# Package: gdalraster.windows (via r-universe)

June 28, 2026

**Title** Windows GDAL Runtime and gdalraster Bootstrap Tools

**Version** 0.2.1

**Description** Installs and activates a self-contained GDAL runtime on Windows and provides helpers to install gdalraster from source against that GDAL instead of the default Rtools GDAL.

**License** MIT + file LICENSE

**URL** https://github.com/jimbrig/gdalraster.windows, https://docs.jimbrig.com/gdalraster.windows/

**BugReports** https://github.com/jimbrig/gdalraster.windows/issues

**Imports** cli, httr2, rlang, stats, tools, utils, withr

**Suggests** gdalraster, knitr, rmarkdown, testthat (>= 3.0.0)

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Encoding** UTF-8

**Language** en-US

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 8.0.0

**Config/pak/sysreqs** libssl-dev

**Repository** https://jimbrig.r-universe.dev

**Date/Publication** 2026-06-28 14:45:52 UTC

**RemoteUrl** https://github.com/jimbrig/gdalraster.windows

**RemoteRef** HEAD

**RemoteSha** e72e735a45ff1aa351d8481953edea5b6c66bdcf

# Contents



---

`activate_gdal_runtime` *Activate GDAL runtime for current R session*

---

## Description

Prepends runtime paths, sets GDAL/PROJ env vars, and preloads GDAL DLL.

## Usage

```
activate_gdal_runtime(
  gdal_home = default_gdal_home(),
  preload = TRUE,
  quiet = FALSE
)
```

## Arguments

| | |
|---|---|
| `gdal_home` | GDAL home directory. |
| `preload` | Whether to preload `libgdal-*.dll`. |
| `quiet` | Suppress informational CLI output. |

## Details

When the runtime bundle contains a `python/` directory (pure-python `osgeo_utils` package from GDAL's `gdal-utils` distribution), it is prepended to `PYTHONPATH` so GDAL algorithms that embed Python at runtime (e.g. `gdal driver gpkg validate`) can import it. This is session-scoped and does not modify machine or user environment variables.

## Value

Invisibly returns a list with configured paths.

## add_gdal_rprofile_hook *Add or update an .Rprofile hook for bundled GDAL*

### Description

Writes a managed hook block into an `.Rprofile` file. The block loads the bundled GDAL DLL before package attach and prepends the custom gdalraster library path.

### Usage

```
add_gdal_rprofile_hook(
  rprofile = "~/.Rprofile",
  gdal_home = default_gdal_home(),
  lib = default_gdalraster_lib(),
  dry_run = FALSE
)
```

### Arguments

| | |
|---|---|
| `rprofile` | Target `.Rprofile` path. |
| `gdal_home` | GDAL home directory. |
| `lib` | Library path containing the custom gdalraster install. |
| `dry_run` | If `TRUE`, return the updated file contents without writing. |

### Value

Invisibly returns the updated `.Rprofile` text.

## configure_gdal_home *Configure GDAL home for current session*

### Description

Sets GDAL home for this session using either an R option or environment variable. This does not write to user profile files.

### Usage

```
configure_gdal_home(path, mode = c("option", "env"))
```

### Arguments

| | |
|---|---|
| `path` | GDAL home directory path. |
| `mode` | Either `"option"` or `"env"`. |

### Value

Invisibly returns the normalized GDAL home path.

---

## gdal_home *Resolve active GDAL home path*

---

### Description

Returns the currently configured GDAL runtime home used by this package.

### Usage

```
gdal_home()
```

### Details

Resolution order:

1. `options(gdalraster.windows.gdal_home = "...")`
2. `GDALRASTER_WINDOWS_GDAL_HOME` environment variable
3. package-managed user data directory (`tools::R_user_dir()`)

### Value

A single string path.

---

## gdal_rprofile_snippet *Build an .Rprofile hook snippet for bundled GDAL*

---

### Description

Returns R code that loads the bundled GDAL DLL before attaching `gdalraster`, and prepends the custom `lib` path so `library(gdalraster)` resolves to the source build installed by `install_gdalraster()`.

### Usage

```
gdal_rprofile_snippet(
  gdal_home = default_gdal_home(),
  lib = default_gdalraster_lib()
)
```

### Arguments

| | |
|---|---|
| `gdal_home` | GDAL home directory. |
| `lib` | Library path containing the custom gdalraster install. |

### Value

A single string containing R code.

---

`install_gdal_runtime` *Install precompiled GDAL runtime*

---

### Description

Installs the GDAL runtime into `gdal_home` from one of:

### Usage

```
install_gdal_runtime(
  repo = "jimbrig/gdalraster.windows",
  tag = "latest",
  asset_pattern = "gdal-(bundle|ucrt64)-.*\\.zip$",
  gdal_home = default_gdal_home(),
  overwrite = FALSE,
  local_zip = NULL,
  fallback_zip = NULL
)
```

### Arguments

| | |
|---|---|
| `repo` | GitHub repo slug, e.g. `"jimbrig/gdalraster.windows"`. |
| `tag` | Release tag or `"latest"`. |
| `asset_pattern` | Regex used to select the release asset. |
| `gdal_home` | Destination GDAL home directory. |
| `overwrite` | Whether to replace existing `gdal_home`. |
| `local_zip` | Optional local GDAL runtime zip to install directly. |
| `fallback_zip` | Optional fallback zip path used when release download fails. When `NULL` (default), a vendored zip at `inst/extdata/gdal-ucrt64-fallback.zip` is used if present; none ships with the package, so by default no fallback is attempted. |

### Details

- `local_zip` (highest precedence),
- GitHub release asset lookup/download,
- `fallback_zip` when release lookup/download fails.

The selected zip must contain a GDAL root with `bin/libgdal-*.dll`.

### Value

Invisibly returns installed GDAL home path.

### Offline / air-gapped installation

On machines without network access, download the release asset manually from https://github.com/jimbrig/gdalraster.windows/releases, transfer it to the target machine, and install directly:

```
gdalraster.windows::install_gdal_runtime(
  local_zip = "C:/Downloads/gdal-ucrt64-v3.13.1-windows-x64.zip"
)
```

Note that no fallback zip is shipped with the package (a full runtime bundle is too large to vendor), so the fallback path only applies when you provide a `fallback_zip` yourself.

---

`install_gdalraster` *Install gdalraster from source against bundled GDAL*

---

## Description

Downloads or uses a local gdalraster source tarball and installs it from source into a dedicated library path (default) so existing user libraries are not overwritten.

## Usage

```
install_gdalraster(
  gdal_home = default_gdal_home(),
  lib = default_gdalraster_lib(),
  source_tarball = NULL,
  repo = "firelab/gdalraster",
  ref = "HEAD",
  upgrade = FALSE,
  repos = getOption("repos")
)
```

## Arguments

| | |
|---|---|
| `gdal_home` | GDAL home directory used for compile/link flags. |
| `lib` | Destination library path for installing gdalraster. |
| `source_tarball` | Optional local path to `gdalraster_*.tar.gz`. |
| `repo` | Source GitHub repo slug for gdalraster. |
| `ref` | Git ref (branch, tag, commit) used when downloading from GitHub. |
| `upgrade` | Whether to allow dependency upgrades during install. |
| `repos` | CRAN-like repositories passed to `utils::install.packages()`. |

## Value

Invisibly returns installed library path.

## load_gdal_dll *Load GDAL DLL from runtime bundle*

### Description

Convenience wrapper over `activate_gdal_runtime()` that ensures the GDAL runtime is activated and the main GDAL DLL is preloaded in the current session.

### Usage

```
load_gdal_dll(gdal_home = default_gdal_home(), quiet = FALSE)
```

### Arguments

| | |
|---|---|
| gdal_home | GDAL home directory. |
| quiet | Suppress informational CLI output. |

### Value

Invisibly returns activation metadata.

## load_gdalraster *Load gdalraster using bundled GDAL runtime*

### Description

Activates bundled GDAL runtime, prepends `lib` to `.libPaths()`, and attaches gdalraster for use in the current R session.

### Usage

```
load_gdalraster(
  lib = default_gdalraster_lib(),
  gdal_home = default_gdal_home(),
  quiet = FALSE
)
```

### Arguments

| | |
|---|---|
| lib | Library path containing the gdalraster source install. |
| gdal_home | GDAL home directory. |
| quiet | Suppress informational CLI output. |

### Value

Invisibly returns TRUE if gdalraster was attached.

verify_gdalraster_runtime *Verify gdalraster algorithm API availability*

## Description

Attempts to load gdalraster and checks the global algorithm registry.

## Usage

```
verify_gdalraster_runtime(
  lib.loc = NULL,
  activate_runtime = TRUE,
  gdal_home = default_gdal_home(),
  quiet = FALSE
)
```

## Arguments

| | |
|---|---|
| lib.loc | Optional library location used for loading gdalraster. |
| activate_runtime | Whether to run activate_gdal_runtime() first. |
| gdal_home | GDAL home used when activate_runtime = TRUE. |
| quiet | If TRUE, suppress sitrep CLI output. |

## Value

TRUE when algorithm API is available, otherwise FALSE.

# Index