Package: potentiomap (via r-universe)

July 19, 2026

Title Build Potentiometric Surfaces and Hydraulic-Gradient Arrows

Version 0.2.0

Date 2026-07-17

Description Prepares groundwater-level observations from measured hydraulic head or from depth-to-water and land-surface elevation, interpolates potentiometric surfaces using thin-plate splines, inverse-distance weighting, ordinary Kriging, universal Kriging, or user-supplied methods, and creates raster, contour, diagnostic, support, and hydraulic-gradient products for review and export. Functions retain method conditions and fit diagnostics, validate explicit prediction tasks, inspect model-conditional uncertainty and monitoring-network sensitivity, identify limited prediction support, and check whether scaled hydraulic-gradient arrows remain within finite raster support and end at lower modeled head. Raster processing uses methods from 'terra' (Hijmans 2025) <doi:10.32614/CRAN.package.terra>, thin-plate splines use methods from 'fields' (Nychka et al. 2021) <doi:10.5065/D6W957CT>, and geostatistical interpolation uses methods from 'gstat' (Pebesma 2004) <doi:10.1016/j.cageo.2004.03.012>.

License GPL-3

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

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

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.1)

Imports fields, grDevices, graphics, gstat, sf, stats, terra, xml2

Suggests knitr, rmarkdown, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/testthat/edition 3

Config/Needs/website pkgdown

Config/Needs/coverage covr

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

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

Date/Publication 2026-07-18 00:59:34 UTC

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

RemoteRef HEAD

RemoteSha 42851bfd7be99dc720da871339c6fe1b52242e48

Contents

`plot.potentiomap_contour_support` *Plot classified contour support*

Description

Draws supported sections as solid, approximate sections as dashed, and optionally unsupported sections as dotted. Colors and line widths are not assigned by the result and can be supplied through .... The title and legend identify the classes as user-defined support criteria, not statistical confidence.

Usage

```
## S3 method for class 'potentiomap_contour_support'
plot(
  x,
  show_unsupported = TRUE,
  label_levels = FALSE,
  legend_position = "topright",
  main = "Contour support (user-defined criteria)",
  ...
)
```

Arguments

x	A potentiomap_contour_support result.
show_unsupported	Draw unsupported sections.
label_levels	Label contour values where possible.
legend_position	Base-graphics legend position, or NULL to omit it.
main	Plot title.
...	Additional arguments passed to terra::plot().

Value

Invisibly returns x.

---

potentiomap_conditions

potentiomap condition classes

Description

Important warnings and errors raised by potentiomap have stable S3 classes so calling code can respond without matching the complete message text. All package warnings inherit from `potentiomap_warning`; all package errors inherit from `potentiomap_error`.

Details

Specific classes include `potentiomap_input_error`, `potentiomap_metadata_error`, `potentiomap_crs_error`, `potentiomap_arrow_endpoint_warning`, `potentiomap_uk_instability_warning`, `potentiomap_kriging_convergenc` `potentiomap_tps_gcv_boundary_warning`, `potentiomap_contour_level_warning`, `potentiomap_contour_support_w` `potentiomap_contour_support_error`, `potentiomap_contour_threshold_error`, `potentiomap_contour_uncertain` `potentiomap_support_warning`, and `potentiomap_export_error`.

potentiomap_result *Structured potentiometric-surface result*

Description

A `potentiomap_result` is the opt-in return from `ps_interpolate(..., return = "result")`. It retains the ordinary named surface list together with method diagnostics and processing context.

Details

Fields are:

- `surfaces`: named `SpatRaster` list.
- `diagnostics`: method-attributed fit and prediction diagnostics.
- `method_parameters`: interpolation controls used for each method.
- `input_summary`: observation counts, metadata, and dropped records.
- `observation_count`: retained observation count.
- `dropped_records`: summary of excluded input records.
- `grid_geometry`: output dimensions, resolution, extent, and cell count.
- `crs`: output coordinate reference system.
- `mask_summary`: whether and how a mask was supplied.
- `support`: optional result from `ps_prediction_support()`.
- `conditions`: captured warnings and messages by method.
- `package_version`: potentiomap version.
- `call`: original interpolation call.

ps_anisotropy *Explore directional anisotropy in hydraulic head*

Description

Calculates directional variograms and exploratory fitted ranges. The major continuity direction is periodic over 180 degrees and uses gstat's clockwise-from-North convention. Weak evidence is warned and anisotropy is never activated in interpolation unless supplied explicitly.

Usage

```
ps_anisotropy(
  points,
  formula = Z ~ 1,
  directions = seq(0, 135, by = 45),
  tolerance = 22.5,
  cutoff = NULL,
  width = NULL,
  model = "Sph",
  minimum_pairs = 20,
  validation = FALSE
)
```

Arguments

`points`	Groundwater-head points.
`formula`	Variogram trend formula.
`directions`	Direction angles.
`tolerance`	Direction tolerance in degrees.
`cutoff`, `width`	Variogram controls.
`model`	Candidate directional model.
`minimum_pairs`	Minimum total pairs required for a directional fit.
`validation`	Optionally request a documented isotropic/anisotropic validation comparison (recorded as not run when no validation design is supplied).

Value

A `potentiomap_anisotropy` with empirical values, directional fits, major direction, minor-to-major range ratio, and conditions.

Examples

```
data("synthetic_wells")
pts <- ps_make_points(synthetic_wells, "x", "y", "gw_elevation",
                      "well_id", "EPSG:26916")
a <- ps_anisotropy(pts, minimum_pairs = 5)
a$summary
# This exploratory result is not applied automatically.
```

ps_arrow_vertices *Extract arrow base or tip points*

Description

Extract arrow base or tip points

Usage

```
ps_arrow_vertices(
  arrows,
  which = c("last", "first"),
  out_file = NULL,
  overwrite = TRUE
)
```

Arguments

arrows	A line SpatVector or path to a line vector file.
which	"first" for bases or "last" for tips.
out_file	Optional vector output path.
overwrite	Overwrite out_file when it exists.

Value

A point terra::SpatVector, including an empty point vector for zero arrows.

Examples

```
line <- terra::vect(list(rbind(c(0, 0), c(1, 1))), type = "lines",
                    crs = "EPSG:3857")
ps_arrow_vertices(line, "last")
```

ps_candidate_network *Rank explicit candidate monitoring locations with recorded constraints*

Description

Rank explicit candidate monitoring locations with recorded constraints

Usage

```
ps_candidate_network(
  existing_points,
  candidates,
  objective = c("spatial_coverage", "support_gap", "kriging_variance_reduction",
    "user_score"),
  n_select = 1,
  target = NULL,
  variogram_model = NULL,
  trend = NULL,
  minimum_existing_distance = 0,
  minimum_candidate_distance = 0,
  allowed_area = NULL,
  exclusion_area = NULL,
  cost = NULL,
  user_score = NULL,
  sequential = TRUE
)
```

Arguments

`existing_points`	Existing monitoring points.
`candidates`	Explicit candidate point locations.
`objective`	Candidate-scoring objective.
`n_select`	Number selected by sequential greedy ranking.
`target`	Explicit target points or raster for target-weighted objectives.
`variogram_model`, `trend`	Model for kriging-variance reduction.
`minimum_existing_distance`, `minimum_candidate_distance`	Spacing rules.
`allowed_area`, `exclusion_area`	Spatial constraints.
`cost`	Optional finite positive candidate costs.
`user_score`	Optional supplied scores.
`sequential`	Update scores after each choice.

Value

A `potentiomap_candidate_network` object. The greedy sequence is not claimed to be globally optimal or to identify drillable sites.

Examples

```
data("synthetic_wells", "synthetic_candidate_sites")
p <- ps_make_points(synthetic_wells, "x", "y", "gw_elevation",
```

```
                    "well_id", "EPSG:26916")
c <- terra::vect(subset(synthetic_candidate_sites, !excluded),
                 geom = c("x", "y"), crs = "EPSG:26916")
design <- ps_candidate_network(p, c, n_select = 2,
                               objective = "spatial_coverage")
design$selected_sequence
# Sequential greedy selection is not a globally optimal drilling plan.
```

`ps_check_observations` *Check groundwater observation records*

Description

Reports deterministic input conflicts and statistical review flags before interpolation. Unusual values are never automatically treated as errors or deleted. Coordinate distance checks require a known projected CRS; longitude and latitude are flagged for planar analysis. Screen and head comparisons assume all elevations use one documented vertical datum.

Usage

```
ps_check_observations(
  data,
  x = NULL,
  y = NULL,
  value = NULL,
  id = NULL,
  datetime = NULL,
  unit = NULL,
  vertical_datum = NULL,
  depth = NULL,
  surface_elevation = NULL,
  screen_top = NULL,
  screen_bottom = NULL,
  unit_group = NULL,
  duplicate_tolerance = 0,
  action = c("report", "return_clean")
)
```

Arguments

`data`	A data frame, `sf` object, or point `SpatVector`.
`x`, `y`, `value`, `id`, `datetime`, `unit`, `vertical_datum`, `depth`, `surface_elevation`	Optional column names for location, measurement, timing, unit, datum, and land-surface fields.
`screen_top`, `screen_bottom`, `unit_group`	Optional screen-elevation and hydrogeologic-unit column names.

duplicate_tolerance
: Nonnegative coordinate distance within which different records are flagged as colocated. Use projected map units.

action
: Return only the report or also deterministically remove records with missing IDs, coordinates, or heads.

Value

A potentiomap_observation_check with issues, original data, retained, removed, counts, settings, metadata, and conditions.

Examples

```
d <- data.frame(id = c("A", "A"), x = c(0, 0), y = c(0, 0),
                head = c(10, 11))
attr(d, "crs") <- "EPSG:26920"
check <- ps_check_observations(d, "x", "y", "head", "id")
check$issues
# Flags require hydrogeologic review; they do not prove a record is wrong.
```

---

ps_compare_methods *Compare validated interpolation methods*

Description

Ranks adequately covered methods within an explicit validation design and support subset. It does not declare a universally best method and does not aggregate multiple objectives unless the user supplies weights.

Usage

```
ps_compare_methods(
  validation,
  metric = "rmse",
  design = NULL,
  support_subset = c("all", "finite", "supported"),
  minimum_coverage = 0.9,
  tie_tolerance = NULL,
  objective_weights = NULL,
  select = FALSE
)
```

Arguments

validation
: A potentiomap_validation or documented compatible metric table.

metric
: One or more metric columns.

design
: One validation design; required when multiple designs exist.

`support_subset` All scheduled, finite, or supported predictions.

`minimum_coverage`
Minimum finite coverage.

`tie_tolerance` Nonnegative absolute metric tolerance; default is a scale-aware square-root machine tolerance.

`objective_weights`
Explicit named nonnegative weights for multiple criteria.

`select` Select one method only after all selection gates pass.

Value

A `potentiomap_method_comparison` with ranking, Pareto status and optional selection.

Examples

```
data("synthetic_wells")
pts <- ps_make_points(synthetic_wells, "x", "y", "gw_elevation", "well_id", "EPSG:26916")
val <- ps_validate(pts, "IDW", "kfold", folds = 3, prediction_mode = "direct")
ps_compare_methods(val)$ranking
# Ranking is specific to this design and objective.
```

---

ps_compare_surfaces *Compare two potentiometric surfaces*

Description

Compares compatible surfaces on their common finite support. Alignment is an explicit operation and defaults to an error. The signed default is surface B minus surface A; percentage head change is not calculated.

Usage

```
ps_compare_surfaces(
  surface_a,
  surface_b,
  direction = c("b_minus_a", "a_minus_b"),
  align = c("error", "to_a", "to_b", "template"),
  template = NULL,
  resampling = c("bilinear", "near"),
  contour_levels = NULL,
  compare_gradient = TRUE,
  min_gradient = 1e-05
)
```

Arguments

surface_a, surface_b	One-layer continuous head rasters.
direction	Signed-difference order.
align	Alignment action; mismatch errors by default.
template	Explicit target for align = "template".
resampling	Continuous bilinear or explicit nearest-neighbor alignment.
contour_levels	Optional head contour levels.
compare_gradient	Compare modeled gradient magnitude and direction.
min_gradient	Threshold below which direction is undefined.

Value

A potentiomap_surface_comparison with difference/support/gradient rasters, contour displacement, summaries, and an alignment manifest.

Examples

```
a <- terra::rast(nrows = 3, ncols = 3, xmin = 0, xmax = 3, ymin = 0, ymax = 3,
                 crs = "EPSG:26920", vals = 1:9)
cmp <- ps_compare_surfaces(a, a + 1)
cmp$summary
# A modeled difference is not a storage or water-budget estimate.
```

---

ps_contour_support *Classify modeled contour sections by local prediction support*

Description

Divides modeled contour lines according to user-defined spatial-support criteria. Sections close to groundwater observations may be classified as supported, while sections crossing larger monitoring gaps may be classified as approximate or unsupported. Classification occurs at the resolution of the supplied prediction-support raster and does not move, smooth, close, or convert the contour lines.

Usage

```
ps_contour_support(
  contours,
  points = NULL,
  surface = NULL,
  support = NULL,
  uncertainty = NULL,
  supported_distance = NULL,
  approximate_distance = NULL,
```

```
  distance_reference = c("map_units", "median_nearest_neighbor"),
  require_inside_hull = TRUE,
  neighbor_radius = NULL,
  minimum_neighbors = NULL,
  supported_uncertainty = NULL,
  approximate_uncertainty = NULL,
  combine = c("worst", "distance", "uncertainty"),
  keep_unsupported = TRUE,
  minimum_segment_length = 0,
  return = c("result", "segments"),
  uncertainty_type = NULL,
  uncertainty_units = NULL
)
```

Arguments

`contours`	Nonempty line SpatVector or line-vector input readable by terra::vect(). A recognizable contour-level field is required.
`points`	Optional groundwater observation points. Required when support is not supplied and for relative-spacing or neighbor criteria unless the support result retains its training points.
`surface`	Optional one-layer potentiometric-surface SpatRaster. Required with points when support is not supplied. When both a surface and support result are supplied, their geometry must match.
`support`	Optional `ps_prediction_support()` result. Its existing distance, training-hull, mask, and finite-prediction layers are reused.
`uncertainty`	Optional one-layer raster containing a user-identified uncertainty measure. It is never resampled and must match support geometry.
`supported_distance`, `approximate_distance`	Optional paired nonnegative distance thresholds. The approximate threshold must be at least the supported threshold. No default distances are assumed.
`distance_reference`	Either "map_units" or "median_nearest_neighbor". In the latter case, supplied thresholds are multipliers of the calculated median nearest-neighbor spacing.
`require_inside_hull`	When TRUE, cells outside the training convex hull cannot be supported but may remain approximate when other criteria permit. A convex hull is not an aquifer boundary.
`neighbor_radius`, `minimum_neighbors`	Optional paired local-density criterion in projected map units and unique observation locations.
`supported_uncertainty`, `approximate_uncertainty`	Optional paired thresholds in the units or scale of uncertainty.
`combine`	"worst" combines all active primary criteria, "distance" uses distance, and "uncertainty" uses uncertainty. Finite prediction, mask, requested hull, and neighbor rules remain applicable.

`keep_unsupported`
: Retain unsupported line sections when TRUE.

`minimum_segment_length`
: Nonnegative minimum retained line length in projected map units.

`return`
: "result" for a potentiomap_contour_support object or "segments" for only the classified line SpatVector.

`uncertainty_type`
: Required description when uncertainty is supplied, such as "kriging prediction variance" or "user support index".

`uncertainty_units`
: Optional units or scale description for the supplied uncertainty raster.

Details

Distance thresholds can be expressed in projected map units or as multiples of the median nearest-neighbor spacing among unique observation locations. Optional training-hull, local-neighbor, and user-identified uncertainty criteria can modify the result. With combine = "worst", the least supported active criterion determines the cell class. Finite prediction and mask status are always enforced.

These classes describe local observation support for the mapped contour. They are not statistical confidence intervals. Proximity to wells does not prove that a contour is correct, and distance from wells does not prove that it is wrong. A solid contour remains an interpolation between observations; an approximate contour is still generated from the modeled surface. Appropriate criteria depend on network geometry, hydrogeology, interpolation method, raster resolution, and intended map use.

Value

A potentiomap_contour_support list with segments, a line-length summary, thresholds, the support information used, settings, and captured conditions; or only segments. Segment attributes include the original contour level and source ID, support class and reason, distance, hull and finite-support statistics, optional neighbor and uncertainty statistics, length, threshold reference, and a line_type style hint.

Examples

```
data("synthetic_wells")
wells <- ps_make_points(synthetic_wells, "x", "y", "gw_elevation",
                        "well_id", "EPSG:26916")
surface <- ps_interpolate(wells, methods = "IDW", grid_res = 300)$IDW
support <- ps_prediction_support(wells, surface = surface)
contours <- ps_contours(surface, interval = 1)
classified <- suppressWarnings(ps_contour_support(
  contours, support = support,
  supported_distance = 500, approximate_distance = 1200,
  require_inside_hull = TRUE
))
classified$summary
```

ps_contour_uncertainty *Construct pointwise contour-uncertainty bands*

Description

Construct pointwise contour-uncertainty bands

Usage

```
ps_contour_uncertainty(
  uncertainty,
  levels,
  probability = 0.9,
  method = c("empirical_crossing", "gaussian_pointwise"),
  keep_realized_contours = FALSE,
  minimum_realizations = 20,
  accept_gaussian = FALSE
)
```

Arguments

uncertainty	A potentiomap_uncertainty object.
levels	Contour head levels.
probability	Central pointwise probability.
method	Empirical realization crossing or Gaussian pointwise method.
keep_realized_contours	Retain contours for each realization.
minimum_realizations	Minimum successful empirical realizations.
accept_gaussian	Explicitly accept the Gaussian pointwise assumption.

Value

A potentiomap_contour_uncertainty object. Bands are pointwise, not simultaneous confidence regions.

Examples

```
data("synthetic_wells")
p <- ps_make_points(synthetic_wells[1:14, ], "x", "y", "gw_elevation",
                    "well_id", "EPSG:26916")
fit <- suppressWarnings(ps_interpolate(p, methods = "OK", grid_res = 300,
                                       return = "result"))
u <- ps_surface_uncertainty(fit, approach = "kriging_variance")
```

```
cu <- ps_contour_uncertainty(u, levels = 168, method = "gaussian_pointwise",
                             accept_gaussian = TRUE)
cu$level_manifest
# This is a pointwise band, not a simultaneous confidence region.
```

---

ps_contours *Create contours and a contour-level inventory*

Description

Extracts contour lines from a one-layer surface. The default remains a line `SpatVector`. The opt-in result inventories every requested level, its relation to the finite surface range, returned feature count, and any omission. Open lines are not converted to polygons.

Usage

```
ps_contours(
  surface,
  interval = 1,
  levels = NULL,
  return = c("contours", "result")
)
```

Arguments

`surface`	One-layer potentiometric-surface `SpatRaster`.
`interval`	Positive contour interval in surface units.
`levels`	Optional finite explicit contour levels. `interval` is ignored when these are supplied.
`return`	Either `"contours"` for the backward-compatible `SpatVector` or `"result"` for a structured inventory.

Value

A line `SpatVector`, or a `potentiomap_contour_result` containing `contours`, `manifest`, `surface_range`, `call`, `interval`, `levels`, `warnings`, and `package_version`.

Examples

```
data("synthetic_wells")
pts <- ps_make_points(synthetic_wells, "x", "y", "gw_elevation",
                      "well_id", "EPSG:26916")
surface <- ps_interpolate(pts, methods = "IDW", grid_res = 150)$IDW
result <- ps_contours(surface, levels = c(165, 168, 171), return = "result")
result$manifest
```

`ps_cross_section` *Build a plot-ready potentiometric cross-section*

Description

Build a plot-ready potentiometric cross-section

Usage

```
ps_cross_section(
  transect,
  head_surface,
  land_surface = NULL,
  wells = NULL,
  screen_top = NULL,
  screen_bottom = NULL,
  well_id = NULL,
  maximum_well_offset = NULL,
  support = NULL,
  uncertainty = NULL,
  step = NULL,
  vertical_exaggeration = 1
)
```

Arguments

`transect`	One line feature.
`head_surface`	Head surface raster.
`land_surface`	Optional land-surface raster.
`wells`	Optional monitoring wells.
`screen_top`, `screen_bottom`, `well_id`	Optional absolute-elevation columns.
`maximum_well_offset`	Maximum perpendicular offset retained.
`support`, `uncertainty`	Optional rasters sampled along the transect.
`step`	Explicit profile spacing.
`vertical_exaggeration`	Plot setting recorded without changing data.

Value

A `potentiomap_cross_section` object. It does not invent hydrostratigraphy or represent a three-dimensional numerical flow model.

Examples

```
data("synthetic_wells", "synthetic_transect")
p <- ps_make_points(synthetic_wells, "x", "y", "gw_elevation",
                    "well_id", "EPSG:26916")
s <- ps_interpolate(p, methods = "IDW", grid_res = 250)$IDW
line <- terra::vect(synthetic_transect, geom = "wkt", crs = "EPSG:26916")
section <- ps_cross_section(line, s, step = 250, vertical_exaggeration = 5)
section$summary
# The section is not a three-dimensional groundwater-flow model.
```

`ps_depth_to_water_surface` *Calculate depth to a water-table or potentiometric surface*

Description

Calculates land-surface elevation minus modeled head after explicit geometry, unit, and vertical-datum checks. Negative values are preserved. For a confined surface the product is depth to the potentiometric surface, not necessarily depth to the water table.

Usage

```
ps_depth_to_water_surface(
  head_surface,
  land_surface,
  surface_type = c("water_table", "potentiometric"),
  align = c("error", "to_head", "to_land", "template"),
  template = NULL,
  resampling = "bilinear",
  tolerance = 0
)
```

Arguments

`head_surface`, `land_surface`	One-layer continuous elevation rasters.
`surface_type`	Water table or potentiometric surface terminology.
`align`, `template`, `resampling`	Explicit alignment controls.
`tolerance`	Nonnegative absolute depth classified as near zero.

Value

A `potentiomap_depth_surface` with depth and review/support masks.

Examples

```
h <- terra::rast(nrows = 2, ncols = 2, xmin = 0, xmax = 2, ymin = 0, ymax = 2,
                 crs = "EPSG:26920", vals = c(9, 11, 8, 10))
land <- h * 0 + 10
z <- ps_depth_to_water_surface(h, land, "potentiometric")
terra::values(z$depth)
# Negative values are retained for hydrogeologic and datum review.
```

`ps_diagnostics` *Extract interpolation diagnostics*

Description

Extract interpolation diagnostics

Usage

```
ps_diagnostics(x, method = NULL)
```

Arguments

`x`	A `potentiomap_result`.
`method`	Optional method name. When omitted, all method diagnostics are returned.

Value

A named diagnostic list, or one method-specific list.

Examples

```
data("synthetic_wells")
pts <- ps_make_points(synthetic_wells, "x", "y", "gw_elevation",
                      "well_id", "EPSG:26916")
result <- ps_interpolate(pts, methods = "IDW", grid_res = 150,
                         return = "result")
ps_diagnostics(result, "IDW")
```

`ps_export_contour_support` *Export classified contour-support products*

Description

Writes the classified line layer and optional CSV summaries. GeoPackage is recommended because it preserves long field names more reliably than a shapefile. The line_type field is a portable style suggestion; the support class and reasons remain explicit attributes.

Usage

```
ps_export_contour_support(
  x,
  out_dir,
  out_stub = "gw",
  vector_format = c("gpkg", "shapefile"),
  write_summary = TRUE,
  write_thresholds = TRUE,
  overwrite = TRUE
)
```

Arguments

`x`	A potentiomap_contour_support result.
`out_dir`	Output directory.
`out_stub`	Safe output filename prefix.
`vector_format`	"gpkg" or "shapefile".
`write_summary, write_thresholds`	Write CSV sidecars.
`overwrite`	Overwrite existing files.

Value

A data frame listing written products.

Examples

```
# See ps_contour_support() for classification. GeoPackage is recommended.
```

ps_export_style *Export open GIS style XML*

Description

Export open GIS style XML

Usage

```
ps_export_style(
  x,
  file,
  format = c("qml", "sld"),
 layer_type = c("head_raster", "depth_raster", "contours", "contour_support", "arrows",
   "wells", "support"),
  field = NULL,
  units = NULL,
  palette = NULL,
  breaks = NULL,
  overwrite = FALSE
)
```

Arguments

x	Object whose values inform default raster breaks.
file	Output QML or SLD file.
format	QGIS QML or standards-based SLD.
layer_type	Styled layer type.
field	Attribute used for labels or support categories.
units	Optional label units.
palette	Colors or color function.
breaks	Optional explicit continuous breaks.
overwrite	Permit replacement of an existing file.

Value

A potentiomap_style_export manifest. Optional properties can render differently among GIS versions.

Examples

```
r <- terra::rast(nrows = 2, ncols = 2, xmin = 0, xmax = 2,
                 ymin = 0, ymax = 2, crs = "EPSG:26920", vals = 1:4)
file <- tempfile(fileext = ".qml")
style <- ps_export_style(r, file, "qml", "head_raster", units = "m")
style$manifest
# GIS versions can render optional style properties differently.
```

`ps_export_surfaces` *Export potentiometric-surface products*

Description

Writes deterministic GeoTIFF, vector, contour-manifest, quicklook, support, and diagnostic products only when an output directory is supplied. GeoPackage is recommended because it preserves field names and supports multiple layers better than shapefiles; the shapefile default is retained for compatibility.

Usage

```
ps_export_surfaces(
  surfaces,
  out_dir,
  out_stub = "gw",
  contour_interval = 1,
  points = NULL,
  write_raster = TRUE,
  write_contours = TRUE,
  write_png = TRUE,
  contour_levels = NULL,
  vector_format = c("shapefile", "gpkg"),
  support = NULL,
  diagnostics = NULL,
  write_contour_manifest = TRUE,
  write_support = FALSE,
  write_diagnostics = FALSE,
  write_manifest = TRUE,
  overwrite = TRUE
)
```

Arguments

`surfaces`	Named raster list or `potentiomap_result`.
`out_dir`	Output directory.
`out_stub`	Safe file prefix.
`contour_interval`	Positive contour interval.
`points`	Optional observation points for quicklooks.
`write_raster`, `write_contours`, `write_png`	Choose outputs.
`contour_levels`	Optional explicit levels.
`vector_format`	Either `"shapefile"` or `"gpkg"`.

support	Optional potentiomap_support; defaults to support stored in a structured interpolation result.
diagnostics	Optional diagnostic list; defaults to structured-result diagnostics.
write_contour_manifest, write_support, write_diagnostics, write_manifest	Choose sidecar products.
overwrite	Overwrite existing outputs.

Value

A data frame describing written files.

Examples

```
data("synthetic_wells")
pts <- ps_make_points(synthetic_wells, "x", "y", "gw_elevation",
                      "well_id", "EPSG:26916")
surfaces <- ps_interpolate(pts, methods = "IDW", grid_res = 200)
ps_export_surfaces(surfaces, tempdir(), points = pts)
```

---

ps_flow_arrows *Generate hydraulic-gradient arrows*

Description

Derives the local negative modeled-head gradient from a potentiometric surface. Arrow direction comes from raster aspect; arrow length is a display convention based on gradient, raster resolution, and scale. Arrows are map symbols, not groundwater velocities, travel times, particle paths, or traced groundwater paths.

Usage

```
ps_flow_arrows(
  surface,
  res_factor = 7,
  scale = 50,
  min_gradient = 1e-05,
  log_gradient = FALSE,
  log_arrow = FALSE,
  out_dir = NULL,
  out_stub = "gw",
  endpoint_action = c("flag", "shorten", "drop", "none"),
  endpoint_tolerance = 1e-06,
  endpoint_extraction = c("bilinear", "simple"),
  max_shortening = 12L,
  overwrite = TRUE
)
```

Arguments

surface	A one-layer groundwater-elevation SpatRaster.
res_factor	Positive integer used to thin arrows on a coarser grid.
scale	Positive cartographic length multiplier.
min_gradient	Nonnegative gradient threshold.
log_gradient	Store log1p() gradient in the returned raster.
log_arrow	Use log1p() gradient for arrow display length.
out_dir	Optional output directory. No files are written when NULL.
out_stub	Safe file prefix.
endpoint_action	One of "flag", "shorten", "drop", or "none". "flag" preserves geometry and warns; "shorten" retains direction while reducing failed lines; "drop" removes failures; "none" reproduces the version 0.1.0 unvalidated geometry.
endpoint_tolerance	Nonnegative head tolerance.
endpoint_extraction	Either "bilinear" or "simple" raster extraction.
max_shortening	Positive maximum number of length halvings.
overwrite	Overwrite gradient files when out_dir is supplied.

Details

Endpoint checking compares each straight line with the supplied raster. It can flag, shorten, or drop lines whose tip is nonfinite or higher than the base. Shortening retains the original direction and repeatedly halves length; arrows are never reversed or bent. A passing check does not establish that the interpolated surface is physically correct.

Value

A list containing at least raster, points, and arrows, plus tips, bases, validation, and validation_summary.

Examples

```
data("synthetic_wells")
pts <- ps_make_points(synthetic_wells, "x", "y", "gw_elevation",
                      "well_id", "EPSG:26916")
surface <- ps_interpolate(pts, methods = "IDW", grid_res = 150)$IDW
flow <- ps_flow_arrows(surface, res_factor = 6, scale = 30,
                       endpoint_action = "shorten")
flow$validation_summary
```

ps_head_change *Compare paired measurements and modeled head change between events*

Description

Compare paired measurements and modeled head change between events

Usage

```
ps_head_change(
  event_a,
  event_b,
  pair_by,
  event_a_time = NULL,
  event_b_time = NULL,
  method = "TPS",
  template = NULL,
  mask = NULL,
  grid_res = NULL,
  interpolation_control = list(),
  compare_gradient = TRUE
)
```

Arguments

`event_a`, `event_b`
: Point observations for two identified events.

`pair_by`
: Well identifier column.

`event_a_time`, `event_b_time`
: Optional explicit event times.

`method`
: Interpolation method.

`template`, `mask`, `grid_res`
: Fixed surface controls.

`interpolation_control`
: Named interpolation arguments.

`compare_gradient`
: Compare down-gradient directions.

Value

A `potentiomap_head_change` object. Modeled head change is not a storage, depletion, or volumetric-change estimate.

Examples

```
data("synthetic_events")
a <- subset(synthetic_events, event == "spring")
b <- subset(synthetic_events, event == "autumn")
pa <- ps_make_points(a, "x", "y", "head", "well_id", "EPSG:26916")
pb <- ps_make_points(b, "x", "y", "head", "well_id", "EPSG:26916")
change <- ps_head_change(pa, pb, "well_id", method = "IDW", grid_res = 300)
change$summary
# Modeled head change is not a storage-change estimate.
```

`ps_interpolate` *Interpolate potentiometric surfaces*

Description

Creates one raster for each requested method. Thin-plate splines (`"TPS"`) remain the software default for backward compatibility, but method selection should reflect the hydrogeologic setting, monitoring-network geometry, spatial trend, sample density, prediction support, validation design, and intended map use. Other built-in methods are inverse-distance weighting (`"IDW"`), ordinary kriging (`"OK"`), and universal kriging (`"UK"`) with a quadratic drift. Named custom functions are also supported.

Usage

```
ps_interpolate(
  points,
  value = "Z",
  methods = "TPS",
  grid_res = NULL,
  template = NULL,
  mask = NULL,
  padding = NULL,
  idw_power = 2,
  idw_nmax = 15,
  tps_lambda = NULL,
  kr_auto_cutoff = TRUE,
  kr_cutoff = NA_real_,
  kr_width = NA_real_,
  custom_methods = NULL,
  x = "x",
  y = "y",
  name_col = NULL,
  crs = NULL,
  return = c("surfaces", "result"),
  duplicate_action = c("error", "mean", "median", "first"),
  allow_geographic = FALSE,
  uk_coordinate_scaling = c("center_scale", "none"),
```

```
  diagnostic_control = NULL,
  support = FALSE,
  support_max_distance = NULL,
  trend = NULL,
  covariates = NULL,
  covariate_alignment = c("error", "bilinear", "near"),
  standardize_covariates = TRUE,
  variogram_model = NULL,
  anisotropy = NULL,
  kriging_control = list()
)
```

Arguments

points	A point SpatVector, sf object, or coordinate table.
value	Data column name when points is not standardized. Defaults to "Z".
methods	Character vector containing "TPS", "IDW", "OK", "UK", or names in custom_methods.
grid_res	Positive output cell size in projected map units.
template	Optional one-layer template SpatRaster; overrides extent construction from grid_res, padding, and mask.
mask	Optional polygon mask. A convex hull or supplied mask describes the computational domain, not an aquifer boundary.
padding	Nonnegative padding around the template extent.
idw_power, idw_nmax	Positive IDW power and optional positive maximum neighbor count.
tps_lambda	Optional nonnegative TPS smoothing parameter. NULL uses the selection performed by fields::Tps().
kr_auto_cutoff	Use an automatically derived variogram cutoff and lag width.
kr_cutoff, kr_width	Positive manual variogram values when kr_auto_cutoff = FALSE.
custom_methods	Optional named list of functions called as fun(points, template, grid). Each must return a matching SpatRaster or one numeric value per template cell.
x, y, name_col, crs	Used for a coordinate table.
return	Either "surfaces" (the backward-compatible named raster list) or "result" for a potentiomap_result.
duplicate_action	Handling for duplicate coordinates: "error", "mean", "median", or "first".
allow_geographic	Allow distance calculations in longitude/latitude degrees with a classed warning. The default is FALSE.
uk_coordinate_scaling	Either "center_scale" (the safer default) or "none" for a documented legacy comparison.

`diagnostic_control`
: Optional named list overriding heuristic UK warning thresholds. Supported names are `condition_number_warning`, `predicted_range_ratio_warning`, `overshoot_range_ratio_warning`, `warn_on_rank_deficiency`, and `warn_on_nonfinite_predictio`

`support`
: Calculate `ps_prediction_support()` for the first returned surface.

`support_max_distance`
: Optional support distance threshold.

`trend`
: Optional universal-kriging trend formula. `NULL` retains the coordinate-trend default for `"UK"`.

`covariates`
: Optional named raster covariates for external drift.

`covariate_alignment`
: Policy for a covariate that is not aligned to the output template: error, bilinear resampling, or nearest-neighbor resampling.

`standardize_covariates`
: Standardize finite covariate values before fitting an external-drift model.

`variogram_model`
: Optional explicit `gstat::vgm()` covariance model.

`anisotropy`
: Optional anisotropy parameters passed to the fitted variogram model.

`kriging_control`
: Named controls for kriging neighborhood or fitting behavior.

Details

Requested methods are never silently replaced. Kriging conditions and fit information, TPS selection information, prediction ranges, and method messages are available in the opt-in structured result.

Value

By default, a named list of `terra::SpatRaster` surfaces. With `return = "result"`, a documented `potentiomap_result`.

Examples

```
data("synthetic_wells")
pts <- ps_make_points(synthetic_wells, "x", "y", "gw_elevation",
                      "well_id", "EPSG:26916")
result <- ps_interpolate(
  pts, methods = c("IDW", "TPS"), grid_res = 150,
  return = "result", support = TRUE
)
ps_surfaces(result)
ps_diagnostics(result, "TPS")
```

`ps_interpolate_grouped` *Interpolate grouped groundwater observations*

Description

Runs separate analyses for explicit event, aquifer, water-bearing-unit, season, or other grouping columns. Training records are never combined across groups. A shared template controls only output geometry; it does not share observations or fit information.

Usage

```
ps_interpolate_grouped(
  data,
  group_cols,
  value = "Z",
  x = "x",
  y = "y",
  name_col = NULL,
  crs = NULL,
  template_mode = c("shared", "group"),
  mask = NULL,
  mask_mode = c("shared", "group"),
  output_dir = NULL,
  progress = NULL,
  ...
)
```

Arguments

`data`	Observation data accepted by `ps_make_points()`.
`group_cols`	One or more explicit grouping columns.
`value`, `x`, `y`, `name_col`, `crs`	Point-preparation arguments.
`template_mode`	`"shared"` for one output grid or `"group"` for a grid derived independently within each group.
`mask`	Optional shared mask, or a named list when `mask_mode = "group"`.
`mask_mode`	`"shared"` or `"group"`.
`output_dir`	Optional directory for group exports. No files are written when `NULL`.
`progress`	Optional function called as `progress(index, total, group_id, status)`.
`...`	Arguments passed to `ps_interpolate()`. Structured results are always retained by group.

Value

A potentiomap_grouped_result containing results, a deterministic manifest, group_keys, captured conditions, and the original call. Empty and failed groups remain in the manifest.

Examples

```
data("synthetic_wells")
grouped <- transform(
  synthetic_wells,
  event = rep(c("spring", "autumn"), each = 16),
  unit = rep(c("upper", "lower"), times = 16)
)
result <- ps_interpolate_grouped(
  grouped, c("event", "unit"), value = "gw_elevation",
  name_col = "well_id", crs = "EPSG:26916",
  methods = "IDW", grid_res = 300
)
result$manifest
```

---

ps_interpolate_regions

Interpolate explicit regions independently

Description

Interpolate explicit regions independently

Usage

```
ps_interpolate_regions(
  points,
  regions,
  region_id,
  methods = "TPS",
  template = NULL,
  grid_res = NULL,
  interpolation_control = list(),
  mosaic = TRUE,
  overlap_priority = NULL,
  progress = NULL
)
```

Arguments

points	Groundwater-head points.
regions	Region polygons.
region_id	Unique region field.

methods Interpolation methods.

template, grid_res
Mapping geometry controls.

interpolation_control
Named interpolation arguments.

mosaic Return a boundary-preserving mosaic.

overlap_priority
Required region priority when regions overlap.

progress Optional callback.

Value

A potentiomap_regional_result object. No groundwater-flow boundary conditions are imposed and no smoothing occurs across region boundaries.

Examples

```
data("synthetic_wells", "synthetic_regions")
p <- ps_make_points(synthetic_wells, "x", "y", "gw_elevation",
                    "well_id", "EPSG:26916")
regions <- terra::vect(synthetic_regions, geom = "wkt", crs = "EPSG:26916")
regional <- ps_interpolate_regions(p, regions, "region_id", "IDW",
                                   grid_res = 300)
regional$region_method_manifest
# Independent regional fits do not impose flow-boundary conditions.
```

---

ps_make_points *Make groundwater observation points*

Description

Converts a coordinate table, sf point object, or terra point vector to a SpatVector with standard Z and Name fields. Optional unit and vertical reference information is retained as package metadata; a horizontal CRS is never interpreted as a vertical datum.

Usage

```
ps_make_points(
  data,
  x = "x",
  y = "y",
  value,
  name_col = NULL,
  crs = NULL,
  metadata = NULL,
  head_unit = NULL,
  output_unit = NULL,
```

```
  vertical_datum = NULL,
  surface_reference = NULL,
  metadata_mode = c("legacy", "warn", "strict"),
  invalid_action = c("drop", "error")
)
```

Arguments

data	A data frame, sf object, or terra::SpatVector containing point observations.
x, y	Coordinate column names for tabular data.
value	Groundwater elevation column name.
name_col	Optional well or station name column.
crs	Coordinate reference system for tabular data, such as "EPSG:26916".
metadata	Optional named list containing scientific metadata.
head_unit	Unit of value; accepted spellings represent metres or the international foot.
output_unit	Desired unit of Z. When supplied with head_unit, values are converted using exactly 1 ft = 0.3048 m.
vertical_datum	Documented vertical datum. It is recorded, not transformed.
surface_reference	Measurement reference, such as "land_surface" or "measuring_point".
metadata_mode	One of "legacy", "warn", or "strict". Legacy mode accepts numeric data without metadata; warning mode reports omissions; strict mode rejects them.
invalid_action	Either "drop" to report and remove invalid records or "error" to stop.

Value

A point terra::SpatVector with standardized attributes and optional metadata available through ps_metadata(). The dropped_records attribute summarizes invalid observations.

Examples

```
data("synthetic_wells")
pts <- ps_make_points(
  synthetic_wells,
  x = "x", y = "y",
  value = "gw_elevation",
  name_col = "well_id",
  crs = "EPSG:26916",
  head_unit = "m", output_unit = "m",
  vertical_datum = "synthetic example datum",
  surface_reference = "land_surface"
)
pts
```

ps_metadata *Inspect scientific metadata*

Description

Returns the length-unit, vertical-reference, measurement-reference, and depth-sign information attached by `ps_make_points()` or `ps_potentiometric_points()`. Structured interpolation results also retain this information in their input summary.

Usage

```
ps_metadata(x)
```

Arguments

`x`	A potentiomap point object or structured result.

Details

R-level attributes are not guaranteed to survive export to every GIS vector format. Use an output manifest or sidecar table when these details must accompany exported files. Matching units do not establish that two vertical datums are compatible, and potentiomap does not transform vertical datums.

Value

A named list, or `NULL` when no metadata are attached.

Examples

```
data("synthetic_wells")
pts <- ps_make_points(
  synthetic_wells, "x", "y", "gw_elevation", "well_id", "EPSG:26916",
  head_unit = "m", output_unit = "m", vertical_datum = "synthetic datum"
)
ps_metadata(pts)
```

ps_method_disagreement *Map disagreement among interpolation methods*

Description

Computes head and down-gradient-direction differences among compatible surfaces. Directed bearings use circular differences from 0 to 180 degrees; they are not treated as axial orientations.

Usage

```
ps_method_disagreement(
  surfaces,
  head_measures = c("range", "sd", "mad", "mean_pairwise_absolute"),
  gradient = TRUE,
  min_gradient = 1e-05,
  support = c("intersection", "union"),
  minimum_methods = 2
)
```

Arguments

surfaces	Named compatible one-layer head rasters or a potentiomap_result.
head_measures	Requested head-disagreement measures.
gradient	Include gradient-direction disagreement.
min_gradient	Gradients below this magnitude are undefined.
support	Use only the intersection or allow the union of finite cells.
minimum_methods	Minimum finite component count. The default is all methods for intersection and one for union.

Value

A potentiomap_disagreement with disagreement rasters, pairwise and direction summaries, a flat mask, and method-pair manifest.

Examples

```
r <- terra::rast(nrows = 3, ncols = 3, xmin = 0, xmax = 3, ymin = 0, ymax = 3,
                 crs = "EPSG:26920", vals = 1:9)
d <- ps_method_disagreement(list(a = r, b = r + 1))
d$pairwise_summary
# Head offsets are method differences, not statistical uncertainty.
```

---

ps_network_thinning *Evaluate reproducible monitoring-network thinning scenarios*

Description

Evaluate reproducible monitoring-network thinning scenarios

Usage

```
ps_network_thinning(
  points,
  retain = c(0.75, 0.5, 0.25),
  design = c("random", "spatial_coverage", "user"),
  method = "TPS",
  repeats = 10,
  subsets = NULL,
  template = NULL,
  mask = NULL,
  grid_res = NULL,
  seed = 1,
  progress = NULL
)
```

Arguments

`points`	Groundwater-head points.
`retain`	Fractions or exact retained counts.
`design`	Random, spatial-coverage, or user subsets.
`method`	Interpolation method.
`repeats`	Number of planned runs per fraction.
`subsets`	User-supplied lists of retained IDs.
`template`, `mask`, `grid_res`	Fixed surface controls.
`seed`	Deterministic seed.
`progress`	Optional callback.

Value

A `potentiomap_network_thinning` object.

Examples

```
data("synthetic_wells")
p <- ps_make_points(synthetic_wells[1:10, ], "x", "y", "gw_elevation",
                    "well_id", "EPSG:26916")
thin <- ps_network_thinning(p, retain = 0.7, method = "IDW",
                            repeats = 2, grid_res = 350, seed = 8)
thin$summary
# Full-surface differences are descriptive, not error against truth.
```

`ps_potentiometric_points` *Build potentiometric points from depth-to-water measurements*

Description

Calculates groundwater elevation from land-surface elevation and a documented depth convention. Surface elevation can come from a DEM, a column in the depth table, or separate surface-elevation points. Separate points are matched by name where possible and otherwise interpolated by IDW.

Usage

```
ps_potentiometric_points(
  data,
  x = "x",
  y = "y",
  depth_col,
  surface = NULL,
  surface_col = NULL,
  name_col = NULL,
  surface_name_col = name_col,
  crs = NULL,
  idw_power = 2,
  metadata = NULL,
  depth_unit = NULL,
  surface_unit = NULL,
  output_unit = NULL,
  vertical_datum = NULL,
  surface_reference = NULL,
  depth_sign = NULL,
  measuring_point_offset = NULL,
  metadata_mode = c("legacy", "warn", "strict"),
  invalid_action = c("drop", "error")
)
```

Arguments

`data`	Depth-to-water observations as a data frame, `sf`, or `terra::SpatVector`.
`x`, `y`	Coordinate column names for tabular depth data.
`depth_col`	Depth-to-water column name.
`surface`	A DEM `SpatRaster`, separate surface-elevation observations, or `NULL` when `surface_col` is used.
`surface_col`	Surface-elevation column in `data`, or in `surface` when `surface` is a point/table object.
`name_col`	Optional name column in `data`.

`surface_name_col`
: Optional name column in separate surface observations.

`crs`
: CRS for tabular depth data.

`idw_power`
: Positive IDW power for unmatched surface points.

`metadata`
: Optional named list of scientific metadata.

`depth_unit`, `surface_unit`, `output_unit`
: Supported length units.

`vertical_datum`
: Documented vertical datum; it is recorded, not transformed.

`surface_reference`
: Either `"land_surface"` or `"measuring_point"`.

`depth_sign`
: Either `"positive_down"` or `"signed"`.

`measuring_point_offset`
: Height of the measuring point above land surface, expressed in `surface_unit`. It is not inferred.

`metadata_mode`
: One of `"legacy"`, `"warn"`, or `"strict"`.

`invalid_action`
: Either `"drop"` or `"error"`.

Details

With `depth_sign = "positive_down"`, groundwater elevation equals reference elevation minus depth. With `depth_sign = "signed"`, a negative value denotes water below the reference and is added to the reference elevation. A measuring-point offset is added to land-surface elevation only when `surface_reference = "measuring_point"`. potentiomap converts supported units but does not transform vertical datums or guess measuring-point offsets.

Value

A point `terra::SpatVector` with `surface_elevation`, `depth_to_water`, and `Z` in `output_unit`, plus metadata available through `ps_metadata()`.

Examples

```
data("synthetic_wells")
gw <- ps_potentiometric_points(
  synthetic_wells, "x", "y", "depth_to_water",
  surface_col = "surface_elevation", name_col = "well_id",
  crs = "EPSG:26916", depth_unit = "m", surface_unit = "m",
  output_unit = "m", vertical_datum = "synthetic example datum",
  surface_reference = "land_surface", depth_sign = "positive_down"
)
head(terra::values(gw))
```

ps_prediction_support *Describe prediction support and extrapolation*

Description

Classifies raster cells using the training-point convex hull, distance to the nearest observation, an optional maximum distance, mask membership, and finite prediction availability. The convex hull is a training-network geometry, not an aquifer boundary. Predictions are described, not removed.

Usage

```
ps_prediction_support(
  points,
  surface = NULL,
  template = NULL,
  mask = NULL,
  max_distance = NULL,
  allow_geographic = FALSE
)
```

Arguments

points	Training observations as a point SpatVector, sf object, or object readable by terra::vect().
surface	Optional one-layer prediction SpatRaster.
template	Optional one-layer template when surface is not supplied.
mask	Optional polygon mask used to classify cells.
max_distance	Optional positive distance threshold in projected map units.
allow_geographic	Allow longitude/latitude calculations with a classed warning. The default is FALSE.

Details

Distance calculations require projected coordinates by default. An explicit override reports that longitude/latitude degrees are not linear ground units.

Value

A potentiomap_support list containing rasters, a reason-code lookup table, a cell-level records table, summary, the standardized training points, and call. Stable support classes are supported, outside_training_hull, beyond_maximum_distance, outside_mask, prediction_unavailable, and multiple_limitations.

Examples

```
data("synthetic_wells")
pts <- ps_make_points(synthetic_wells, "x", "y", "gw_elevation",
                      "well_id", "EPSG:26916")
surface <- ps_interpolate(pts, methods = "IDW", grid_res = 200)$IDW
support <- ps_prediction_support(pts, surface = surface,
                                 max_distance = 1000)
support$summary
```

`ps_quicklook` *Draw a quicklook surface plot*

Description

Draw a quicklook surface plot

Usage

```
ps_quicklook(
  surface,
  contours = NULL,
  points = NULL,
  file = NULL,
  title = "Potentiometric surface",
  label_points = TRUE,
  width = 1600,
  height = 1200,
  res = 180,
  contour_units = NULL,
  label_contours = TRUE,
  overwrite = TRUE
)
```

Arguments

`surface`	One-layer `SpatRaster`.
`contours`	Optional contour `SpatVector` or contour result.
`points`	Optional observation points.
`file`	Optional PNG path. No file is written when `NULL`.
`title`	Plot title.
`label_points`	Label points with `Name` and `Z` when available.
`width`, `height`, `res`	Positive PNG dimensions and resolution.
`contour_units`	Optional units appended to contour labels.
`label_contours`	Draw contour labels.
`overwrite`	Overwrite `file` when it exists.

Value

Invisibly returns `file`.

Examples

```
data("synthetic_wells")
pts <- ps_make_points(synthetic_wells, "x", "y", "gw_elevation",
                      "well_id", "EPSG:26916")
surface <- ps_interpolate(pts, methods = "IDW", grid_res = 150)$IDW
ps_quicklook(surface, points = pts, title = "Synthetic IDW")
```

---

ps_report *Render a package-owned technical report*

Description

Render a package-owned technical report

Usage

```
ps_report(
  x,
  output_file,
  format = c("html", "docx"),
  title = NULL,
  sections = "auto",
  include_session = TRUE,
  include_conditions = TRUE,
  overwrite = FALSE
)
```

Arguments

`x`	A potentiomap analysis or interpolation result.
`output_file`	Selected `.html` or `.docx` output.
`format`	HTML or Word.
`title`	Optional safely escaped title.
`sections`	`"auto"` or a character vector selecting from `"summary"`, `"metadata"`, `"settings"`, `"validation"`, `"uncertainty"`, `"change_network"`, `"conditions"`, `"limitations"`, and `"session"`.
`include_session`	Include session information.
`include_conditions`	Include captured conditions.
`overwrite`	Permit replacement of an existing output.

Value

An invisible report manifest.

Examples

```
data("synthetic_wells")
p <- ps_make_points(synthetic_wells, "x", "y", "gw_elevation",
                    "well_id", "EPSG:26916")
checked <- ps_check_observations(p, value = "Z", id = "Name")
if (rmarkdown::pandoc_available()) {
  file <- tempfile(fileext = ".html")
  ps_report(checked, file, "html", include_session = FALSE)
}
# Generated reports are not professional certification.
```

---

ps_sample_aoi *Make the sample area-of-interest polygon*

Description

Make the sample area-of-interest polygon

Usage

```
ps_sample_aoi()
```

Value

A `terra::SpatVector` polygon in EPSG:26916. Coordinates are synthetic and expressed in metres.

Examples

```
aoi <- ps_sample_aoi()
aoi
```

---

ps_screen_groups *Assign or validate monitoring-well screen groups*

Description

Preserves an existing water-bearing-unit field, applies explicit interval rules, or creates descriptive depth/elevation bins. Screen intervals alone do not establish hydrostratigraphic identity, and depth bins are never labeled as aquifers.

Usage

```
ps_screen_groups(
  data,
  mode = c("existing", "rules", "depth_bins"),
  unit_col = NULL,
  screen_top,
  screen_bottom,
  rules = NULL,
  breaks = NULL,
  labels = NULL,
  overlap_required = 0.5,
  ambiguous_action = c("report", "error")
)
```

Arguments

`data`	Observation data frame.
`mode`	Existing labels, explicit interval rules, or descriptive bins.
`unit_col`	Existing-label column.
`screen_top, screen_bottom`	Screen-elevation column names; top must be greater than bottom.
`rules`	Data frame with `unit`, `top`, and `bottom` absolute elevations.
`breaks, labels`	Bin specification for `depth_bins`.
`overlap_required`	Minimum fraction of screen length overlapping a rule.
`ambiguous_action`	Return or error on multiple qualifying groups.

Value

A `potentiomap_screen_groups` with assignments, overlap records, ambiguous/unclassified records, rules and summary.

Examples

```
d <- data.frame(well = c("A", "B"), top = c(10, 5), bottom = c(8, 1))
rules <- data.frame(unit = c("shallow", "deep"), top = c(12, 6), bottom = c(6, 0))
g <- ps_screen_groups(d, "rules", screen_top = "top", screen_bottom = "bottom", rules = rules)
g$assigned
# Rule assignments remain conditional on user-supplied hydrogeologic rules.
```

ps_select_event *Select a groundwater monitoring event*

Description

Selects at most one record per well inside a symmetric time window. The result records the actual measurement span; being inside a window does not by itself make an event synoptic and repeated values are never averaged.

Usage

```
ps_select_event(
  data,
  id,
  datetime,
  center,
  window,
  rule = c("nearest", "earliest", "latest", "best_quality"),
  quality = NULL,
  timezone = "UTC",
  maximum_span = NULL,
  maximum_span_action = c("warn", "error")
)
```

Arguments

data	Observation data frame.
id, datetime	Column names for well ID and measurement time.
center	Target time coercible to POSIXct.
window	Nonnegative seconds, a difftime, or a string accepted by as.difftime().
rule	Deterministic selection rule.
quality	Optional quality column for best_quality; lower sorted value is preferred and time distance breaks ties.
timezone	Time zone used to parse and retain times.
maximum_span	Optional maximum selected-event span.
maximum_span_action	Warn or error when the maximum is exceeded.

Value

A potentiomap_event_selection containing selected, excluded and tie records plus an event summary.

Examples

```
d <- data.frame(well = c("A", "A", "B"),
  time = c("2026-01-01 00:00", "2026-01-01 02:00", "2026-01-01 01:00"))
ev <- ps_select_event(d, "well", "time", "2026-01-01 01:00", 3 * 3600)
ev$selected
# The recorded span still requires a study-specific synoptic judgment.
```

`ps_smooth_surface` *Smooth a potentiometric surface raster*

Description

Applies a focal moving-window smoother to a potentiometric surface raster. This can be useful when an interpolated surface is technically valid but too locally rough for contour development or hydraulic-gradient visualization.

Usage

```
ps_smooth_surface(
  surface,
  window_size = 3,
  method = c("mean", "median"),
  weights = NULL,
  iterations = 1,
  na.rm = TRUE,
  preserve_na = TRUE,
  filename = "",
  overwrite = FALSE
)
```

Arguments

`surface`	A `terra::SpatRaster` potentiometric surface.
`window_size`	Odd integer window size used when `weights` is `NULL`.
`method`	Smoothing statistic. Supported values are `"mean"` and `"median"`.
`weights`	Optional odd-dimension numeric matrix of focal weights. `NA` values in the matrix are ignored by `terra::focal()`.
`iterations`	Number of smoothing passes.
`na.rm`	Ignore missing values inside the focal window.
`preserve_na`	Preserve the original `NA` footprint after smoothing.
`filename`	Optional output raster filename.
`overwrite`	Overwrite `filename` when it exists.

Value

A smoothed `terra::SpatRaster`.

Examples

```
data("synthetic_wells")
pts <- ps_make_points(synthetic_wells, "x", "y", "gw_elevation",
                      "well_id", "EPSG:26916")
s <- ps_interpolate(pts, grid_res = 100)
smoothed <- ps_smooth_surface(s$TPS, window_size = 5)
smoothed
```

`ps_split_domain` *Validate and split an explicit hydrogeologic domain*

Description

Validate and split an explicit hydrogeologic domain

Usage

```
ps_split_domain(
  domain,
  regions,
  region_id,
  points = NULL,
  overlap_action = c("error", "priority"),
  gap_action = c("report", "error"),
  boundary_action = c("error", "assign_by_priority", "duplicate")
)
```

Arguments

`domain`	Domain polygon.
`regions`	Explicit region polygons.
`region_id`	Unique region identifier field.
`points`	Optional monitoring points to assign.
`overlap_action`	Error or preserve priority order.
`gap_action`	Report or error for domain gaps.
`boundary_action`	Error, priority assignment, or duplicate assignments.

Value

A `potentiomap_domain_split` object. Regions are never inferred from monitoring points and invalid geometry is never silently repaired.

Examples

```
data("synthetic_regions")
regions <- terra::vect(synthetic_regions, geom = "wkt", crs = "EPSG:26916")
domain <- terra::as.polygons(terra::ext(500000, 503000, 4640000, 4642500),
                             crs = "EPSG:26916")
split <- ps_split_domain(domain, regions, "region_id")
split$summary
# Explicit regions are not inferred groundwater-flow boundaries.
```

`ps_surface_ensemble` *Combine compatible potentiometric surfaces*

Description

Calculates a cellwise ensemble and method-spread layers from compatible head surfaces. Method spread is disagreement among supplied surfaces, not statistical uncertainty, and an ensemble is not automatically more accurate than a component method.

Usage

```
ps_surface_ensemble(
  surfaces,
  statistic = c("mean", "median", "weighted_mean", "quantile"),
  weights = NULL,
  probabilities = c(0.1, 0.5, 0.9),
  support = c("intersection", "union"),
  minimum_methods = NULL,
  method_metadata = NULL
)
```

Arguments

`surfaces`	Named compatible one-layer head rasters or a `potentiomap_result`.
`statistic`	Cellwise mean, median, named weighted mean, or quantiles.
`weights`	Named nonnegative weights for `weighted_mean`. Attribute `origin` may record how they were derived.
`probabilities`	Quantile probabilities.
`support`	Use only the intersection or allow the union of finite cells.
`minimum_methods`	Minimum finite component count. The default is all methods for intersection and one for union.
`method_metadata`	Optional named metadata list supplementing raster metadata.

Value

A potentiomap_ensemble with ensemble, count, extrema, SD, MAD and method/support manifests.

Examples

```
r <- terra::rast(nrows = 2, ncols = 2, xmin = 0, xmax = 2, ymin = 0, ymax = 2,
                 crs = "EPSG:26920", vals = 1:4)
e <- ps_surface_ensemble(list(a = r, b = r + 2))
e$ensemble
# The spread records method disagreement, not a confidence interval.
```

---

ps_surface_profile *Extract one or more surface profiles along explicit lines*

Description

Extract one or more surface profiles along explicit lines

Usage

```
ps_surface_profile(
  lines,
  surfaces,
  step = NULL,
  n = NULL,
  support = NULL,
  distance_method = c("projected", "geodesic")
)
```

Arguments

lines	One or more line features.
surfaces	Named compatible surfaces or an interpolation result.
step, n	Explicit spacing or count per line.
support	Optional support/uncertainty rasters to extract.
distance_method	Projected or explicit geodesic distance.

Value

A potentiomap_profile with monotonically increasing chainage.

Examples

```
data("synthetic_wells", "synthetic_transect")
p <- ps_make_points(synthetic_wells, "x", "y", "gw_elevation",
                    "well_id", "EPSG:26916")
s <- ps_interpolate(p, methods = "IDW", grid_res = 250)$IDW
line <- terra::vect(synthetic_transect, geom = "wkt", crs = "EPSG:26916")
profile <- ps_surface_profile(line, list(head = s), n = 12)
head(profile$profile)
# Unsupported raster sections remain missing rather than being invented.
```

---

ps_surface_sensitivity

Evaluate explicit interpolation sensitivity scenarios

Description

Evaluate explicit interpolation sensitivity scenarios

Usage

```
ps_surface_sensitivity(
  points,
  method,
  scenarios,
  reference = NULL,
  template = NULL,
  mask = NULL,
  maximum_runs = 100,
  contour_levels = NULL,
  compare_gradient = TRUE,
  seed = 1,
  progress = NULL
)
```

Arguments

`points`	Groundwater-head points.
`method`	Interpolation method.
`scenarios`	Explicit data frame or named parameter grid.
`reference`	Scenario ID or row used as reference.
`template, mask`	Default mapping controls.
`maximum_runs`	Maximum scenario guard.
`contour_levels`	Optional contour levels.
`compare_gradient`	Compare gradient direction.
`seed`	Deterministic seed.
`progress`	Optional callback.

Value

A potentiomap_sensitivity object. No preferred scenario is selected.

Examples

```
data("synthetic_wells")
p <- ps_make_points(synthetic_wells[1:12, ], "x", "y", "gw_elevation",
                    "well_id", "EPSG:26916")
scenarios <- data.frame(idw_power = c(1.5, 2), grid_res = c(300, 300))
sensitivity <- ps_surface_sensitivity(p, "IDW", scenarios, reference = 1)
sensitivity$comparisons[, c("scenario_id", "mean_absolute_difference")]
# Sensitivity comparison does not select a universally preferred setting.
```

---

ps_surface_uncertainty

Quantify model-conditional or resampling surface variability

Description

Quantify model-conditional or resampling surface variability

Usage

```
ps_surface_uncertainty(
  x = NULL,
  points = NULL,
  method = NULL,
  approach = c("kriging_variance", "conditional_simulation", "tps_standard_error",
    "resampling_sensitivity"),
  nsim = 100,
  probabilities = c(0.05, 0.5, 0.95),
  resampling_design = NULL,
  template = NULL,
  mask = NULL,
  keep_realizations = FALSE,
  output_directory = NULL,
  seed = 1,
  progress = NULL,
  exceedance_levels = NULL
)
```

Arguments

x	A structured interpolation result.
points	Points used for resampling sensitivity when x is absent.
method	Interpolation method for resampling.

approach	Uncertainty or sensitivity approach.
nsim	Number of simulations or resamples.
probabilities	Pointwise quantile probabilities.
resampling_design	"case", "jackknife", or a list with a type and spatial group vector.
template, mask	Mapping geometry controls.
keep_realizations	Retain realization rasters in memory.
output_directory	Optional realization directory.
seed	Deterministic seed.
progress	Optional callback.
exceedance_levels	Optional head levels for exceedance probability.

Value

A potentiomap_uncertainty object. Resampling products are sensitivity summaries, not formal confidence intervals.

Examples

```
data("synthetic_wells")
p <- ps_make_points(synthetic_wells[1:14, ], "x", "y", "gw_elevation",
                    "well_id", "EPSG:26916")
fit <- suppressWarnings(ps_interpolate(p, methods = "OK", grid_res = 250,
                                       return = "result"))
uncertainty <- ps_surface_uncertainty(fit, approach = "kriging_variance")
uncertainty$method_manifest
# Kriging variance is conditional on the retained covariance model.
```

---

ps_surfaces *Extract interpolated surfaces*

Description

Extract interpolated surfaces

Usage

```
ps_surfaces(x)
```

Arguments

x	A potentiomap_result or named list of SpatRaster surfaces.

Value

A named list of `terra::SpatRaster` objects.

Examples

```
data("synthetic_wells")
pts <- ps_make_points(synthetic_wells, "x", "y", "gw_elevation",
                      "well_id", "EPSG:26916")
result <- ps_interpolate(pts, methods = "IDW", grid_res = 150,
                         return = "result")
ps_surfaces(result)
```

---

ps_tune_interpolation *Tune interpolation parameters under recorded validation partitions*

Description

Compares explicit candidate configurations using the same deterministic inner partitions. With an outer design, tuning occurs only inside each outer training partition and the selected configuration is evaluated on its outer holdout. Without an outer design, reported performance is tuning performance, not an unbiased estimate of final predictive performance.

Usage

```
ps_tune_interpolation(
  points,
  method,
  candidates,
  inner_design = "spatial_block",
  outer_design = NULL,
  inner_folds = 5,
  outer_folds = 5,
  repeats = 1,
  metric = "rmse",
  minimum_coverage = 0.9,
  template = NULL,
  mask = NULL,
  grid_res = NULL,
  refit = TRUE,
  seed = 1,
  progress = NULL,
  search = c("grid", "random"),
  maximum_runs = 1000
)
```

Arguments

points	Groundwater-head points.
method	One of "TPS", "IDW", "OK", or "UK".
candidates	Candidate table or named parameter list.
inner_design, outer_design	Inner and optional outer validation designs.
inner_folds, outer_folds, repeats	Fold and repeat counts.
metric	Objective metric minimized during selection.
minimum_coverage	Minimum finite-prediction coverage.
template, mask, grid_res	Fixed mapping controls.
refit	Refit the selected configuration to all observations.
seed	Deterministic seed.
progress	Optional callback.
search	Exhaustive grid or reproducible row sampling.
maximum_runs	Maximum candidate-by-fold run guard.

Value

A potentiomap_tuning object.

Examples

```
data("synthetic_wells")
p <- ps_make_points(synthetic_wells[1:12, ], "x", "y", "gw_elevation",
                    "well_id", "EPSG:26916")
tuned <- ps_tune_interpolation(p, "IDW", list(idw_power = c(1.5, 2)),
                               inner_design = "kfold", inner_folds = 3,
                               refit = FALSE, seed = 4)
tuned$candidates[, c("candidate_id", "metric", "coverage", "selected")]
# These are tuning scores, not unbiased final performance estimates.
```

---

ps_validate *Validate potentiometric-surface interpolation*

Description

Evaluates requested methods under an explicit leave-one-out, k-fold, spatial block, cluster, user-fold, or independent validation prediction task. Held-out heads never construct folds and held-out records never enter their training fit. Cross-validation performance is not automatically area-wide map accuracy.

Usage

```
ps_validate(
  points,
  methods = c("TPS", "IDW", "OK", "UK"),
 design = c("loocv", "kfold", "spatial_block", "leave_cluster_out", "user_folds",
    "independent"),
  validation_points = NULL,
  fold_id = NULL,
  cluster = NULL,
  folds = 5,
  repeats = 1,
  block_size = NULL,
  template = NULL,
  mask = NULL,
  grid_res = NULL,
  domain_policy = c("fixed", "training"),
  prediction_mode = c("raster", "direct"),
  metrics = c("me", "mae", "rmse", "medae", "maxae"),
  support = TRUE,
  sampling_weight = NULL,
  interpolation_control = list(),
  seed = 1,
  progress = NULL
)
```

Arguments

points	Training groundwater-head points with stable IDs.
methods	Interpolation methods.
design	Validation design.
validation_points	Independent compatible validation points.
fold_id, cluster	Assignment vector or column.
folds, repeats	Positive counts.
block_size	Spatial-block size in projected units.
template, mask, grid_res	Fixed mapping geometry controls.
domain_policy	Fixed or fold-training-derived raster domain.
prediction_mode	Full raster/extraction sequence or supported direct prediction.
metrics	Error metrics; residual is predicted minus observed.
support	Calculate held-out hull/distance/mask support.
sampling_weight	Optional weights for explicitly independent probability validation records only.

interpolation_control	Named arguments passed to interpolation.
seed	Deterministic master seed.
progress	Optional callback (index, total, run_id, status).

Value

A potentiomap_validation with predictions, metrics, fold/partition/ fit manifests, support summary, settings and captured conditions.

References

Roberts et al. (2017), doi:10.1111/ecog.02881; Wadoux et al. (2021), doi:10.1016/j.ecolmodel.2021.109692.

Examples

```
data("synthetic_wells")
pts <- ps_make_points(synthetic_wells, "x", "y", "gw_elevation",
                      "well_id", "EPSG:26916")
val <- ps_validate(pts, methods = "IDW", design = "kfold", folds = 3,
                   prediction_mode = "direct", seed = 7)
val$metrics
# These scores describe the stated folds, not design-unbiased map accuracy.
```

---

ps_validate_arrows *Validate hydraulic-gradient arrow endpoints*

Description

Samples modeled head at each line base and tip and checks finite raster support along the line. A downhill pass requires a finite line and a tip no higher than the base within tolerance. Geometry is not reversed or bent.

Usage

```
ps_validate_arrows(
  surface,
  arrows,
  tolerance = 1e-06,
  extraction = c("bilinear", "simple")
)
```

Arguments

surface	One-layer modeled-head SpatRaster.
arrows	Line SpatVector or readable vector path.
tolerance	Nonnegative head tolerance.
extraction	Either "bilinear" or "simple" endpoint extraction.

Value

A potentiomap_arrow_validation list with validated arrows, an arrow-level records data frame, and a concise summary.

Examples

```
data("synthetic_wells")
pts <- ps_make_points(synthetic_wells, "x", "y", "gw_elevation",
                      "well_id", "EPSG:26916")
surface <- ps_interpolate(pts, methods = "IDW", grid_res = 150)$IDW
legacy <- ps_flow_arrows(surface, endpoint_action = "none")
checked <- ps_validate_arrows(surface, legacy$arrows)
checked$summary
```

---

ps_validation_plot *Plot validation diagnostics with base graphics*

Description

Plot validation diagnostics with base graphics

Usage

```
ps_validation_plot(
  x,
  type = c("metric", "observed_predicted", "residual_map", "residual_distribution",
    "fold_map", "support", "coverage", "method_conditions"),
  methods = NULL,
  design = NULL,
  support_subset = "all",
  display_limits = NULL,
  legend = TRUE,
  ...
)
```

Arguments

x	A validation or method-comparison result.
type	Diagnostic plot type.
methods, design	Optional subsets.
support_subset	Support subset.
display_limits	Optional explicit axis/value limits. Values remain in returned plot data and requested clipping is disclosed.
legend	Draw a legend.
...	Base graphics arguments.

Value

Plot data invisibly.

Examples

```
data("synthetic_wells")
pts <- ps_make_points(synthetic_wells, "x", "y", "gw_elevation", "well_id", "EPSG:26916")
val <- ps_validate(pts, "IDW", "kfold", folds = 3, prediction_mode = "direct")
ps_validation_plot(val, "observed_predicted")
```

---

ps_variogram *Calculate an empirical groundwater-head variogram*

Description

Calculates an ordinary or residual empirical variogram without fitting a covariance model. Directions follow the gstat convention: degrees clockwise from positive Y (North), periodic over 180 degrees. Pair counts and projected coordinate units are retained.

Usage

```
ps_variogram(
  points,
  formula = Z ~ 1,
  cutoff = NULL,
  width = NULL,
  boundaries = NULL,
  directions = 0,
  direction_tolerance = NULL,
  robust = FALSE,
  cloud = FALSE
)
```

Arguments

points	Groundwater-head points with a Z column.
formula	Head/trend formula, such as Z ~ 1 or Z ~ elevation.
cutoff	Maximum pair distance.
width	Positive lag width.
boundaries	Optional strictly increasing lag upper boundaries.
directions	Direction angles clockwise from North.
direction_tolerance	Horizontal tolerance in degrees.
robust	Use gstat's Cressie robust estimator.
cloud	Return individual pair semivariances.

Value

A potentiomap_variogram with empirical values, formula, directions, point summary, settings and conditions.

References

Pebesma (2004), doi:10.1016/j.cageo.2004.03.012.

Examples

```
data("synthetic_wells")
pts <- ps_make_points(synthetic_wells, "x", "y", "gw_elevation",
                      "well_id", "EPSG:26916")
v <- ps_variogram(pts, cutoff = 3000, width = 300)
head(v$empirical)
# An empirical variogram does not establish one correct covariance model.
```

---

ps_variogram_compare *Compare fitted variogram models*

Description

Fits each requested gstat variogram candidate independently and preserves initial values, fitted parameters, singular status, weighted fitting error, warnings, and errors. Smallest variogram SSE is not proof of best predictive performance.

Usage

```
ps_variogram_compare(
  variogram,
  models = c("Sph", "Exp", "Gau", "Mat"),
  initial = NULL,
  fit_method = 7,
  anisotropy = NULL,
  validation = NULL,
  select = FALSE,
  selection_metric = c("validation_rmse", "weighted_sse")
)
```

Arguments

variogram	A potentiomap_variogram or gstat empirical variogram.
models	Candidate model abbreviations.
initial	Optional model or named model-specific initial values.
fit_method	gstat fit method.
anisotropy	Optional explicit anisotropy result or c(angle, ratio).

validation Optional compatible validation result/table.

select Select a model using an explicitly supplied criterion.

selection_metric
Validation RMSE or weighted variogram SSE.

Value

A potentiomap_variogram_comparison with every fit and ranking.

Examples

```
data("synthetic_wells")
pts <- ps_make_points(synthetic_wells, "x", "y", "gw_elevation",
                      "well_id", "EPSG:26916")
cmp <- ps_variogram_compare(ps_variogram(pts), c("Sph", "Exp"))
cmp$ranking
# Weighted SSE alone is not a predictive-performance guarantee.
```

---

ps_vertical_gradient *Calculate a vertical hydraulic gradient*

Description

Uses absolute upper and lower observation elevations. With an upward-positive convention, the primary gradient is (lower_head - upper_head) / (upper_elevation - lower_elevation). The result indicates potential vertical direction and is not vertical groundwater flux.

Usage

```
ps_vertical_gradient(
  upper_head,
  lower_head,
  upper_elevation,
  lower_elevation,
  positive = c("upward", "downward"),
  tolerance = 0,
  align = c("error", "to_upper", "to_lower", "template"),
  template = NULL,
  event_metadata = NULL
)
```

Arguments

upper_head, lower_head
Numeric paired heads or one-layer rasters.

upper_elevation, lower_elevation
Matching absolute elevations, not unidentified depths.

positive	Sign convention.
tolerance	Nonnegative near-zero gradient tolerance.
align, template	Explicit raster alignment controls.
event_metadata	Optional list documenting compatible event, datum, units, interval and screen-midpoint assumptions.

Value

A potentiomap_vertical_gradient with component products, direction class, support, and sign convention. No flux field is returned.

Examples

```
vg <- ps_vertical_gradient(10, 12, 100, 90)
vg$gradient
# The positive gradient indicates upward driving potential, not flux.
```

---

ps_well_influence *Calculate conditional leave-one-well influence*

Description

Calculate conditional leave-one-well influence

Usage

```
ps_well_influence(
  points,
  method = "TPS",
  template = NULL,
  mask = NULL,
  grid_res = NULL,
  contour_levels = NULL,
  difference_threshold = NULL,
  interpolation_control = list(),
  progress = NULL
)
```

Arguments

points	Groundwater-head points.
method	Interpolation method.
template, mask, grid_res	Fixed surface controls.
contour_levels	Optional comparison contours.

`difference_threshold`
: Optional absolute-difference area threshold.

`interpolation_control`
: Named interpolation controls.

`progress`
: Optional callback.

Value

A `potentiomap_well_influence` object. Influence is conditional on this network and method and is not evidence that a well is erroneous.

Examples

```
data("synthetic_wells")
p <- ps_make_points(synthetic_wells[1:7, ], "x", "y", "gw_elevation",
                    "well_id", "EPSG:26916")
influence <- ps_well_influence(p, method = "IDW", grid_res = 350)
influence$influence[, c("well_id", "held_out_residual", "status")]
# High conditional influence is not an automatic data-error classification.
```

`synthetic_anisotropic_points` *Synthetic anisotropic head points*

Description

A fixed-seed elongated periodic field with a generating major-continuity direction of 35 degrees clockwise from north. Sampling noise and finite extent mean exploratory estimates need not equal 35 degrees exactly.

Usage

```
synthetic_anisotropic_points
```

Format

A 45-row projected point table with head in arbitrary consistent elevation units.

Source

Generated by `data-raw/generate-expansion-data.R` with seed 20260717.

synthetic_candidate_sites *Synthetic candidate monitoring sites*

Description

Twelve explicit candidate coordinates with exclusions, costs, and arbitrary user scores for constraint and ranking examples. The coordinates are not proposed drill sites.

Usage

```
synthetic_candidate_sites
```

Format

A 12-row data frame in EPSG:26916.

Source

Generated by `data-raw/generate-expansion-data.R`.

synthetic_covariates *Synthetic surface shapes and hydrogeologic covariates*

Description

A small projected grid containing land-surface elevation, distance from a valley axis, and known planar, bowl, saddle, and elongated-valley head surfaces. These are mathematical fixtures, not real aquifers.

Usage

```
synthetic_covariates
```

Format

A 224-row grid in EPSG:26916; elevations and heads are metres and distances are metres.

Source

Generated by `data-raw/generate-expansion-data.R`.

synthetic_dem *Synthetic DEM raster*

Description

A small artificial DEM matching `synthetic_wells`, stored as a packed `terra` raster. Use `terra::rast(synthetic_dem)` to unpack it.

Usage

```
synthetic_dem
```

Format

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

Examples

```
data("synthetic_dem")
dem <- terra::rast(synthetic_dem)
dem
```

synthetic_events *Synthetic repeated groundwater-monitoring events*

Description

Two small UTC monitoring events on a planar-to-gently-curved synthetic head field. Membership changes between events and event B includes a known head decline. Values are generated, not observations from a real aquifer.

Usage

```
synthetic_events
```

Format

A data frame with 36 rows and fields for well ID, projected coordinates (metres, EPSG:26916), UTC time, event, head (metres), quality, unit, vertical datum, and explicit water-bearing unit.

Source

Generated by `data-raw/generate-expansion-data.R` with seed 20260717.

synthetic_nested_wells *Synthetic nested monitoring wells*

Description

Eight two-interval nests containing known upward, downward, and near-zero vertical-gradient cases. Screen and representative elevations are absolute metres in the named synthetic vertical datum.

Usage

```
synthetic_nested_wells
```

Format

A data frame with 16 interval records.

Source

Generated by `data-raw/generate-expansion-data.R`.

synthetic_regions *Two synthetic interpolation compartments*

Description

Adjacent disconnected-analysis rectangles supplied as explicit WKT. They are user-defined compartments, not inferred aquifers or flow boundaries.

Usage

```
synthetic_regions
```

Format

A two-row data frame with region IDs, bounds, WKT, and CRS.

Source

Generated by `data-raw/generate-expansion-data.R`.

synthetic_surface_points *Synthetic surface elevation measurement points*

Description

Artificial land-surface elevation points for analyses that do not start with a DEM raster.

Usage

```
synthetic_surface_points
```

Format

A data frame with coordinate, surface-elevation, and name columns.

Examples

```
data("synthetic_surface_points")
head(synthetic_surface_points)
```

synthetic_transect *Synthetic cross-section transect*

Description

One projected polyline for profile and cross-section examples.

Usage

```
synthetic_transect
```

Format

A one-row data frame with WKT and EPSG:26916.

Source

Generated by `data-raw/generate-expansion-data.R`.

synthetic_validation_points *Synthetic independent validation points*

Description

Fourteen fixed-seed points sampled independently from a known planar head field with small Gaussian measurement perturbations.

Usage

```
synthetic_validation_points
```

Format

A 14-row projected point table in EPSG:26916 with head in metres.

Source

Generated by `data-raw/generate-expansion-data.R` with seed 20260717.

synthetic_wells *Synthetic groundwater monitoring wells*

Description

A small artificial monitoring dataset with coordinates, land-surface elevation, positive depth below land surface, and calculated groundwater elevation. Elevations and depths are synthetic metres relative to a synthetic example datum.

Usage

```
synthetic_wells
```

Format

A data frame with 32 rows and 6 columns:

well_id Synthetic well identifier.

x Synthetic easting in EPSG:26916 metres.

y Synthetic northing in EPSG:26916 metres.

surface_elevation Synthetic land-surface elevation in metres.

depth_to_water Positive depth below land surface in metres.

gw_elevation Synthetic groundwater elevation in metres.

Examples

```
data("synthetic_wells")
head(synthetic_wells)
```

Index