

Package: vrtstack (via r-universe)

October 15, 2026

Title Create GDAL Multidimensional VRTs from 2D Raster Collections

Version 0.1.0.9001

Description Stack 2D rasters (COGs, GeoTIFFs, etc.) along a new dimension, producing GDAL Multidimensional VRT files. This enables treating a collection of 2D rasters as a single multidimensional array, readable by GDAL's multidim API, xarray backends (gdx, xgdal), and other tools. Supports regular axes (start + increment), explicit values, and automatic extraction of dimension values from filenames using regex patterns with optional date/time parsing.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 4.1.0)

Imports gdalraster

Suggests vapour, testthat (>= 3.0.0)

Config/testthat/edition 3

URL <https://github.com/hypertidy/vrtstack>

BugReports <https://github.com/hypertidy/vrtstack/issues>

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libgdal-dev gdal-bin libgeos-dev libxml2-dev libzstd-dev

Repository <https://hypertidy.r-universe.dev>

Date/Publication 2026-08-26 04:31:08 UTC

RemoteUrl <https://github.com/hypertidy/vrtstack>

RemoteRef HEAD

RemoteSha 175198eff3641c88106c47003fe4a1f265c34bb5

Contents

gdal_mdimg_stack	2
mdimg_template_extend	4
vrt_depth_stack	5
vrt_regular_stack	6
vrt_time_stack	7
vrtstack	9
Index	11

gdal_mdimg_stack	<i>Create a Multidimensional VRT from 2D Rasters</i>
------------------	--

Description

Stack 2D rasters along a new dimension, producing a GDAL Multidimensional VRT. This enables treating a collection of 2D rasters (COGs, GeoTIFFs, etc.) as a single multidimensional array, readable by GDAL’s multidim API and xarray backends.

Usage

```
gdal_mdimg_stack(  
    files,  
    dim_name,  
    dim_values = NULL,  
    dim_start = NULL,  
    dim_increment = NULL,  
    dim_type = NULL,  
    dim_direction = NULL,  
    dim_unit = NULL,  
    time_origin = NULL,  
    time_unit = "days",  
    parse_format = NULL,  
    array_name = NULL,  
    output = NULL  
)
```

Arguments

files	Character vector of file paths (COGs, GeoTIFFs, /vsicurl/ URIs, etc.)
dim_name	Name for the new dimension (e.g., "time", "depth", "level")
dim_values	Dimension values. One of: • Numeric vector of values (length must match files) • A regex pattern with one capturing group to extract from filenames • A function taking filename (basename) and returning a numeric value • NULL if using dim_start + dim_increment

dim_start, dim_increment	For regular axes, specify start and increment instead of explicit values. Files are assumed to be in the correct order.
dim_type	Optional dimension type: "TEMPORAL", "VERTICAL", "PARAMETRIC"
dim_direction	Optional direction: "NORTH", "SOUTH", "EAST", "WEST", "UP", "DOWN", "FUTURE", "PAST"
dim_unit	Unit string for the dimension (e.g., "days since 1970-01-01", "cm")
time_origin	For temporal dimensions, origin date as "YYYY-MM-DD" or POSIXct. When set with parse_format, values are encoded as offsets from this origin.
time_unit	For temporal dimensions: "seconds", "minutes", "hours", "days"
parse_format	For date/time extraction, a <code>strptime()</code> format string (e.g., "%Y%m%d" for "20230115")
array_name	Name for the data array in the VRT. Default: derived from common filename prefix, or "data".
output	Output VRT path. If NULL, returns XML string without writing.

Details

The function generates a GDAL Multidimensional VRT that virtually stacks 2D rasters along a new dimension. Key features:

- **Regular axes:** When dimension values are evenly spaced (or specified via `dim_start/dim_increment`), the VRT uses `<RegularlySpacedValues>` for compact representation.
- **Auto-detection:** Even with explicit `dim_values`, regular spacing is detected automatically and converted to the compact form.
- **Filename parsing:** Extract dimension values from filenames using regex, with optional date/time parsing. Useful for time series like "temperature_20230101.tif".
- **Cloud-native:** Supports `/vsicurl/` URIs for remote files. HTTP(S) URLs are automatically prefixed with `/vsicurl/`.

The resulting VRT can be read by:

- GDAL's multidim API: `gdalmdiminfo`, `gdalmdimtranslate`
- Python xarray backends: `gdx`, `xgdal`
- Any tool supporting GDAL multidimensional rasters

Value

Path to VRT file (invisibly), or XML string if output is NULL.

Examples

```
## Not run:
# Stack soil layers by depth
gdal_mdimm_stack(
  files = c("soil_0cm.tif", "soil_30cm.tif", "soil_100cm.tif"),
  dim_name = "depth",
  dim_values = c(0, 30, 100),
```

```

    dim_type = "VERTICAL",
    dim_direction = "DOWN",
    dim_unit = "cm",
    output = "soil_stack.vrt"
)

# Time series with regex extraction
gdal_mdim_stack(
  files = list.files(pattern = "temp_\\d{8}\\\\.tif$"),
  dim_name = "time",
  dim_values = "(\\d{8})",
  parse_format = "%Y%m%d",
  dim_type = "TEMPORAL",
  time_origin = "1970-01-01",
  time_unit = "days",
  output = "temperature.vrt"
)

# Regular axis - monthly data
gdal_mdim_stack(
  files = sprintf("precip_%02d.tif", 1:12),
  dim_name = "month",
  dim_start = 1,
  dim_increment = 1,
  output = "monthly.vrt"
)

# Cloud-hosted COGs
gdal_mdim_stack(
  files = paste0(
    "https://storage.googleapis.com/solus100pub/claytotal_",
    c(0, 5, 15, 30, 60, 100, 150), "_cm_p.tif"
  ),
  dim_name = "depth",
  dim_values = c(0, 5, 15, 30, 60, 100, 150),
  dim_type = "VERTICAL",
  dim_unit = "cm",
  array_name = "claytotal",
  output = "solus_clay.vrt"
)

## End(Not run)

```

mdim_template_extend *Extend an mdim VRT recipe to a full file set*

Description

Pure XML-tree rewrite (no GDAL needed). Sets the concat dimension size, rewrites the concat coordinate array's values, and regenerates the <Source> list of every *data* array (an <Array> that

owns a <Source>). Per-array SourceArray path and slab geometry are read from the array's first source, never assumed from the array name.

Usage

```
mdim_template_extend(xml, files, concat_values = NULL, concat_dim = "time")
```

Arguments

xml	Template VRT as an XML string (or anything <code>xml2::read_xml()</code> takes).
files	Full character vector of sources.
concat_values	Values for the concat coordinate array, or NULL to keep the template's values (only valid when length matches files).
concat_dim	Concat dimension name (default "time"); if not found, the dimension whose size equals the template's per-array source count is used.

Value

An xml_document (serialise with `as.character()` / `xml2::write_xml()`).

vrt_depth_stack	<i>Stack Rasters by Depth/Level into Multidimensional VRT</i>
-----------------	---

Description

A convenience wrapper for `gdal_mdim_stack()` for vertical profile data (soil layers, atmospheric levels, ocean depths, etc.)

Usage

```
vrt_depth_stack(
  files,
  values,
  direction = "DOWN",
  unit = NULL,
  dim_name = NULL,
  array_name = NULL,
  output = NULL
)
```

Arguments

files	Character vector of file paths
values	Numeric vector of depth/level values, or regex pattern to extract
direction	Vertical direction: "DOWN" (depth increases), "UP" (elevation). Default: "DOWN"
unit	Unit string (e.g., "m", "cm", "hPa")

dim_name	Dimension name. Default: "depth" if direction is DOWN, "level" if UP.
array_name	Name for the data array
output	Output VRT path

Value

Path to VRT file (invisibly), or XML string if output is NULL.

Examples

```
## Not run:
# Soil layers at specific depths
vrt_depth_stack(
  files = c("soil_0cm.tif", "soil_30cm.tif", "soil_100cm.tif"),
  values = c(0, 30, 100),
  unit = "cm",
  output = "soil_profile.vrt"
)

# Extract depth from filename
vrt_depth_stack(
  files = list.files(pattern = "temp_\\d+m\\.tif$"),
  values = "(\\d+)m", # extracts "50" from "temp_50m.tif"
  unit = "m",
  output = "ocean_profile.vrt"
)

## End(Not run)
```

vrt_regular_stack	<i>Stack Rasters with Regular Spacing</i>
-------------------	---

Description

A convenience wrapper for [gdal_mdimg_stack\(\)](#) when files represent regularly spaced values (e.g., monthly data, hourly snapshots, uniform vertical levels).

Usage

```
vrt_regular_stack(
  files,
  dim_name,
  start,
  increment,
  unit = NULL,
  array_name = NULL,
  output = NULL
)
```

Arguments

<code>files</code>	Character vector of file paths, in order
<code>dim_name</code>	Name for the new dimension
<code>start</code>	Starting value
<code>increment</code>	Spacing between values
<code>unit</code>	Unit string
<code>array_name</code>	Name for the data array
<code>output</code>	Output VRT path

Value

Path to VRT file (invisibly), or XML string if output is NULL.

Examples

```
## Not run:
# 12 monthly files
vrt_regular_stack(
  files = sprintf("precip_%02d.tif", 1:12),
  dim_name = "month",
  start = 1,
  increment = 1,
  output = "monthly_precip.vrt"
)

# Pressure levels at 50 hPa intervals
vrt_regular_stack(
  files = level_files,
  dim_name = "pressure",
  start = 1000,
  increment = -50,
  unit = "hPa",
  output = "pressure_levels.vrt"
)

## End(Not run)
```

Description

A convenience wrapper for [gdal_mdimg_stack\(\)](#) tailored for time series data. Extracts dates from filenames and encodes them as numeric offsets from an origin.

Usage

```
vrt_time_stack(
  files,
  pattern = "(\\d{8})",
  format = "%Y%m%d",
  origin = "1970-01-01",
  unit = "days",
  array_name = NULL,
  output = NULL
)
```

Arguments

files	Character vector of file paths
pattern	Regex pattern with one capturing group for the date portion. Default: "(\\d{8})" matches 8-digit dates like 20230115.
format	strptime format for parsing. Default: "%Y%m%d"
origin	Time origin as "YYYY-MM-DD" string. Default: "1970-01-01"
unit	Time unit: "days", "hours", "minutes", "seconds". Default: "days"
array_name	Name for the data array
output	Output VRT path

Value

Path to VRT file (invisibly), or XML string if output is NULL.

Examples

```
## Not run:
# Daily temperature files: temp_20230101.tif, temp_20230102.tif, ...
vrt_time_stack(
  files = list.files(pattern = "temp_\\d{8}\\..tif$"),
  output = "temperature_timeseries.vrt"
)

# Custom pattern for ISO dates: data_2023-01-15.tif
vrt_time_stack(
  files = my_files,
  pattern = "(\\d{4}-\\d{2}-\\d{2})",
  format = "%Y-%m-%d",
  output = "timeseries.vrt"
)

## End(Not run)
```

vrtstack	<i>Stack rasters into a multidimensional VRT (two paths, one entry point)</i>
----------	---

Description

`vrtstack()` builds a GDAL multidimensional VRT that concatenates a set of sources along one axis. Two construction paths share a single concat-axis resolver:

Usage

```
vrtstack(
  files,
  concat = NULL,
  template = NULL,
  concat_dim = "time",
  n_template = 10,
  parse_format = NULL,
  origin = NULL,
  unit = "days",
  output = NULL,
  ...
)
```

Arguments

<code>files</code>	Character vector of source paths/URIs (full set).
<code>concat</code>	Concat-axis spec, resolved by <code>resolve_concat()</code> : a numeric/ character vector (length == <code>files</code>), a single regex with one capture group to pull from base-names, a function of basename, or NULL to use <code>concat_start/concat_increment</code> (from-scratch) or the template's own values (template-extend, when <code>concat</code> is NULL).
<code>template</code>	One of: NULL (from-scratch); a path to a .vrt; an XML string; or TRUE to auto-generate the recipe from <code>head(files, n_template)</code> via <code>gdal mdim mosaic</code> .
<code>concat_dim</code>	Name of the concatenation dimension. Default "time".
<code>n_template</code>	Files used to build the auto-generated recipe when <code>template = TRUE</code> . Default 10.
<code>parse_format, origin, unit</code>	Passed to <code>resolve_concat()</code> for date parsing.
<code>output</code>	Output VRT path, or NULL to return the XML string.
<code>...</code>	Passed to the from-scratch builder (<code>dim_type</code> , <code>dim_unit</code> , <code>array_name</code> , etc.) and ignored on the template path.

Details

- **from-scratch** (template = NULL) — the original GeoTIFF/COG path. Each 2D source contributes one slab of a *single* new 3D array. Sources use the classic <SourceBand> + <SourceTranspose> form.
- **template-extend** (template = <vrt> or template = TRUE) — for mdim-recognised sources (NetCDF/Zarr/HDF5) that may carry *several* arrays (OISST: anom/err/ice/sst). A small "first-n" mdim VRT is taken as the recipe, and the full file set is fleshed out by re-parameterising only the concat dimension and the per-array <Source> lists. Everything else (coordinate arrays, slab geometry, scale/offset/unit/attributes) is carried verbatim from the template.

The template is the GDAL-native analogue of the "recipe-not-payload" principle: GDAL's own `gdal mdim mosaic` writes the recipe for the first n files; we only template the axis that varies with the file set.

Value

Path to the VRT (invisibly) or the XML string if output is NULL.

Examples

```
#vrt <- vrtstack(<files>, concat = "\\d{8}"), parse_format = "%Y%m%d",
#origin = "1978-01-01", unit = "days",    # must match the template
#template = TRUE, concat_dim = "time")
```

Index

`as.character()`, 5

`gdal_mdimg_stack`, 2
`gdal_mdimg_stack()`, 5–7

`mdimg_template_extend`, 4

`resolve_concat()`, 9

`strptime()`, 3

`vrt_depth_stack`, 5
`vrt_regular_stack`, 6
`vrt_time_stack`, 7
`vrtstack`, 9

`xml2::read_xml()`, 5
`xml2::write_xml()`, 5