Package 'ceramic'

February 27, 2024

Title Download Online Imagery Tiles

Version 0.9.5

Description Download imagery tiles to a standard cache and load the data into raster objects. Facilities for 'AWS' terrain <https://registry.opendata.aws/terrain-tiles/> terrain and 'Mapbox' <https://www.mapbox.com/> servers are provided.

Depends R (>= 3.5.0), terra

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.0

Imports curl, dplyr, fs (>= 1.3.0), glue, graphics, purrr, rappdirs, rlang, sp, slippymath (>= 0.3.0), stats, tibble, utils, crsmeta, vapour, stringr, wk

Suggests covr, spelling, testthat (>= 3.0.0)

URL https://hypertidy.github.io/ceramic/

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

Language en-US

LazyData true

Config/testthat/edition 3

NeedsCompilation no

Author Michael Sumner [aut, cre] (<https://orcid.org/0000-0002-2471-7511>), Miles McBain [ctb] (<https://orcid.org/0000-0003-2865-2548>), Ben Raymond [ctb] (regex wizardry)

Maintainer Michael Sumner <mdsumner@gmail.com>

Repository CRAN

Date/Publication 2024-02-27 06:30:02 UTC

R topics documented:

`cc_location` *Obtain tiled imagery by location query*

Description

Obtain imagery or elevation data by location query. The first argument `loc` may be a spatial object (sp, raster, sf) or a 2-column matrix with a single longitude and latitude value. Use `buffer` to define a width and height to pad around the raw longitude and latitude in metres. If `loc` has an extent, then `buffer` is ignored.

Usage

```
cc_location(
  loc = NULL,
  buffer = 5000,
  type = "mapbox.satellite",
  ...,
  zoom = NULL,
  max_tiles = NULL,
  debug = FALSE,
  dimension = NULL
)

cc_macquarie(
  loc = c(158.93835, -54.49871),
  buffer = 5000,
  type = "mapbox.satellite",
  ...,
  zoom = NULL,
  max_tiles = NULL,
  debug = FALSE,
  dimension = NULL
```

```
)

cc_davis(
  loc = c(77 + 58/60 + 3/3600, -(68 + 34/60 + 36/3600)),
  buffer = 5000,
  type = "mapbox.satellite",
  ...,
  zoom = NULL,
  max_tiles = NULL,
  debug = FALSE,
  dimension = NULL
)

cc_mawson(
  loc = c(62 + 52/60 + 27/3600, -(67 + 36/60 + 12/3600)),
  buffer = 5000,
  type = "mapbox.satellite",
  ...,
  zoom = NULL,
  max_tiles = NULL,
  debug = FALSE,
  dimension = NULL
)

cc_casey(
  loc = cbind(110 + 31/60 + 36/3600, -(66 + 16/60 + 57/3600)),
  buffer = 5000,
  type = "mapbox.satellite",
  ...,
  zoom = NULL,
  max_tiles = NULL,
  debug = FALSE,
  dimension = NULL
)

cc_heard(
  loc = c(73 + 30/60 + 30/3600, -(53 + 0 + 0/3600)),
  buffer = 5000,
  type = "mapbox.satellite",
  ...,
  zoom = NULL,
  max_tiles = NULL,
  debug = FALSE,
  dimension = NULL
)

cc_kingston(
  loc = c(147.2901, -42.98682),
```

```
  buffer = 5000,
  type = "mapbox.satellite",
  ...,
  zoom = NULL,
  max_tiles = NULL,
  debug = FALSE,
  dimension = NULL
)

cc_elevation(
  loc = NULL,
  buffer = 5000,
  type = NULL,
  ...,
  zoom = NULL,
  max_tiles = NULL,
  debug = FALSE,
  dimension = NULL
)
```

Arguments

loc	a longitude, latitude pair of coordinates, or a spatial object
buffer	with in metres to extend around the location, ignored if 'loc' is a spatial object with extent
type	character string of provider imagery type (see Details)
...	deprecated arguments that *used_to_be* passed to internal function now ignored since v 0.8.0 (see `get_tiles()`)
zoom	deprecated (use `dimension`)
max_tiles	deprecated
debug	deprecated
dimension	one or two numbers, used to determine the number of pixels width, height - set one to zero to let GDAL figure it out, or leave as NULL to get something suitable

Details

`cc_elevation` does extra work to unpack the DEM tiles from the RGB format.

Available types are 'elevation-tiles-prod' for AWS elevation tiles, and 'mapbox.satellite', and 'mapbox.terrain-rgb', 'tasmap' or one of 'tasmap_street' (TTSA), 'tasmap_aerialphoto2020', 'tasmap_aerialphoto2021', 'tasmap_aerialphoto2022', 'tasmap_aerialphoto2023', 'tasmap_esgismapbookpublic', 'tasmap_hillshadegrey', 'tasmap_hillshade', 'tasmap_orthophoto', 'tasmap_simplebasemap', 'tasmap_tasmap100k', 'tasmap_tasmap250k', 'tasmap_tasmap25k', 'tasmap_tasmap500k', 'tasmap_tasmapraster', 'tasmap_topographicgrayscale', 'tasmap_topographic'.

Note that arguments `max_tiles` and `zoom` are mutually exclusive. One or both must be `NULL`. If both are NULL then `max_tiles = 16L`.

Value

A terra::rast() object, either with three layers (Red, Green, Blue) or with a single layer in the case of cc_elevation().

Examples

```
if (!is.null(get_api_key())) {

 img <- cc_location(cbind(147, -42), buffer = 1e5)

 ## this source does not need the Mapbox API, but we won't run the example unless it's set
 dem <- cc_kingston(buffer = 1e4, type = "elevation-tiles-prod")
 terra::plot(dem, col = grey(seq(0, 1, length = 94)))

 ## Mapbox imagery
 ## Not run:
 im <- cc_macquarie()
 plotRGB(im)

## End(Not run)
 }
```

---

ceramic_cache *Ceramic file cache*

Description

File system location where ceramic stores its cache.

Usage

```
ceramic_cache(force = FALSE)
```

Arguments

force	set to TRUE to create the location without asking the user

Details

If allowed, the cache will be created at file.path(rappdirs::user_cache_dir(), ".ceramic"), which corresponds to '~/.cache/.ceramic' for a given user.

If the file cache location does not exist, the user will be asked in interactive mode for permission. For non-interactive mode use the force argument.

It is not currently possible to customize the cache location. To clear the cache (completely) see clear_ceramic_cache().

Value

A character vector, the file path location of the cache.

Examples

```
if (interactive()) {
 ceramic_cache()
}
```

`ceramic_tiles` *Tile files*

Description

Find existing files in the cache. Various options can be controlled, this is liable to change pending generalization across providers.

Usage

```
ceramic_tiles(
  zoom = NULL,
  type = "mapbox.satellite",
  source = "api.mapbox.com",
  glob = NULL,
  regexp = NULL
)
```

Arguments

`zoom`	zoom level
`type`	imagery type
`source`	imagery source
`glob`	see `fs::dir_ls`
`regexp`	see `fs::dir_ls`

Value

A data frame of tile file paths with tile index, zoom, type, version, source and spatial extent.

Examples

```
if (interactive() && !is.null(get_api_key())) {
 tiles <- ceramic_tiles(zoom = 0)
}
```

cities *Cities locations*

Description

Dataset from package maps.

Details

Data frame with columns "name" "country.etc" "pop" "lat" "long" "capital".

clear_ceramic_cache *Clear ceramic cache*

Description

Delete all downloaded files in the `ceramic_cache()`.

Usage

```
clear_ceramic_cache(clobber = FALSE, ...)
```

Arguments

`clobber`	set to `TRUE` to avoid checks and delete files
`...`	reserved for future arguments, currently ignored

Value

This function is called for its side effect, but also returns the file paths as a character vector whether deleted or not, or NULL if the user cancels.

`get-tiles-constrained` *Get tiles with specific constraints*

Description

Get tiles by zoom, by overall dimension, or by buffer on a single point.

Usage

```
get_tiles_zoom(x, zoom = 0, ..., format = "png")

get_tiles_dim(x, dim = c(512, 512), ..., format = "png")

get_tiles_buffer(x, buffer = NULL, ..., max_tiles = 9, format = "png")
```

Arguments

`x`	a spatial object with an extent
`zoom`	desired zoom for tiles, use with caution - cannot be unset in `get_tiles_zoom`
`...`	passed to `get_tiles()`
`format`	defaults to "png", also available is "jpg"
`dim`	for `get_tiles_dim` the overall maximum dimensions of the image (padded out to tile size of 256x256)
`buffer`	width in metres to extend around the location, ignored if 'x' is a spatial object with extent
`max_tiles`	maximum number of tiles - if `NULL` is set by zoom constraints

Details

Each function expects an extent in longitude latitude or a spatial object with extent as the first argument.

`get_tiles_zoom()` requires a zoom value, defaulting to 0

`get_tiles_dim()` requires a dim value, default to `c(512, 512)`, a set of 4 tiles

`get_tiles_buffer()` requires a single location (longitude, latitude) and a buffer in metres

Value

A list with files downloaded in character vector, a data frame of the tile indices, the zoom level used and the extent in xmin,xmax,ymin,ymax form.

See Also

get_tiles

Examples

```
if (!is.null(get_api_key())) {
 ex <- ext(146, 147, -43, -42)
 tile_infoz <- get_tiles_zoom(ex,  zoom = 1)

 tile_infod <- get_tiles_dim(ex,  dim = c(256, 256))

 tile_infob <- get_tiles_buffer(cbind(146.5, -42.5), buffer = 5000)
}
```

`get_api_key` *Get API key for Mapbox service*

Description

Mapbox tile providers require an API key. Other providers may not need a key and so this is ignored.

Usage

```
get_api_key(api = "mapbox", ..., silent = FALSE)
```

Arguments

`api`	character string denoting which service ("mapbox" only)
`...`	currently ignored
`silent`	run in completely silent mode, default is to provide a warning

Details

The mapdeck package has a more comprehensive tool for setting the Mapbox API key, if this is in use ceramic will find it first and use it.

To set your Mapbox API key obtain a key from https://account.mapbox.com/access-tokens/

```
1) Run this to set for the session 'Sys.setenv(MAPBOX_API_KEY=<yourkey>)'

OR,

2) To set permanently store 'MAPBOX_API_KEY=<yourkey>' in '~/.Renviron'.
```

There is a fairly liberal allowance for the actual name of the environment variable, any of 'MAPBOX_API_KEY', 'MAPBOX_API_TOKEN', 'MAPBOX_KEY', 'MAPBOX_TOKEN', or 'MAPBOX' will work (and they are sought in that order).

If no key is available, `NULL` is returned, with a warning.

Value

The stored API key value, see Details.

Examples

```
get_api_key()
```

`get_tiles` *Download imagery tiles*

Description

Obtain imagery or elevation tiles by location query. The first argument `loc` may be a spatial object (sp, raster, sf) or a 2-column matrix with a single longitude and latitude value. Use `buffer` to define a width and height to pad around the raw longitude and latitude in metres. If `loc` has an extent, then `buffer` is ignored.

Usage

```
read_tiles(
  x,
  buffer,
  type = "mapbox.satellite",
  crop_to_buffer = TRUE,
  format = NULL,
  ...,
  zoom = NULL,
  debug = FALSE,
  max_tiles = NULL,
  base_url = NULL,
  verbose = TRUE,
  filename = ""
)

get_tiles(
  x,
  buffer,
  type = "mapbox.satellite",
  crop_to_buffer = TRUE,
  format = NULL,
  ...,
  zoom = NULL,
  debug = FALSE,
  max_tiles = NULL,
  base_url = NULL,
  verbose = TRUE
)
```

Arguments

x	a longitude, latitude pair of coordinates, or a spatial object
buffer	width in metres to extend around the location, ignored if 'x' is a spatial object with extent
type	character string of provider imagery type (see Details)
crop_to_buffer	crop to the user extent, used for creation of output objects (otherwise is padded tile extent)
format	tile format to use, defaults to "jpg" for Mapbox satellite imagery and "png" otherwise
...	arguments passed to internal function, specifically base_url (see Details)
zoom	desired zoom for tiles, use with caution - if NULL is chosen automatically
debug	optionally avoid actual download, but print out what would be downloaded in non-debug mode
max_tiles	maximum number of tiles - if NULL is set by zoom constraints
base_url	tile provider URL expert use only
verbose	report messages or suppress them
filename	purely for read_tiles() this is passed down to the the terra package function 'merge'

Details

get_tiles() may be run with no arguments, and will download (and report on) the default tile source at zoom 0. Arguments type, zoom (or max_tiles), format may be used without setting loc or buffer and the entire world extent will be used. Please use with caution! There is no maximum on what will be downloaded, but it can be interrupted at any time.

Use debug = TRUE to avoid download and simply report on what would be done.

Available types are 'elevation-tiles-prod' for AWS elevation tiles, and 'mapbox.satellite', 'mapbox.terrain-rgb'. (The RGB terrain values are not unpacked.)

Function read_tiles() will match what get_tiles() does and actually build a raster object.

Value

A list with files downloaded in character vector, a data frame of the tile indices, the zoom level used and the extent in xmin,xmax,ymin,ymax form.

See Also

get_tiles_zoom get_tiles_dim get_tiles_buffer

Examples

```
if (!is.null(get_api_key())) {
   tile_info <- get_tiles(ext(146, 147, -43, -42), type = "mapbox.satellite", zoom = 5)
}
```

`mercator_tile_extent` *Tile extent*

Description

Calculate tile extent for a given x, y tile at a zoom level.

Usage

```
mercator_tile_extent(tile_x, tile_y, zoom, tile_size = 256)
```

Arguments

`tile_x`	x coordinate of tile
`tile_y`	y coordinate of tile
`zoom`	zoo level
`tile_size`	tile dimensions (assumed square, i.e. 256x256)

Details

Currently only spherical Mercator is supported.

Value

A numeric vector of the spatial extent, in 'xmin', 'xmax', 'ymin', 'ymax' form.

Examples

```
mercator_tile_extent(2, 4, zoom = 10)

global <- mercator_tile_extent(0, 0, zoom = 0)
plot(NA, xlim = global[c("xmin", "xmax")], ylim = global[c("ymin", "ymax")])
rect_plot <- function(x) rect(x["xmin"], x["ymin"], x["xmax"], x["ymax"])
rect_plot(mercator_tile_extent(1, 1, zoom = 2))
rect_plot(mercator_tile_extent(2, 1, zoom = 2))
rect_plot(mercator_tile_extent(1, 2, zoom = 2))

rect_plot(mercator_tile_extent(1, 1, zoom = 4))
rect_plot(mercator_tile_extent(2, 1, zoom = 4))
rect_plot(mercator_tile_extent(1, 2, zoom = 4))
```

plot_tiles *Plot slippy map tiles*

Description

Create a new plot of tile rectangles, or add to an existing plot.

Usage

```
plot_tiles(
  x,
  ...,
  add = FALSE,
  label = TRUE,
  cex = 0.6,
  add_coast = TRUE,
  include_zoom = TRUE
)

tiles_to_polygon(x)
```

Arguments

x	tiles as create by `ceramic_tiles()`
...	arguments passed to `graphics::rect()`
add	add to an existing plot?
label	include text label?
cex	relative size of text label if drawn (see `text()`)
add_coast	include a basic coastline on the plot?
include_zoom	include zoom level with text label if drawn?

Details

The extent ('xmin', 'xmax', 'ymin', 'ymax') is used directly to draw the tiles so must be in the native Mercator coordinate system used by most tile servers.

Value

`plot_tiles()` is called for its side-effect, a plot, and returns `NULL` invisibly. tiles_to_polygon returns a wk rct vector

Examples

```
if (!is.null(get_api_key())) {
  get_tiles_zoom(zoom = 1)
  tiles <- ceramic_tiles(zoom = 1)
  plot_tiles(tiles)
}
```

---

unpack_rgb *Unpack Mapbox terrain-RGB*

Description

Mapbox terrain-rgb stores global elevation packed into Byte integers.

Usage

```
unpack_rgb(x, filename = "")
```

Arguments

x	three layer raster object
filename	optional, filename to store the output

Details

This function unpacks the three layers of a raster to give floating point elevation data.

Value

terra rast object with one numeric layer

Examples

```
if (interactive() && !is.null(get_api_key())) {
unpack_rgb(read_tiles(type = "mapbox.terrain-rgb"))
}
```

Index