"SpatVector"` or `"sf"`.
`positive_depth`	Optional logical. If `TRUE`, `depths` are converted to positive values. If `FALSE`, they are converted to negative values. If `NULL`, `depths` are used exactly as supplied.
`...`	Additional arguments reserved for future extensions.

Details

`width` is interpreted as a one-sided buffer distance in the CRS map units. Projected CRS are strongly recommended. If the raster uses longitude/latitude, the function warns before buffering because distance interpretation may be misleading. Corridors are returned as independent buffers and can overlap; their summaries are therefore not mutually exclusive or additive.

Value

Isobath corridor polygons as `terra::SpatVector` by default.

See Also

`extract_isobaths()`, `summarize_isobath_terrain()`

Examples

```
bathy <- read_bathy(blueterra_example("bathy"))
make_isobath_corridors(bathy, depths = c(-40, -60), width = 5)
```

`make_transects` *Create transects across polygon zones*

Description

Builds regularly spaced straight transects through polygon features.

Usage

```
make_transects(
  area,
  spacing,
  angle = NULL,
  bathy = NULL,
  orientation = c("auto", "surface", "bbox", "manual"),
  orientation_weight = c("slope", "none"),
  min_slope = 0,
  length = NULL,
  id_field = NULL,
  as = c("SpatVector", "sf")
)
```

Arguments

`area`	A `terra::SpatVector`, local vector path, or optional `sf` polygon object.
`spacing`	Distance between transects in map units.
`angle`	Optional transect direction in degrees counterclockwise from the projected x axis. When supplied, this manual value overrides orientation estimation.
`bathy`	Optional bathymetry raster used to estimate a terrain-based transect orientation when `angle = NULL`.
`orientation`	Orientation strategy. `"auto"` uses surface orientation when `bathy` is supplied and otherwise falls back to the historical horizontal line angle with a warning. `"surface"` requires `bathy`, `"bbox"` uses the polygon bounding-box axis, and `"manual"` requires `angle`.
`orientation_weight`	Weighting for surface orientation. `"slope"` gives steeper cells more influence; `"none"` averages aspect components equally.
`min_slope`	Minimum slope, in degrees, used when `orientation_weight = "slope"`.
`length`	Optional transect length in map units. If `NULL`, a length based on the polygon bounding box is used.
`id_field`	Optional field in `area` used as the zone identifier.
`as`	Output type: `"SpatVector"` or `"sf"`.

Details

Transect spacing and length are interpreted in map units. Use a projected CRS for metric distances. Candidate lines are created through each polygon bounding box and clipped to the polygon with `terra::intersect()`.

With `angle = NULL` and a supplied `bathy` raster, the default transect direction is estimated from the mean surface aspect within each polygon. Aspect is converted to northness and eastness, averaged as circular components, and converted to the mathematical line angle used for transect generation. For example, a south-facing mean aspect near 180 degrees yields a transect angle near 90 degrees, producing north-south transects in projected coordinates. The estimated angle, source metadata, and circular resultant length are stored on the output lines. Resultant lengths near zero indicate weakly concentrated aspects and an unstable mean direction.

Value

A `terra::SpatVector` by default. If `as = "sf"`, an `sf` object is returned and the optional `sf` package must be installed.

See Also

`estimate_surface_orientation()`, `sample_transects()`, `extract_cross_sections()`

Examples

```
zones <- terra::vect(blueterra_example("zones"))
bathy <- read_bathy(blueterra_example("bathy"))
transects <- make_transects(zones[1, ], spacing = 50, bathy = bathy)
transects

manual <- make_transects(zones[1, ], spacing = 50, angle = 90)
manual[, c("angle_deg", "angle_source")]
```

`metric_catalog` *Terrain metric catalog*

Description

Returns a catalog of terrain metrics, labels, process groups, assumptions, and source functions used by `blueterra`.

Usage

```
metric_catalog()

process_groups()
```

Details

The catalog is an interpretation aid, not a claim that terrain metrics directly measure ecological or oceanographic processes. Groups such as seafloor aspect, slope gradient, accumulation potential, seafloor position, seafloor rugosity, downslope pathway proximity, transport potential, and curvature describe terrain form and terrain-routing proxies. Accumulation, pathway, and transport interpretations require separate validation and are not direct current or sediment-flux measurements.

Value

A tibble with columns `metric`, `label`, `process_group`, `description`, `units`, `source_function`, `requires_optional_dependency`, `scale_sensitive`, and `interpretation_notes`.

See Also

`derive_terrain()`, `assign_process_groups()`

Examples

```
metric_catalog()
process_groups()
```

---

metric_catalog_data *Terrain metric catalog data*

Description

A compact table describing the metrics returned by core `blueterra` functions and their process-oriented interpretation groups.

Usage

```
metric_catalog_data
```

Format

A tibble with columns:

metric Stable metric name.

label Human-readable label.

process_group Process-oriented terrain group.

description Metric description.

units Expected units.

source_function Function that derives the metric.

requires_optional_dependency Whether an optional dependency is needed.

scale_sensitive Whether interpretation depends on raster scale.

interpretation_notes Important interpretation notes.

Details

Use `metric_catalog()` to retrieve this table in normal workflows.

`pca_axis_labels` *Label PCA axes with variance and dominant loadings*

Description

Builds axis labels for `plot_process_pca()` from a `terrain_pca()` object.

Usage

```
pca_axis_labels(
  pca,
  components = c("PC1", "PC2"),
  top_n = 2,
  unique = TRUE,
  include_variance = TRUE
)
```

Arguments

`pca`	Output from `terrain_pca()`.
`components`	Components to label.
`top_n`	Number of high-loading variables included per component.
`unique`	Logical. Avoid repeating the same dominant variable across component labels when possible.
`include_variance`	Logical. Include percent variance explained.

Details

Variables are ranked by absolute loading for each component. These labels are descriptive aids for ordination plots; they do not replace inspection of the full loading table.

Value

A named character vector of axis labels.

See Also

`terrain_pca()`, `plot_process_pca()`

Examples

```
df <- data.frame(a = rnorm(10), b = rnorm(10), c = rnorm(10))
pca_axis_labels(terrain_pca(df))
```

plot_bathy *Plot bathymetry and terrain rasters*

Description

Creates ggplot2 maps for bathymetry and derived terrain metrics using terra rasters and vectors. Hillshade can be drawn as a semitransparent visual relief layer, and bathymetric contours, sampling rectangles, transects, or other terra::SpatVector geometries can be overlaid.

Usage

```
plot_bathy(
  x,
  hillshade = TRUE,
  hillshade_alpha = 0.3,
  contours = TRUE,
  contour_interval = NULL,
  contour_color = "white",
  contour_linewidth = 0.25,
  vectors = NULL,
  vector_color = "white",
  vector_linewidth = 0.5,
  labels = NULL,
  label_field = NULL,
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  legend_title = "Bathymetry",
  max_cells = getOption("blueterra.max_plot_cells", 10000)
)

plot_metric(
  x,
  metric = NULL,
  bathy = NULL,
  hillshade = TRUE,
  hillshade_alpha = 0.3,
  contours = FALSE,
  contour_interval = NULL,
  contour_color = "white",
  contour_linewidth = 0.25,
  vectors = NULL,
  vector_color = "white",
  vector_linewidth = 0.5,
  labels = NULL,
  label_field = NULL,
  title = NULL,
```

```
  subtitle = NULL,
  caption = NULL,
  legend_title = NULL,
  max_cells = getOption("blueterra.max_plot_cells", 10000)
)

plot_terrain_map(
  bathy,
  metric = NULL,
  vectors = NULL,
  contours = TRUE,
  contour_interval = 20,
  hillshade = TRUE,
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  ...
)

plot_hillshade(x, max_cells = getOption("blueterra.max_plot_cells", 10000))

plot_sampling_rectangles(bathy, rectangles, label_field = "site_id", ...)

plot_transects(bathy, transects, color_by = NULL, show_legend = FALSE, ...)

plot_metric_stack(x, max_cells = getOption("blueterra.max_plot_cells", 10000))
```

Arguments

x	A raster-like object accepted by as_bathy().
hillshade	Logical. Draw hillshade as a visual relief layer.
hillshade_alpha	Maximum alpha for the hillshade shadow overlay.
contours	Logical. Draw contour lines from bathy or x.
contour_interval	Optional contour interval in raster units.
contour_color	Contour line color.
contour_linewidth	Contour line width.
vectors	Optional terra::SpatVector, sf object, or local vector path drawn over the raster.
vector_color	Vector outline color.
vector_linewidth	Vector outline width.
labels	Optional label source. Use TRUE to label vectors, or supply a vector object/path.

label_field	Optional field used for vector labels.
title, subtitle, caption	Plot text passed to ggplot2::labs().
legend_title	Optional raster legend title.
max_cells	Maximum raster cells used for plotting.
metric	Optional metric raster or a layer name/index in bathy.
bathy	Optional bathymetry raster used to derive hillshade and contours when x is a metric raster.
...	Additional plotting options passed from convenience wrappers to plot_bathy() or plot_metric().
rectangles	Sampling rectangles or polygon zones.
transects	Transect line geometry.
color_by	Optional transect attribute used to color lines, such as "transect_id".
show_legend	Logical. Show a legend when color_by is supplied.

Details

Plotting functions require ggplot2, which is suggested rather than imported. Large rasters are regularly sampled before plotting to keep examples and interactive work responsive. Hillshade is used only as a visual relief layer; it is not a terrain predictor unless a user explicitly derives and analyzes it as one.

Value

A ggplot object.

See Also

derive_terrain(), plot_metric_stack()

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  bathy <- read_bathy(blueterra_example("bathy"))
  zones <- terra::vect(blueterra_example("zones"))
  plot_bathy(bathy, vectors = zones, labels = TRUE, label_field = "site_id")
}
```

`plot_cross_sections` *Plot sampled cross-sections*

Description

Plots raster values against transect distance.

Usage

```
plot_cross_sections(
  samples,
  value_col = NULL,
  group_col = "transect_id",
  color_col = NULL,
  show_legend = TRUE,
  points = FALSE,
  mean_profile = FALSE,
  mean_profile_na_rm = TRUE,
  normalize_distance = FALSE,
 profile_direction = c("top_to_bottom", "bottom_to_top", "as_sampled", "max_to_min",
    "min_to_max", "high_to_low", "low_to_high"),
  positive_depth = NULL,
  depth_increases_down = TRUE,
  title = NULL,
  subtitle = NULL,
  caption = NULL
)
```

Arguments

`samples`	Output from `sample_transects()`.
`value_col`	Optional value column.
`group_col`	Grouping column for transect lines.
`color_col`	Optional column used to color profiles. Defaults to `group_col`.
`show_legend`	Logical. Show the line-color legend.
`points`	Logical. Draw sample points over profile lines.
`mean_profile`	Logical. Overlay an interpolated mean profile across transects.
`mean_profile_na_rm`	Logical. Remove missing interpolated profile values when averaging the mean profile. The default, `TRUE`, lets the mean profile use the full available distance range rather than stopping where the shortest transect ends. Set to `FALSE` to require every profile to contribute at each distance.
`normalize_distance`	Logical. Plot distance as 0-1 normalized position along each transect.

`profile_direction`
: Direction used to orient distance before plotting. `"top_to_bottom"` (the default) orients bathymetric or elevation profiles from the shallow or top endpoint toward the deeper or bottom endpoint. With negative-elevation bathymetry this means higher numeric values to lower numeric values. With positive-depth bathymetry, set `positive_depth = TRUE` so the profile runs from lower depth values to higher depth values. `"bottom_to_top"` reverses that convention. `"max_to_min"` and `"min_to_max"` provide explicit numeric endpoint controls for metrics, and `"as_sampled"` preserves the sampled line order. Legacy values `"high_to_low"` and `"low_to_high"` are accepted as aliases for `"top_to_bottom"` and `"bottom_to_top"`.

`positive_depth`
: Logical depth convention for `value_col`. This affects top-to-bottom profile orientation for depth-like variables and y-axis display for positive-depth values.

`depth_increases_down`
: Logical. If `TRUE`, positive-depth profiles are plotted with a reversed y-axis so larger depths appear lower in the panel.

`title`, `subtitle`, `caption`
: Plot text.

Value

A `ggplot` object.

See Also

`sample_transects()`, `plot_depth_profile()`

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  bathy <- read_bathy(blueterra_example("bathy"))
  zones <- terra::vect(blueterra_example("zones"))
  transects <- make_transects(zones[1, ], spacing = 100, bathy = bathy)
  samples <- sample_transects(transects, bathy, n = 5)
  plot_cross_sections(samples)
}
```

---

plot_depth_profile *Plot a depth profile*

Description

Plots sampled raster values along a transect profile or against depth.

Usage

```
plot_depth_profile(
  data,
  distance_col = "distance",
  depth_col = NULL,
  value_col = NULL,
  group_col = NULL,
  points = TRUE,
  line = TRUE,
  profile_layout = c("auto", "distance", "metric_by_depth"),
 profile_direction = c("top_to_bottom", "bottom_to_top", "as_sampled", "max_to_min",
    "min_to_max", "high_to_low", "low_to_high"),
  positive_depth = NULL,
  depth_increases_down = TRUE,
  title = NULL,
  subtitle = NULL,
  caption = NULL
)
```

Arguments

`data`	A data frame.
`distance_col`	Distance column name.
`depth_col`	Depth or elevation column name. If `NULL`, a depth-like column is inferred where needed.
`value_col`	Value column to plot. Use this for sampled variables such as bathymetry, slope, rugosity, BPI, or curvature.
`group_col`	Optional grouping column.
`points`	Logical. Draw profile points.
`line`	Logical. Draw profile lines when at least two finite samples are available.
`profile_layout`	Plot layout. `"auto"` uses a distance profile when only one value column is supplied and uses a metric-by-depth profile when both `depth_col` and `value_col` identify different columns. `"distance"` plots distance on x and the selected value on y. `"metric_by_depth"` plots the selected metric on x and depth or elevation on y.
`profile_direction`	Direction used to orient distance before plotting. `"top_to_bottom"` (the default) orients bathymetric or elevation profiles from the shallow or top endpoint toward the deeper or bottom endpoint. With negative-elevation bathymetry this means higher numeric values to lower numeric values. With positive-depth bathymetry, set `positive_depth = TRUE` so the profile runs from lower depth values to higher depth values. `"bottom_to_top"` reverses that convention. `"max_to_min"` and `"min_to_max"` provide explicit numeric endpoint controls for metrics, and `"as_sampled"` preserves the sampled line order. Legacy values `"high_to_low"` and `"low_to_high"` are accepted as aliases for `"top_to_bottom"` and `"bottom_to_top"`.

`positive_depth` Logical depth convention for `value_col`. This affects top-to-bottom profile orientation for depth-like variables and y-axis display for positive-depth values.

`depth_increases_down`
Logical. If `TRUE`, positive-depth profiles are plotted with a reversed y-axis so larger depths appear lower in the panel.

`title`, `subtitle`, `caption`
Plot text.

Details

With `profile_layout = "distance"`, the y-axis can be any sampled raster variable: elevation, depth, slope, rugosity, BPI, curvature, or a custom metric. With `profile_layout = "metric_by_depth"`, depth or elevation is placed on the y-axis and the selected terrain metric is placed on the x-axis. Metadata columns such as transect angle, offset, width, and height are ignored during automatic value-column inference.

Value

A `ggplot` object.

See Also

`sample_transects()`

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  df <- data.frame(distance = 1:5, depth = -c(10, 12, 20, 25, 30))
  plot_depth_profile(df, depth_col = "depth")

  metric_df <- data.frame(
    distance = 1:5,
    bathy_m = -c(10, 12, 20, 25, 30),
    slope_deg = c(4, 6, 9, 12, 15)
  )
  plot_depth_profile(metric_df, depth_col = "bathy_m", value_col = "slope_deg")
}
```

---

plot_isobath_corridors

Plot isobath corridors

Description

Plots isobath corridor polygons, optionally over hillshaded bathymetry with contour lines. The hillshade layer is visual context only.

Usage

```
plot_isobath_corridors(
  corridors,
  bathy = NULL,
  isobaths = NULL,
  hillshade = TRUE,
  background_contours = FALSE,
  source_isobaths = TRUE,
  isobath_color = "black",
  isobath_linewidth = 0.6,
  corridor_color = "white",
  corridor_linewidth = 0.45,
  corridor_fill = NA,
  corridor_alpha = 0.2,
  labels = TRUE,
  label_field = NULL,
  title = "Isobath Corridors",
  subtitle = NULL,
  caption = NULL,
  contours = NULL,
  ...
)
```

Arguments

`corridors`	Corridor polygons.
`bathy`	Optional raster background.
`isobaths`	Optional source isobath lines to draw over the corridors.
`hillshade`	Logical. Draw hillshade from `bathy` when available.
`background_contours`	Logical. Draw general bathymetric background contours. Defaults to `FALSE` to keep corridor figures readable.
`source_isobaths`	Logical. Draw the source isobaths used to create the corridors.
`isobath_color, isobath_linewidth`	Source-isobath line styling.
`corridor_color, corridor_linewidth, corridor_fill, corridor_alpha`	Corridor polygon styling.
`labels`	Logical. Label corridors with `label_field`.
`label_field`	Attribute used for labels. Defaults to `depth_label` when present, otherwise `contour_value`.
`title, subtitle, caption`	Plot text.
`contours`	Deprecated alias for `background_contours`.
`...`	Additional arguments passed to `plot_bathy()`.

Value

A ggplot object.

See Also

make_isobath_corridors(), plot_bathy()

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  bathy <- read_bathy(blueterra_example("bathy"))
  corridors <- make_isobath_corridors(bathy, depths = -50, width = 5)
  isobaths <- extract_isobaths(bathy, depths = -50)
  plot_isobath_corridors(corridors, bathy, isobaths = isobaths)
}
```

---

plot_process_density *Plot process density*

Description

Plots density curves for one or more process-group metrics.

Usage

```
plot_process_density(data, value, group = NULL)
```

Arguments

data	A data frame of terrain values.
value	Character name of the numeric value column.
group	Optional grouping column.

Value

A ggplot object.

See Also

assign_process_groups()

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  df <- data.frame(value = rnorm(20), process = rep(c("a", "b"), each = 10))
  plot_process_density(df, value = "value", group = "process")
}
```

plot_process_pca *Plot terrain PCA*

Description

Plots the first two principal component score axes from terrain_pca().

Usage

```
plot_process_pca(
  pca,
  color_col = NULL,
  shape_col = NULL,
  label_loadings = TRUE,
  loading_arrows = TRUE,
  top_loadings = 3,
  axis_labels = c("loadings", "variance", "plain"),
  title = NULL,
  subtitle = NULL,
  caption = NULL
)
```

Arguments

pca	Output from terrain_pca().
color_col	Optional score metadata column used for point color.
shape_col	Optional score metadata column used for point shape.
label_loadings	Logical. Label the largest loading vectors.
loading_arrows	Logical. Draw loading arrows scaled into score space.
top_loadings	Number of high-magnitude loading vectors to draw or label.
axis_labels	Axis label style: include dominant loadings and variance, variance only, or plain component names.
title, subtitle, caption	Plot text.

Value

A ggplot object.

See Also

terrain_pca()

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  df <- data.frame(a = rnorm(20), b = rnorm(20), c = rnorm(20))
  plot_process_pca(terrain_pca(df))
}
```

`plot_terrain_summary` *Plot terrain summaries*

Description

Plots a summary column from `summarize_terrain()` or related functions.

Usage

```
plot_terrain_summary(summary, value, group = NULL)
```

Arguments

`summary`	A summary data frame.
`value`	Summary value column.
`group`	Optional x-axis grouping column. Defaults to `zone_id` when present.

Value

A `ggplot` object.

See Also

`summarize_terrain()`

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  df <- data.frame(zone_id = 1:3, slope_mean = c(5, 7, 2))
  plot_terrain_summary(df, value = "slope_mean")
}
```

prepare_bathy *Prepare a bathymetric or elevation raster*

Description

Applies common preprocessing steps for terrain analysis: optional projection, resampling, cropping, masking, depth filtering, sign conversion, smoothing, and optional file-backed output.

Usage

```
prepare_bathy(
  x,
  crs = NULL,
  resolution = NULL,
  extent = NULL,
  mask = NULL,
  depth_range = NULL,
  positive_depth = NULL,
  method = "bilinear",
  smooth = FALSE,
  smooth_window = 3,
  filename = "",
  overwrite = FALSE
)
```

Arguments

x	A raster-like object accepted by as_bathy().
crs	Optional target CRS passed to project_bathy().
resolution	Optional target cell size. A single value is used for both axes; two values are used as x and y resolution.
extent	Optional crop extent: SpatExtent, numeric xmin, xmax, ymin, ymax, raster, or vector object.
mask	Optional polygon/vector mask.
depth_range	Optional numeric length-two range of retained depths or elevations.
positive_depth	Optional logical. If TRUE, convert output to positive depth. If FALSE, convert output to negative depth/elevation. If NULL, preserve sign convention.
method	Resampling and projection method passed to terra.
smooth	Logical. If TRUE, apply smooth_bathy().
smooth_window	Odd integer focal-window size for smoothing.
filename	Optional output raster path.
overwrite	Logical. Allow overwriting filename.

Details

`prepare_bathy()` never automatically reprojects or flips depth signs. Those operations occur only when `crs` or `positive_depth` are supplied. Distance- based geomorphometry should generally be performed in a projected CRS with linear units.

Value

A `terra::SpatRaster`.

See Also

`crop_bathy()`, `mask_bathy()`, `depth_filter()`

Examples

```
bathy <- read_bathy(blueterra_example("bathy"))
prepared <- prepare_bathy(bathy, depth_range = c(-90, -20), smooth = TRUE)
prepared
```

---

prepare_model_matrix *Prepare a model matrix from terrain data*

Description

Converts a terrain table to a numeric predictor matrix and optional response vector.

Usage

```
prepare_model_matrix(
  data,
  vars = NULL,
  response = NULL,
  scale = FALSE,
  na.rm = TRUE
)
```

Arguments

`data`	A data frame.
`vars`	Optional predictor variable names.
`response`	Optional response column name.
`scale`	Logical. If `TRUE`, center and scale predictors.
`na.rm`	Logical. Remove incomplete rows.

Value

A list with x, y, and data.

See Also

sample_terrain_cells(), terrain_pca()

Examples

```
df <- data.frame(y = c(0, 1, 0), slope = c(1, 2, 3), bpi = c(0.2, 0.1, 0.4))
prepare_model_matrix(df, response = "y")
```

---

read_bathy *Read a bathymetric or elevation raster*

Description

Reads a local raster file with terra::rast() and optionally validates that the result is usable as a bathymetric or elevation surface.

Usage

```
read_bathy(path, ..., check = TRUE)
```

Arguments

path	Local raster file path.
...	Additional arguments passed to terra::rast().
check	Logical. If TRUE, validate the raster before returning it.

Details

read_bathy() is data-source agnostic. The input can be any local raster format supported by terra, including GeoTIFF and many GDAL-readable files. The function treats the raster as a user-supplied terrain surface and preserves its values, CRS, resolution, and vertical sign convention.

Value

A terra::SpatRaster.

See Also

as_bathy(), validate_bathy(), prepare_bathy()

Examples

```
path <- blueterra_example("bathy")
bathy <- read_bathy(path)
bathy
```

`sample_terrain_cells` *Sample terrain raster cells*

Description

Draws random or regular samples from raster cells and returns a table.

Usage

```
sample_terrain_cells(
  metrics,
  size,
  method = c("random", "regular"),
  na.rm = TRUE,
  xy = TRUE,
  seed = NULL
)
```

Arguments

`metrics`	A metric raster stack.
`size`	Number of cells to sample.
`method`	Sampling method, `"random"` or `"regular"`.
`na.rm`	Logical. Omit rows with missing values.
`xy`	Logical. Include cell coordinates.
`seed`	Optional random seed used before sampling.

Value

A tibble of sampled cell values.

See Also

`extract_terrain_points()`, `prepare_model_matrix()`

Examples

```
bathy <- read_bathy(blueterra_example("bathy"))
sample_terrain_cells(bathy, size = 10)
```

`sample_transects` *Sample rasters along transects*

Description

Extracts raster values along transect lines at regular distances.

Usage

```
sample_transects(
  transects,
  x,
  spacing = NULL,
  n = NULL,
  method = "bilinear",
  drop_na = FALSE
)

extract_cross_sections(
  transects,
  x,
  spacing = NULL,
  n = NULL,
  method = "bilinear"
)
```

Arguments

`transects`	Line geometry from `make_transects()` or another source.
`x`	A `terra::SpatRaster` or local raster path. Multi-layer rasters are accepted when sampling bathymetry together with derived terrain metrics.
`spacing`	Optional sample spacing in map units.
`n`	Optional number of sample points per transect.
`method`	Extraction method passed to `terra::extract()`.
`drop_na`	Logical. If `TRUE`, remove sample rows where all raster value columns are missing.

Details

If `spacing` and `n` are both `NULL`, twenty points are sampled per transect. Distances are measured from the first line vertex and are in map units. Transect attribute columns, including orientation metadata created by `make_transects()`, are preserved in the sampled table.

Value

A tibble with transect identifiers, distances, coordinates, and raster values.

See Also

make_transects(), summarize_cross_sections()

Examples

```
bathy <- read_bathy(blueterra_example("bathy"))
zones <- terra::vect(blueterra_example("zones"))
transects <- make_transects(zones[1, ], spacing = 100, bathy = bathy)
sample_transects(transects, bathy, n = 5)
```

---

select_process_representatives

Select representative metrics for each process group

Description

Chooses a small set of catalog metrics for process-oriented summaries.

Usage

```
select_process_representatives(
  catalog = metric_catalog(),
  groups = NULL,
  metrics_available = NULL,
  representatives = NULL
)
```

Arguments

catalog	Optional catalog table. Defaults to metric_catalog().
groups	Optional process groups to retain.
metrics_available	Optional vector of available metric names.
representatives	Optional named character vector mapping process-group names to representative metric names.

Details

The default representative is the first implemented metric in the catalog for each process group. Users should review and override representatives based on their raster resolution, focal scales, and scientific question.

Value

A tibble with one representative metric per process group.

See Also

metric_catalog(), assign_process_groups()

Examples

```
select_process_representatives()
```

---

smooth_bathy *Smooth a bathymetric or elevation raster*

Description

Applies a square moving-window mean filter.

Usage

```
smooth_bathy(x, window = 3, na.rm = TRUE, filename = "", overwrite = FALSE)
```

Arguments

x	A raster-like object accepted by as_bathy().
window	Odd integer focal-window size.
na.rm	Logical. Remove missing values inside the focal window.
filename	Optional output raster path.
overwrite	Logical. Allow overwriting filename.

Details

Smoothing changes local gradients and can strongly affect slope, curvature, rugosity, TPI, and BPI. Use it deliberately and report the window size.

Value

A terra::SpatRaster.

See Also

prepare_bathy()

Examples

```
bathy <- read_bathy(blueterra_example("bathy"))
smooth_bathy(bathy, window = 3)
```

`standardize_metric_names` *Standardize or rename metric layers*

Description

Converts metric names to stable snake-case names or applies a user-supplied dictionary.

Usage

```
standardize_metric_names(x)

rename_metric_layers(x, dictionary = NULL)
```

Arguments

`x`	A `terra::SpatRaster`, character vector, or data frame.
`dictionary`	Optional named character vector. Names are old metric names; values are new metric names.

Details

Standardization is conservative and does not change raster values. Use this helper to make layer names predictable before joining to the metric catalog or exporting model-ready tables.

Value

An object of the same broad type as `x`.

See Also

`metric_catalog()`

Examples

```
standardize_metric_names(c("Slope (deg)", "Broad BPI"))
```

summarize_cross_sections *Summarize sampled cross-sections*

Description

Summarizes raster values sampled along transects.

Usage

```
summarize_cross_sections(
  samples,
  value_col = NULL,
  group_col = "transect_id",
  fun = c("mean", "sd", "min", "max", "median"),
  na.rm = TRUE,
  normalize_distance = FALSE,
  n_bins = 50
)
```

Arguments

samples	Output from sample_transects() or extract_cross_sections().
value_col	Optional value column. Defaults to the first numeric column that is not an identifier or coordinate.
group_col	Column used to group cross-section samples.
fun	Summary functions.
na.rm	Logical. Remove missing values.
normalize_distance	Logical. If TRUE, summarize values by normalized position along each transect.
n_bins	Number of normalized-distance bins when normalize_distance = TRUE.

Value

A tibble with one row per group.

See Also

sample_transects()

Examples

```
bathy <- read_bathy(blueterra_example("bathy"))
zones <- terra::vect(blueterra_example("zones"))
transects <- make_transects(zones[1, ], spacing = 100, bathy = bathy)
samples <- sample_transects(transects, bathy, n = 5)
summarize_cross_sections(samples)
```

summarize_depth_bands *Summarize terrain by depth bands*

Description

Groups raster cells into depth or elevation bands and summarizes metric values within each band.

Usage

```
summarize_depth_bands(
  bathy,
  metrics = NULL,
  breaks,
  positive_depth = NULL,
  fun = c("mean", "sd", "min", "max", "median"),
  na.rm = TRUE
)
```

Arguments

bathy	A bathymetric or elevation raster.
metrics	Optional metric raster stack. If NULL, bathy is summarized.
breaks	Numeric band breaks.
positive_depth	Optional logical. If TRUE, bands are applied to absolute depth. If FALSE or NULL, bands are applied to stored values.
fun	Summary functions.
na.rm	Logical. Remove missing values.

Details

Depth bands are sensitive to vertical sign convention. For negative-elevation bathymetry, use negative breaks or set positive_depth = TRUE with positive depth breaks.

Value

A tibble with one row per depth band and metric.

See Also

depth_filter(), summarize_terrain()

Examples

```
bathy <- read_bathy(blueterra_example("bathy"))
summarize_depth_bands(bathy, breaks = c(-90, -60, -30, 0))
```

summarize_isobath_terrain *Summarize terrain by isobath corridor*

Description

Computes summary statistics for metric rasters inside corridor polygons.

Usage

```
summarize_isobath_terrain(
  metrics,
  corridors,
  fun = c("mean", "sd", "min", "max", "median"),
  na.rm = TRUE,
  exact = FALSE,
  ...
)
```

Arguments

metrics	A metric raster stack.
corridors	Corridor polygons.
fun	Summary functions.
na.rm	Logical. Remove missing values.
exact	Logical. Use coverage-fraction-weighted exact intersections through summarize_terrain() when available.
...	Additional arguments passed to summarize_terrain().

Value

A tibble with one row per corridor.

See Also

make_isobath_corridors(), summarize_terrain()

Examples

```
bathy <- read_bathy(blueterra_example("bathy"))
terrain <- derive_terrain(bathy, metrics = c("slope", "bpi"))
corridors <- make_isobath_corridors(bathy, depths = -50, width = 5)
summarize_isobath_terrain(terrain, corridors)
```

summarize_process_groups *Summarize process group representation*

Description

Counts available metrics by process group.

Usage

```
summarize_process_groups(x, catalog = metric_catalog(), groups = NULL)
```

Arguments

x	A terra::SpatRaster, character vector, or data frame with metric columns.
catalog	Optional catalog table. Defaults to metric_catalog().
groups	Optional named character vector passed to assign_process_groups().

Details

This function summarizes which process groups are represented by a metric stack. It does not compute raster statistics; use summarize_terrain() for spatial summaries.

Value

A tibble with process group counts.

See Also

assign_process_groups(), summarize_terrain()

Examples

```
terrain <- derive_terrain(read_bathy(blueterra_example("bathy")))
summarize_process_groups(terrain)
```

summarize_terrain *Summarize terrain metrics by polygon zones*

Description

Computes summary statistics for raster metrics inside polygons, buffers, or corridor features.

Usage

```
summarize_terrain(
  metrics,
  zones,
  fun = c("mean", "sd", "min", "max", "median"),
  na.rm = TRUE,
  exact = FALSE,
  ...
)

summarize_terrain_by_zone(
  metrics,
  zones,
  fun = c("mean", "sd", "min", "max", "median"),
  na.rm = TRUE,
  exact = FALSE,
  ...
)
```

Arguments

metrics	A metric raster stack.
zones	Polygon zones as sf, terra::SpatVector, or a local vector path.
fun	Summary functions. Supported values are "mean", "sd", "min", "max", "median", "sum", and "count".
na.rm	Logical. Remove missing values before summarizing.
exact	Logical. If TRUE, use exactextractr for exact raster-polygon intersections and coverage-fraction-weighted summaries. The optional package must be installed. Weighted count is an effective cell count and weighted sum is a coverage-fraction-weighted cell-value sum.
...	Additional arguments passed to extraction functions.

Details

summarize_terrain() does not assume specific zones, depth ranges, or ecological labels. For distance-sensitive summaries, use zones and rasters in a projected CRS. With exact = TRUE, positive coverage_fraction values weight means, population standard deviations, sums, counts, and medians. Minimum and maximum are evaluated over intersected cells with positive coverage. The resulting exact mean is area-weighted when raster cells have equal area in the working CRS.

Value

A tibble with zone identifiers, zone attributes, and wide summary columns named `metric_function`.

See Also

`summarize_depth_bands()`, `extract_terrain_points()`

Examples

```
bathy <- read_bathy(blueterra_example("bathy"))
terrain <- derive_terrain(bathy, metrics = c("slope", "bpi"))
zones <- terra::vect(blueterra_example("zones"))
summarize_terrain(terrain, zones)
```

---

terrain_correlation *Correlation table for terrain variables*

Description

Computes pairwise correlations among numeric terrain variables.

Usage

```
terrain_correlation(
  data,
  vars = NULL,
  method = "pearson",
  use = "pairwise.complete.obs"
)
```

Arguments

`data`	A data frame.
`vars`	Optional character vector of numeric variables.
`method`	Correlation method passed to `stats::cor()`.
`use`	Missing-value handling passed to `stats::cor()`.

Value

A tibble with variable pairs and correlation coefficients.

See Also

`terrain_pca()`

Examples

```
bathy <- read_bathy(blueterra_example("bathy"))
terrain <- derive_terrain(bathy, metrics = c("slope", "bpi", "roughness"))
cells <- sample_terrain_cells(terrain, size = 30)
terrain_correlation(cells)
```

---

terrain_effect_size *Terrain effect sizes*

Description

Computes standardized differences between two groups for numeric terrain variables.

Usage

```
terrain_effect_size(data, group, vars = NULL, method = "cohens_d", ...)
```

Arguments

data	A data frame.
group	Character name of the grouping column.
vars	Optional character vector of numeric variables.
method	Effect-size method. Currently "cohens_d".
...	Reserved for future methods.

Details

Cohen's d is computed as the difference in group means divided by pooled standard deviation. Exactly two non-missing groups are required.

Value

A tibble with one row per variable.

See Also

terrain_pca()

Examples

```
df <- data.frame(group = rep(c("a", "b"), each = 5), slope = 1:10)
terrain_effect_size(df, group = "group", vars = "slope")
```

`terrain_pca` *Principal components analysis for terrain tables*

Description

Runs PCA on numeric terrain variables and returns tidy scores, loadings, variance explained, and the fitted model.

Usage

```
terrain_pca(
  data,
  vars = NULL,
  center = TRUE,
  scale. = TRUE,
  metadata_cols = NULL,
  keep_metadata = TRUE,
  ...
)
```

Arguments

`data`	A data frame.
`vars`	Optional character vector of numeric variables.
`center`	Logical passed to `stats::prcomp()`.
`scale.`	Logical passed to `stats::prcomp()`.
`metadata_cols`	Optional non-PCA columns appended to the score table.
`keep_metadata`	Logical. Preserve non-PCA columns in `scores`.
`...`	Additional arguments passed to `stats::prcomp()`.

Details

Rows with incomplete values in selected variables are omitted before PCA. PCA is descriptive and should be interpreted with scale, CRS, and sampling design in mind.

Value

A list with `scores`, `loadings`, `variance`, `model`, `vars`, and `complete_rows`.

See Also

`prepare_model_matrix()`, `terrain_correlation()`

Examples

```
bathy <- read_bathy(blueterra_example("bathy"))
terrain <- derive_terrain(bathy, metrics = c("slope", "bpi", "roughness"))
cells <- sample_terrain_cells(terrain, size = 30)
terrain_pca(cells)
```

---

terrain_pca_by_group *Run PCA overall and within groups*

Description

Fits one terrain PCA across all rows and one PCA within each group level.

Usage

```
terrain_pca_by_group(
  data,
  group,
  vars = NULL,
  center = TRUE,
  scale. = TRUE,
  min_rows = 5,
  ...
)
```

Arguments

`data`	A data frame.
`group`	Character name of the grouping column.
`vars`	Optional numeric variables used in PCA.
`center, scale.`	Logical values passed to `terrain_pca()`.
`min_rows`	Minimum rows required for a group-specific PCA.
`...`	Additional arguments passed to `terrain_pca()`.

Details

Group-specific PCA is useful for checking whether ordination structure is dominated by one site or sampling frame. Groups with fewer than `min_rows` rows are omitted with a warning.

Value

A named list with `overall` and `groups`. `groups` is a named list of PCA objects.

See Also

`terrain_pca()`, `plot_process_pca()`

Examples

```
df <- data.frame(site = rep(c("a", "b"), each = 8), slope = rnorm(16),
                 tri = rnorm(16), bpi = rnorm(16))
terrain_pca_by_group(df, group = "site")
```

`validate_bathy` *Validate a bathymetric or elevation raster*

Description

Checks that an object is a readable `terra::SpatRaster` with dimensions, layers, and values suitable for terrain analysis.

Usage

```
validate_bathy(
  x,
  require_crs = FALSE,
  require_values = TRUE,
  allow_multi = FALSE
)
```

Arguments

`x`	A raster-like object accepted by `as_bathy()`.
`require_crs`	Logical. If `TRUE`, require a declared CRS.
`require_values`	Logical. If `TRUE`, require raster values.
`allow_multi`	Logical. If `FALSE`, warn when more than one layer is supplied.

Details

Validation does not decide whether values represent positive depth or negative elevation. Use `check_bathy_units()`, `set_depth_positive()`, or `set_depth_negative()` when sign convention matters.

Value

Invisibly returns the input raster.

See Also

`check_bathy_crs()`, `check_bathy_units()`

Examples

```
bathy <- read_bathy(blueterra_example("bathy"))
validate_bathy(bathy)
```

Index