Package: XGeoRTR (via r-universe)

May 8, 2026

Title Backend-Neutral Explainable Geometry State and Operators

Date 2026-04-25

Version 0.7.0

Author Frederic Bertrand [cre, aut] (ORCID:
<https://orcid.org/0000-0002-0837-8281>)

Maintainer Frederic Bertrand <frederic.bertrand@lecnam.net>

Description Provides the platform layer for explanation geometry in R. The package standardizes generic explanation tables into a normalized backend state object, computes embeddings, diagnostics, and multiscale level-of-detail summaries, and serializes backend-neutral state for reproducible workflows. It also exposes selected long-table and regular-grid views for downstream use-case packages. Rendering and viewport orchestration are delegated to downstream frontends such as 'ggWebGL'.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports cli, jsonlite

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

VignetteBuilder knitr

Config/testthat/edition 3

Config/Needs/website pkgdown

URL https://fbertran.github.io/XGeoRTR/,
https://github.com/fbertran/XGeoRTR

BugReports https://github.com/fbertran/XGeoRTR/issues

NeedsCompilation no

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

Date/Publication 2026-05-08 17:47:40 UTC

RemoteUrl https://github.com/cran/XGeoRTR

RemoteRef HEAD

RemoteSha d8bcc1b8bc84c9c1971982a69e33e6f4926a4a30

Contents

`as_xgeo_state` *Coerce inputs to* `xgeo_state`

Description

Coerce inputs to `xgeo_state`

Usage

```
as_xgeo_state(x, ...)
```

Arguments

`x`	An object to coerce.
`...`	Passed to method-specific implementations.

Value

An xgeo_state object.

Examples

```
as_xgeo_state(matrix(c(1, -1, 2, 0), nrow = 2))

as_xgeo_state(
  data.frame(
    point_id = c("p1", "p1", "p2"),
    feature = c("f1", "f2", "f1"),
    x = c(0, 0, 1),
    y = c(0, 0, 1),
    value = c(1, -0.5, 0.75)
  )
)
```

---

build_xgeo_lod *Build density-grid LOD summaries for a state*

Description

Build density-grid LOD summaries for a state

Usage

```
build_xgeo_lod(
  state,
  embedding = NULL,
  levels = c(16L, 32L, 64L),
  color_by = c("count", "mean_value"),
  name = NULL,
  auto_threshold = 200L
)
```

Arguments

state	An xgeo_state object.
embedding	Optional embedding name. Defaults to the active embedding.
levels	Integer vector of grid resolutions.
color_by	Statistic stored in each density grid.
name	Optional LOD bundle name.
auto_threshold	Point count threshold used by lod_level = "auto".

Value

The updated xgeo_state.

Examples

```
state <- xgeo_state(matrix(c(1, -1, 2, 0), nrow = 2))
state <- build_xgeo_lod(state, levels = c(4L, 8L), auto_threshold = 2L)

names(state$lod$items)
```

`compute_xgeo_diagnostics` *Compute embedding diagnostics for a state*

Description

Compute embedding diagnostics for a state

Usage

```
compute_xgeo_diagnostics(
  state,
  embedding = NULL,
  source = c("explanations", "point_meta", "points"),
  k = 10L,
  name = NULL
)
```

Arguments

`state`	An `xgeo_state` object.
`embedding`	Optional embedding name. Defaults to the active embedding.
`source`	Reference source space used for neighbour comparisons.
`k`	Number of neighbours.
`name`	Optional diagnostic bundle name.

Value

The updated `xgeo_state`.

Examples

```
state <- as_xgeo_state(
  data.frame(
    point_id = rep(paste0("p", 1:4), each = 2),
    feature = rep(c("f1", "f2"), times = 4),
    x = c(0, 0, 1, 1, 0, 0, 1, 1),
    y = c(0, 0, 0, 0, 1, 1, 1, 1),
    value = c(0.2, 0.7, 0.4, 0.1, 0.8, 0.6, 0.5, 0.3)
  ),
  point_id_col = "point_id",
```

```
  feature_col = "feature"
)
state <- compute_xgeo_embedding(state, method = "pca", source = "explanations", dims = 2)
state <- compute_xgeo_diagnostics(
  state,
  embedding = "pca_explanations",
  source = "explanations",
  k = 1
)

names(state$attributes$diagnostics$items)
```

`compute_xgeo_embedding` *Compute and attach a platform embedding*

Description

Compute and attach a platform embedding

Usage

```
compute_xgeo_embedding(
  state,
  source = c("explanations", "point_meta", "points"),
  method = c("pca", "umap"),
  dims = 2L,
  name = NULL,
  ...
)
```

Arguments

`state`	An `xgeo_state` object.
`source`	Source matrix used to build the embedding. One of `"explanations"`, `"point_meta"`, or `"points"`.
`method`	Embedding backend. `"pca"` is always available; `"umap"` requires `uwot`.
`dims`	Number of output dimensions.
`name`	Optional embedding name. Defaults to `<method>_<source>`.
`...`	Passed to backend-specific implementations.

Value

The updated `xgeo_state`.

Examples

```
state <- as_xgeo_state(
  data.frame(
    point_id = rep(paste0("p", 1:4), each = 2),
    feature = rep(c("f1", "f2"), times = 4),
    x = c(0, 0, 1, 1, 0, 0, 1, 1),
    y = c(0, 0, 0, 0, 1, 1, 1, 1),
    value = c(0.2, 0.7, 0.4, 0.1, 0.8, 0.6, 0.5, 0.3)
  ),
  point_id_col = "point_id",
  feature_col = "feature"
)
state <- compute_xgeo_embedding(state, method = "pca", source = "explanations", dims = 2)

names(state$attributes$embeddings$items)
```

---

read_xgeo_state *Read a backend state from JSON*

Description

Read a backend state from JSON

Usage

```
read_xgeo_state(path)
```

Arguments

path	Path to a JSON state file.

Value

An xgeo_state object.

Examples

```
state <- xgeo_state(matrix(c(1, -1, 2, 0), nrow = 2))
path <- tempfile(fileext = ".json")
write_xgeo_state(state, path)

restored <- read_xgeo_state(path)
class(restored)
```

set_active_embedding *Set the active embedding on a state*

Description

Set the active embedding on a state

Usage

```
set_active_embedding(state, name)
```

Arguments

state	An xgeo_state object.
name	Name of an embedding stored in state$attributes$embeddings$items.

Value

The updated xgeo_state.

Examples

```
state <- as_xgeo_state(
  data.frame(
    point_id = rep(paste0("p", 1:4), each = 2),
    feature = rep(c("f1", "f2"), times = 4),
    x = c(0, 0, 1, 1, 0, 0, 1, 1),
    y = c(0, 0, 0, 0, 1, 1, 1, 1),
    value = c(0.2, 0.7, 0.4, 0.1, 0.8, 0.6, 0.5, 0.3)
  ),
  point_id_col = "point_id",
  feature_col = "feature"
)
state <- compute_xgeo_embedding(state, method = "pca", source = "explanations", dims = 2)
state <- set_active_embedding(state, "pca_explanations")

state$attributes$embeddings$active
```

set_xgeo_lod *Set the active level-of-detail state on a state*

Description

Set the active level-of-detail state on a state

Usage

```
set_xgeo_lod(state, name = NULL, level = NULL)
```

Arguments

state	An xgeo_state object.
name	Optional LOD bundle name stored in stateloditems.
level	Optional level inside the selected LOD bundle.

Value

The updated xgeo_state.

Examples

```
state <- xgeo_state(matrix(c(1, -1, 2, 0), nrow = 2))
state <- build_xgeo_lod(state, levels = c(4L, 8L), auto_threshold = 2L)
state <- set_xgeo_lod(state, name = "density_grid_spatial", level = "4")

state$lod$active
```

---

set_xgeo_selection *Set explicit point and feature selection on a state*

Description

Set explicit point and feature selection on a state

Usage

```
set_xgeo_selection(state, point_ids = NULL, features = NULL)
```

Arguments

state	An xgeo_state object.
point_ids	Optional character vector of selected point ids.
features	Optional character vector of selected feature ids.

Value

The updated xgeo_state.

Examples

```
state <- as_xgeo_state(
  data.frame(
    point_id = c("p1", "p1", "p2", "p2"),
    feature = c("f1", "f2", "f1", "f2"),
    x = c(0, 0, 1, 1),
    y = c(0, 0, 1, 1),
    value = c(1, -0.25, 0.75, 2)
  ),
  point_id_col = "point_id",
  feature_col = "feature"
)
state <- set_xgeo_selection(state, point_ids = "p1", features = "f2")

xgeo_selection(state)
```

---

validate_xgeo_state *Validate an* xgeo_state *object*

Description

Validate an xgeo_state object

Usage

```
validate_xgeo_state(x)
```

Arguments

x	An object to validate.

Value

x, invisibly, when validation succeeds.

Examples

```
state <- xgeo_state(matrix(c(1, -1, 2, 0), nrow = 2))

validate_xgeo_state(state)
```

write_xgeo_state *Write a backend state to JSON*

Description

Write a backend state to JSON

Usage

```
write_xgeo_state(state, path, pretty = TRUE)
```

Arguments

state	An xgeo_state object.
path	Output file path.
pretty	Whether to pretty-print the JSON.

Value

The normalized output path, invisibly.

Examples

```
state <- xgeo_state(matrix(c(1, -1, 2, 0), nrow = 2))
path <- tempfile(fileext = ".json")

write_xgeo_state(state, path)
file.exists(path)
```

xgeo_attributes *Get attributes from an* xgeo_state

Description

Get attributes from an xgeo_state

Usage

```
xgeo_attributes(state)
```

Arguments

state	An xgeo_state object.

Value

The attributes field.

Examples

```
state <- xgeo_state(matrix(c(1, 2, 3, 4), nrow = 2))

names(xgeo_attributes(state))
```

`xgeo_explanation_table` *Build a long explanation table from backend state*

Description

`xgeo_explanation_table()` exposes the selected explanation records together with point coordinates and metadata. It is renderer-agnostic and contains no use-case-specific presentation semantics.

Usage

```
xgeo_explanation_table(state, selected = TRUE)
```

Arguments

`state`	An `xgeo_state` object.
`selected`	Whether to apply the state's point and feature selection.

Value

A data frame containing `point_id`, `feature`, `value`, `x`, `y`, `z`, and any point-, feature-, prediction-, or uncertainty-level metadata.

Examples

```
state <- as_xgeo_state(
  data.frame(
    point_id = c("p1", "p1", "p2"),
    feature = c("f1", "f2", "f1"),
    x = c(0, 0, 1),
    y = c(0, 0, 1),
    value = c(1, -0.5, 0.75),
    cluster = c("A", "A", "B")
  ),
  point_id_col = "point_id",
  feature_col = "feature"
)

xgeo_explanation_table(state)
```

xgeo_geometry *Get geometry from an* xgeo_state

Description

Get geometry from an xgeo_state

Usage

```
xgeo_geometry(state)
```

Arguments

state	An xgeo_state object.

Value

The geometry field.

Examples

```
state <- xgeo_state(matrix(c(1, 2, 3, 4), nrow = 2))

xgeo_geometry(state)$points
```

xgeo_indices *Get indices from an* xgeo_state

Description

Get indices from an xgeo_state

Usage

```
xgeo_indices(state)
```

Arguments

state	An xgeo_state object.

Value

The indices field.

Examples

```
state <- xgeo_state(matrix(c(1, 2, 3, 4), nrow = 2))

xgeo_indices(state)
```

xgeo_metadata *Get metadata from an* xgeo_state

Description

Get metadata from an xgeo_state

Usage

```
xgeo_metadata(state)
```

Arguments

state	An xgeo_state object.

Value

The metadata field.

Examples

```
state <- xgeo_state(
  matrix(c(1, 2, 3, 4), nrow = 2),
  metadata = list(source = "demo-state")
)

xgeo_metadata(state)
```

xgeo_point_values *Aggregate explanation values per point*

Description

xgeo_point_values() exposes a selected, renderer-neutral point table with coordinates, aggregated explanation values, and point-level metadata.

Usage

```
xgeo_point_values(state, aggregate = sum, selected = TRUE)
```

Arguments

state	An xgeo_state object.
aggregate	Aggregation function applied across selected features per point. Defaults to sum.
selected	Whether to apply the state's point and feature selection.

Value

A data frame containing `point_id`, `x`, `y`, `z`, `value`, and any point-, prediction-, or uncertainty-level metadata.

Examples

```
state <- as_xgeo_state(
  data.frame(
    point_id = c("p1", "p1", "p2", "p2"),
    feature = c("f1", "f2", "f1", "f2"),
    x = c(0, 0, 1, 1),
    y = c(0, 0, 1, 1),
    value = c(1, -0.25, 0.75, 2),
    cluster = c("A", "A", "B", "B")
  ),
  point_id_col = "point_id",
  feature_col = "feature"
)
state <- set_xgeo_selection(state, features = "f1")

xgeo_point_values(state)
```

---

xgeo_regular_grid *Validate and normalize regular-grid data*

Description

`xgeo_regular_grid()` validates a complete 2D regular grid and returns the grid vectors and value matrix without creating a renderer-specific object.

Usage

```
xgeo_regular_grid(data, x = "x", y = "y", value = "value")
```

Arguments

`data`	A data frame containing x, y, and value columns.
`x`	Name of the x-coordinate column.
`y`	Name of the y-coordinate column.
`value`	Name of the value column.

Value

A list with `x`, `y`, and `z`, where `z` is a value matrix indexed by the returned x and y coordinates.

Examples

```
xgeo_regular_grid(
  data.frame(
    x_coord = c(0, 1, 0, 1),
    y_coord = c(0, 0, 1, 1),
    score = c(1, 2, 3, 4)
  ),
  x = "x_coord",
  y = "y_coord",
  value = "score"
)
```

---

xgeo_selection *Get selection from an* xgeo_state

Description

Get selection from an xgeo_state

Usage

```
xgeo_selection(state)
```

Arguments

state	An xgeo_state object.

Value

The selection field.

Examples

```
state <- xgeo_state(matrix(c(1, 2, 3, 4), nrow = 2))
state <- set_xgeo_selection(state, point_ids = state$indices$point_ids[[1]])

xgeo_selection(state)
```

xgeo_state *Create an* xgeo_state

Description

Create an xgeo_state

Usage

```
xgeo_state(
  x,
  embeddings = NULL,
  diagnostics = NULL,
  lod = NULL,
  selection = NULL,
  metadata = list()
)
```

Arguments

x	A matrix, data frame, or object coercible to backend geometry state.
embeddings	Optional embedding state.
diagnostics	Optional diagnostic state.
lod	Optional level-of-detail state.
selection	Optional explicit selection state.
metadata	Optional state metadata.

Value

An xgeo_state object.

Examples

```
# Matrix input is converted into a backend regular grid state.
xgeo_state(matrix(c(1, -1, 2, 0), nrow = 2))

# Long-tabular input preserves point ids and feature ids.
xgeo_state(
  data.frame(
    point_id = c("p1", "p1", "p2"),
    feature = c("f1", "f2", "f1"),
    x = c(0, 0, 1),
    y = c(0, 0, 1),
    value = c(1, -0.5, 0.75)
  )
)
```

Index