Package: sds (via r-universe)

June 2, 2026

Title Spatial Data Sources

Version 0.0.1.9015

Description Functions for descriptions of spatial data sources that can be read directly with or without download. Most sources are readable by 'GDAL' software. There are functions for imagery, elevation, and a variety of easily accessible data streams including image tile servers, links to online files, and geographic data services.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

URL https://github.com/hypertidy/sds

BugReports https://github.com/hypertidy/sds/issues

Imports countrycode, tibble

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Depends R (>= 2.10)

LazyData true

Config/roxygen2/version 8.0.0

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

Date/Publication 2026-06-02 05:45:25 UTC

RemoteUrl https://github.com/hypertidy/sds

RemoteRef HEAD

RemoteSha 88adc9225a798f8d23a34f6ed74495644493c721

Contents

`addrock` *Medium resolution vector polygons of Antarctic rock outcrop - VERSION 7.3*

Description

https://data.bas.ac.uk/full-record.php?id=GB/NERC/BAS/PDC/01409

Usage

```
addrock()
```

`CGAZ` *the geoBoundaries countries*

Description

CGAZ() returns the DSN, a shapefile of geo boundaries, CGAZ_sql() returns SQL suitable for use with GDAL, for the names or codes of countries.

Usage

```
CGAZ(old = FALSE)

CGAZ_sql(codes)
```

Arguments

old	logical, return the old slow zipped shapefile or the new Parquet copy
codes	a list of iso3 country codes, or country names (this is a bit sketchy)

Examples

```
CGAZ_sql(c("Australia", "New Zealand"))
CGAZ_sql(c("AUS", "NZL"))
## do something like gdal_raster_data(gebco(), target_res = 1,
##                                         options = c("-crop_to_cutline",
##                                         "-cutline", CGAZ(),
##                                    "-csql", CGAZ_sql(c("Australia", "New Zealand")) ))
```

---

dea_250m_dem *DEA 250m dem*

Description

DEA 250m dem

Usage

```
dea_250m_dem(vsi = TRUE)
```

---

dsn-sources *Imagery online sources*

Description

Raster and imagery online

Usage

```
wms_arcgis_mapserver_ESRI.WorldImagery_tms()

wms_bluemarble_s3_tms()

wms_googlehybrid_tms()

wms_virtualearth()

wms_ESA_worldcover_2020_tms()

wms_mapbox_satellite()
```

```
wms_amazon_elevation()

wms_mapbox_terrain()

wms_openstreetmap_tms()

wms_googleterrainstreets_tms()

wms_virtualearth_street()

wms_arcgis_mapserver_tms()

nasadem()

cop90()

cop30()

srtm15()
```

`esacci_chlor_a` *ESACCI chlorophyll-a in monthly*

Description

ESACCI chlorophyll-a in monthly

Usage

```
esacci_chlor_a(time.resolution = "monthly")
```

Arguments

`time.resolution`
: monthly only for now

Value

data frame with date,source

Examples

```
dsn <- esacci_chlor_a()
```

gebco *GEBCO source dsn*

Description

A data source name to the GEBCO elevation 'COG' GeoTIFF.

Usage

```
gebco(vsi = TRUE)

gebco25(vsi = TRUE)

gebco24(vsi = TRUE)

gebco21(vsi = TRUE)

gebco23_bedrock(vsi = TRUE)

gebco23(vsi = TRUE)

gebco22(vsi = TRUE)

gebco19(vsi = TRUE)
```

Arguments

vsi	include the 'vsicurl' prefix (TRUE is default)

Details

GEBCO 2023 and 2022 is created and hosted by Philippe Massicotte.

GEBCO 2019 and 2021 created and hosted by the Australian Antarctic Division.

See note about which forms of the bedrock vs ice surface are available. Generally we use the ice surface form, because that is what encountered while navigating the surface of the Earth. But, the bedrock is of course also of interest. "If the data sets are used in a presentation or publication then we ask that you acknowledge the source. This should be of the form (see references)."

Value

character string, URL to online GeoTIFF

warning

please note that `gebco21()`, `gebco19()`, `gebco24()` and `gebco25()` return the *ice surface* form, while `gebco22()` returns the bedrock form. With `gebco23()` and `gebco23_bedrock()` these are now both avaiable, again thanks to Philippe Massicotte.

References

https://www.gebco.net/data-products/gridded-bathymetry-data 'GEBCO Compilation Group (2025) GEBCO 2025 Grid (doi:10.5285/37c52e96-24ea-67ce-e063-7086abc05f29)'

Examples

```
gebco()
```

`ghrsst` *GHRSST files, GeoTIFFs on source.coop*

Description

GHRSST files, GeoTIFFs on source.coop

Usage

```
ghrsst(vsi = TRUE)
```

Value

dataframe of source,date

Examples

```
files <- ghrsst()
tail(files$source)
```

`ibcso` *IBCOS source dsn*

Description

A data source name to the IBCSO elevation 'COG' GeoTIFF.

Usage

```
ibcso(vsi = TRUE, chart = FALSE)
```

Arguments

`vsi`	include the 'vsicurl' prefix (`TRUE` is default)
`chart`	the image or the data? set to TRUE for image (it's a PDF)

Details

Currently at v2.

Value

character string, URL to online raster

Examples

```
ibcso()
```

`list_address` *Title*

Description

Title

Usage

```
list_address(address)
```

Arguments

address

Value

string URI for List service

Examples

```
x <- list_address(c(2862, "LYELL",  "HAYES"))
#vals <- jsonlite::fromJSON(readr::read_file ( x), simplifyVector = F)$features[[1]]$attributes
#plot(v <- vect(list_parcel(vals$PID)))
#Sys.setenv("GDAL_DISABLE_READDIR_ON_OPEN"="TRUE")
#dem <- rast("/vsicurl/https://s3.us-west-2.amazonaws.com/us-west-2.opendata.source.coop/alexgleith/tasmania-de
#plotRGB(project(rast(ortho), rast(v, res = .1), by_util = TRUE))
#plot(v, add = T)
```

list_parcel_shp *Cadastral parcel source*

Description

Cadastral parcel source

Usage

```
list_parcel_shp()
```

Value

string URI for List shapefile

Examples

```
list_parcel_shp()  ## read with terra::vect( ) or new(gdalraster::GDALVector, )
```

mapterhorn_elevation *Mapterhorn PMTiles raster elevation*

Description

Rendered calc VRT that unpacks planet.pmtiles from Mapterhorn, a large collection of regional elevation sources encoded in Terrarium Terrain RGB '(R*256 + G + B/256) - 32768'

Usage

```
mapterhorn_elevation(...)
```

Arguments

...	unused

Value

VRT text string, useable by GDAL >= 3.14 with muparser support

Examples

```
writeLines(mapterhorn_elevation())
```

mpc *Microsoft Planetary Computer*

Description

Function `mpc()` will return a full data source name for a Spatio-Temporal Asset Catalog ('STAC').

Usage

```
mpc(
  collection = "sentinel-2-l2a",
  bbox = "146.5,-43.2,147.5,-42.2",
  datetime = Sys.Date() + c(-5, 1),
  asset = "",
  stacit = TRUE
)
```

Arguments

`collection`	name of the collection e.g. "sentinel-2-l2a", "naip", or ""landsat-c2-l2"
`bbox`	a string in the form 'xmin,ymin,xmax,ymax' where x,y are longitude and latitude values (OR a numeric extent xmin,xmax,ymin,ymax)
`datetime`	a datetime of 1 or 2 values to give a range in time, if only one is given it is treated as an open interval to the present
`asset`	name of the asset of interest, e.g. for "sentinel-2-l2a" there is "visual", "AOT", "B04" etc.
`stacit`	logical, treat this as a 'GDAL STACIT' source, or just a generic MPC query, 'TRUE' by default

Details

Each argument has a default, use them to set the collection, datetime range, bounding box, and asset.

If no asset is specfied the description is a complex source composed of multiple subdatasets ('GDAL' terminology). In a 'STAC' context "asset" and "subdataset" are synonomous for 'GDAL'.

Value

a string, a data source name for 'GDAL'

Examples

```
mpc()

## we can be more general than GDAL, and say do a query to get Parquet for MS Buildings
```

```
mpc(collection = "ms-buildings", stacit = FALSE, bbox = "140,-45,145,-30",
                                 datetime = as.Date("2000-01-01"))
## read that url with jsonlite and investigate $features$assets$data$href
## but it's all abfs:// azure special links
```

`mursst` *MURSST (GHRSST) sst Zarr source*

Description

This is used a lot of noisy fanfare about why everyone must move to Zarr in the cloud. It's really big, daily netcdf blended/observation/model data on a 36000x18000 grid, a regular grid in -180, 180, -90, 90.

Usage

```
mursst_zarr(band = 0)

mursst_time(time = NULL)
```

Arguments

`band`	a band number (defaults to 0, which is 2002-06-01)
`time`	a time value to pick, see Details

Details

This zarr and every other one I've found is unuseably out of date. It seems like this source doesn't go past "2020-01-21" or band 6443, don't know why that is.

Value

a string, a dsn for Zarr MURSST

Examples

```
mursst()
mursst_time("2019-10-08")
```

nsidc_seaice *Data sources for NOAA NSIDC GeoTIFF images for sea ice concentration and extent.*

Description

Data is colour-palette GeoTIFF in polar stereographic raster format.

Usage

```
nsidc_seaice(
  date,
  hemisphere = c("south", "north"),
  temporal = c("daily", "monthly"),
  varname = c("concentration", "extent"),
  vsi = TRUE
)
```

Arguments

date	date in POSIXct or Date format, see Details
hemisphere	hemisphere of data source, north or south
temporal	temporal resolution of data source, daily or monthly
varname	variable name to obtain, concentration or extent (note this is a colour image)
vsi	prepend DSN with "/vsicurl/" or not

Details

The availability of the date is not checked currently, the minimum date is 1978-10-26. Please note that these data were only every two days from date, until a later time (FIXME get these details).

Value

character string, GDAL readable DSN to a colour palette GeoTIFF (see args 'vsi' and others for options)

Examples

```
nsidc_seaice("2023-06-27")
#readBin(con <- url(nsidc_seaice("2023-06-27", vsi = FALSE), open = "rb"), "raw"); close(con)
```

rema *REMA reference elevation model of Antarctica*

Description

This is a single description string for all of the 2m REMA. The VRT is crafted with efficient overviews so is much more performant with the warper API than other existing descriptions.

Usage

```
rema()

rema_v2()
```

Details

See rema-ovr for examples.

Value

character string, GDAL-readable raster data source name

seaice_cdr *Sea ice CDR (climate data record) url*

Description

The first subdataset 'cdr_seaice_conc' is the one you want, but also included are "cdr_seaice_conc_interp_spatial_flag", "cdr_seaice_conc_interp_temporal_flag", "cdr_seaice_conc_qa_flag", "cdr_seaice_conc_stdev", "raw_bt_seaice_conc", "raw_nt_seaice_conc" and "surface_type_mask"

Usage

```
seaice_cdr(date, hemisphere = c("south", "north"), vsi = TRUE)
```

Arguments

date	date or or string YYYY-mm-dd
hemisphere	north or south
vsi	in GDAL url form

Details

I've had mixed success setting subdataset and having these return with the correct orientation.

Value

string with path to CDR sea ice netcdf file

Examples

```
seaice_cdr()
```

`sentinel2_wms` *Sentinel 2 WMS*

Description

To use this you must have your own "INSTANCE_ID", set this in env var "SENTINELHUB_INSTANCE_ID".

Usage

```
sentinel2_wms(
  layer = c("TRUE-COLOR-S2L2A", "NDVI", "FALSE-COLOR", "FALSE-COLOR-URBAN",
   "AGRICULTURE", "BATHYMETRIC", "GEOLOGY", "MOISTURE-INDEX", "SWIR", "NATURAL-COLOR"),
  datetime = NA
)
```

Arguments

`layer`	see layer options in the argument, default is "TRUE-COLOR-S2L2A"
`datetime`	a valid datetime or NA for "latest"

Value

a string to a WMS

Examples

```
sentinel2_wms()
```

`stacit` *STAC Query URL Generator*

Description

Generate STAC API search URLs from an extent and date specification. Provides a simple interface for constructing valid STAC queries without manually assembling URL parameters.

Usage

```
stacit(
  extent,
  date = "",
  collections = "sentinel-2-c1-l2a",
  provider = c("https://earth-search.aws.element84.com/v1/search",
    "https://planetarycomputer.microsoft.com/api/stac/v1/search"),
  gdal_stacit = FALSE,
  limit = 300
)
```

Arguments

`extent`	numeric vector (xmin, xmax, ymin, ymax) in longlat, or character MGRS 100km grid code
`date`	character, Date, or POSIXt specifying the temporal query. Length 1 for an implied interval or length 2 for an explicit range. Default `""` uses today's date.
`collections`	character; STAC collection identifier(s) to search. Default is `"sentinel-2-c1-l2a"` (Sentinel-2 Cloud-Optimized GeoTIFFs).
`provider`	character; base URL of the STAC API search endpoint. Default uses Element 84's Earth Search. Also supports Microsoft Planetary Computer.
`gdal_stacit`	logical; if `TRUE`, return a GDAL STACIT driver DSN string instead of a URL. Default `FALSE`.
`limit`	integer; maximum number of items to return per query. Default 300.

Value

Character vector of STAC search URL(s). Returns multiple URLs when the extent crosses the anti-meridian.

Extent specification

The extent can be specified as either:

- A numeric vector of length 4 giving xmin, xmax, ymin, ymax in longitude/latitude (EPSG:4326)
- A character string giving a 100km MGRS grid square code (e.g. "43DFD")

Extents that cross the anti-meridian (i.e. where xmin < -180 or xmax > 180) are automatically split into two queries, one for each side of the ±180° boundary. This handles the case where a projected extent (e.g. UTM) spans the anti-meridian but STAC APIs truncate bounding boxes at -180/180.

Date specification

The date parameter accepts flexible input formats that are expanded to ISO 8601 datetime intervals:

- Year only: `"2024"` becomes `2024-01-01T00:00:00Z/2024-12-31T23:59:59Z`
- Year-month: `"2024-06"` expands to the full month interval
- Full date: `"2024-06-15"` expands to that single day
- Date range: `c("2024-01-01", "2024-01-31")` for explicit start/end
- Date or POSIXt objects are also accepted

References

STAC API specification: https://github.com/radiantearth/stac-api-spec

GDAL STACIT driver: https://gdal.org/en/stable/drivers/raster/stacit.html

See Also

`mpc()` for Microsoft Planetary Computer catalog access; `sentinel2_wms()` for Sentinel-2 WMS layer construction

Examples

```
# Simple bounding box query for today
stacit(c(140, 145, -43, -40))

# MGRS grid square for a specific month
stacit("43DFD", date = "2025-01")

# Date range query
stacit(c(140, 145, -43, -40), date = c("2024-06-01", "2024-06-30"))

# Anti-meridian crossing returns two queries
stacit(c(179, 181, -16, -15), date = "2024-03")

# Use Planetary Computer instead of Earth Search
stacit(c(140, 145, -43, -40),
       provider = "https://planetarycomputer.microsoft.com/api/stac/v1/search")

# Query Landsat collection
stacit(c(140, 145, -43, -40), collections = "landsat-c2-l2", date = "2024")
```

tas_dem *Tasmania DEM (2m)*

Description

by MRT 2021

Usage

```
tas_dem(vsicurl = TRUE)
```

Arguments

vsicurl	prefix with vsicurl or not

Value

data source name for Tasmania 2m DEM

Examples

```
tas_dem()
```

usgs_seamless *USGS seamless DEM*

Description

USGS seamless DEM

Usage

```
usgs_seamless(vsicurl = TRUE)
```

Arguments

vsicurl	if TRUE prefix /vsicurl

Examples

```
usgs_seamless()
```

Index