Package ‘geoviz’

January 12, 2020

Type Package
Title Elevation and GPS Data Visualisation
Version 0.2.2
Author Neil Charles
Maintainer Neil Charles <neil.d.charles@gmail.com>
Description Simpler processing of digital elevation model and GPS trace data for use with the 'rayshader' package.
URL https://github.com/neilcharles/geoviz/
License GPL-3
Encoding UTF-8
LazyData true
RoxygenNote 7.0.1
Language en-GB
Imports dplyr, magrittr, tidyr, readr, tibble, purrr, stringr, raster,
chron, sp, sf, rgeos, glue, png, abind, rgl, slippymath, curl,
progress, methods, rlang, ggplot2, rgdal
Suggests testthat, knitr, rmarkdown
VignetteBuilder knitr
NeedsCompilation no
Repository CRAN
Date/Publication 2020-01-12 10:50:02 UTC

R topics documented:

add_gps_to_rayshader ........................................... 2
crop_raster_square ............................................. 3
crop_raster_track ............................................. 4
dplyr_data ...................................................... 5
drybrush ......................................................... 5
elevation_shade ................................................ 6
add_gps_to_rayshader

Adds a GPS trace to a 'rayshader' scene

Description

Adds a GPS trace to a 'rayshader' scene

Usage

```r
add_gps_to_rayshader(
  raster_input,
  lat,
  long,
  alt,
  zscale,
  line_width = 1,
  colour = "red",
  alpha = 0.8,
  lightsaber = TRUE,
  clamp_to_ground = FALSE,
  raise_agl = 0,
  ground_shadow = FALSE,
  as_line = TRUE,
  point_size = 20
)
```

Arguments

```r
raster_input a raster
lat vector of decimal latitude points
long vector of decimal longitude points
```
**crop_raster_square**

- **alt** vector of altitudes
- **zscale** ratio of raster cells to altitude
- **line_width** line width of the gps trace
- **colour** colour of the gps trace
- **alpha** alpha of the gps trace (has no effect if lightsaber = TRUE)
- **lightsaber** (default = TRUE) gives the GPS trace an inner glow affect
- **clamp_to_ground** (default = FALSE) clamps the gps trace to ground level + raise_agl
- **raise_agl** (default = 0) raises a clamped to ground track by the specified amount. Useful if gps track occasionally disappears into the ground.
- **ground_shadow** (default = FALSE) adds a ground shadow to a flight gps trace
- **as_line** (default = TRUE) Set to FALSE to render single points instead of a trace line (which then ignores line_width & lightsaber)
- **point_size** size of points when as_line = TRUE

**Value**

Adds GPS trace to the current 'rayshader' scene

**Examples**

```r
flight <- example_igc()
add_gps_to_rayshader(example_raster(),
  flight$lat,
  flight$long,
  flight$altitude,
  zscale = 25)
```

---

**crop_raster_square**  
_Crops a raster and returns a smaller square raster_

**Description**

Crops a raster and returns a smaller square raster

**Usage**

`crop_raster_square(rasterIn, lat, long, square_km, increase_resolution = 1)`

**Arguments**

- **rasterIn** a raster
- **lat** WGS84 latitude of the centre of the cropped square
- **long** WGS84 longitude of the centre of the cropped square
- **square_km** length of one side of the square in km
- **increase_resolution** optional multiplier to increase number of cells in the raster
crop_raster_track

Value

A cropped raster

Examples

crop_raster_square(example_raster(), lat = 54.513293, long = -3.045598, square_km = 0.01)

crop_raster_track Crops a raster into a rectangle surrounding a set of lat long points

Description

Crops a raster into a rectangle surrounding a set of lat long points

Usage

crop_raster_track(
  raster_input,
  lat_points,
  long_points,
  width_buffer = 1,
  increase_resolution = 1
)

Arguments

raster_input a raster
lat_points a vector of WGS84 latitudes
long_points a vector of WGS84 longitudes
width_buffer buffer distance around the provided points in km
increase_resolution optional multiplier to increase number of cells in the raster. Default = 1.

Value

cropped raster

Examples

crop_raster_track(example_raster(), example_igc()$lat, example_igc()$long)
Description

See https://github.com/STAT545-UBC/Discussion/issues/451

drybrush

Simulates a dry brushing effect. Differs from elevation_transparency() in that colour is applied based on local altitude peaks, not across the whole raster

Usage

drybrush(
  raster_dem,
  aggregation_factor = 10,
  max_colour_altitude = 30,
  opacity = 0.5,
  elevation_palette = c("#3f3f3f", "#ffa500")
)

Arguments

raster_dem      A raster
aggregation_factor
  grid size to determine local altitude peaks
max_colour_altitude
  Altitude below which colours will be graduated across elevation_palette
opacity
  overall opacity of the returned image
elevation_palette
  Colour scheme c(colour_for_low_altitude, colour_for_high_altitude)

Value

An image with a drybrushed colour effect, highlighting local peaks

Examples

overlay_image <- drybrush(example_raster())
elevation_shade  
*Produces an elevation shaded image from a raster*

**Description**

Produces an elevation shaded image from a raster

**Usage**

```r
elevation_shade(
  raster_dem,
  elevation_palette = c("#54843f", "#808080", "#FFFFFF"),
  return_png = TRUE,
  png_opacity = 0.9
)
```

**Arguments**

- `raster_dem`: a raster
- `elevation_palette`: a vector of colours to use for elevation shading
- `return_png`: TRUE to return an image. FALSE will return a raster
- `png_opacity`: Opacity of the returned image. Ignored if `return_png` = FALSE

**Value**

elevation shaded image

**Examples**

```r
elevation_shade(example_raster())
```

elevation_transparency  
*Turns overlay images transparent based on altitude. Can be used to create an image overlay that will only apply to valleys, or only to hills.*

**Description**

Turns overlay images transparent based on altitude. Can be used to create an image overlay that will only apply to valleys, or only to hills.
elevation_transparency

Usage

elevation_transparency(
    overlay_image,
    raster_dem,
    alpha_max = 0.4,
    alpha_min = 0,
    pct_alt_low = 0.05,
    pct_alt_high = 0.25
)

Arguments

overlay_image the image on which to alter transparency
raster_dem elevation model raster file that will be used to adjust transparency
alpha_max Transparency required at higher altitudes
alpha_min Transparency required at lower altitudes
pct_alt_low The percent of maximum altitude contained in raster_dem at which alpha_max will apply
pct_alt_high The percent of maximum altitude contained in raster_dem at which alpha_min will apply

Value

An image with transparency defined by altitude

Examples

# elevation_transparency defaults to making hills transparent. Flip alpha_max
# and alpha_min values to reverse it.
#
# Transparency in the range between pct_alt_low and pct_alt_high will
# smoothly transition between alpha_max and alpha_min.

overlay_image <- elevation_shade(example_raster(), elevation_palette = c("#000000", "#FF0000"))

#Making hills transparent

ggmap_overlay_transparent_hills <- elevation_transparency(overlay_image,
    example_raster(), alpha_max = 0.8, alpha_min = 0, pct_alt_low = 0.05,
    pct_alt_high = 0.25)

# To make valleys transparent, flip alpha_max and alpha_min

ggmap_overlay_transparent_valleys <- elevation_transparency(overlay_image,
    example_raster(), alpha_max = 0, alpha_min = 0.8, pct_alt_low = 0.05,
    pct_alt_high = 0.25)
example_igc  

Returns an example IGC file using read_igc()

Description

Returns an example IGC file using read_igc()

Usage

element_igc()

Value

a tibble

Examples

# Loads a paragliding flight GPS track, originally downloaded from xcleague.com
igc <- example_igc()

example_raster  

Returns an example digital elevation model raster file()

Description

Returns an example digital elevation model raster file()

Usage

element_raster()

Value

a raster

Examples

# Load elevation data describing a small section of the English Lake District
# Source: EU Copernicus https://land.copernicus.eu/terms-of-use
example_raster <- example_raster()
get_slippy_map

Obtains and merges map tiles from various sources using the 'slippymath' package

Description

Obtains and merges map tiles from various sources using the 'slippymath' package

Usage

```r
get_slippy_map(
  bounding_box,
  image_source = "stamen",
  image_type = "watercolor",
  max_tiles = 10,
  api_key
)
```

Arguments

- **bounding_box**: Any object for which raster::extent() can be calculated.
- **image_source**: Source for the overlay image. Valid entries are "mapbox", "mapzen", "stamen".
- **image_type**: The type of overlay to request. "satellite", "mapbox-streets-v8", "mapbox-terrain-v2", "mapbox-traffic-v1", "terrain-rgb", "mapbox-incidents-v1" (mapbox), "dem" (mapzen) or "watercolor", "toner", "toner-background", "toner-lite" (stamen).
  You can also request a custom Mapbox style by specifying `image_source = "mapbox", image_type = "username/mapid"
- **max_tiles**: Maximum number of tiles to be requested by 'slippymath'
- **api_key**: API key (required for 'mapbox')

Value

a rasterBrick with the same dimensions (but not the same resolution) as bounding_box

Examples

```r
map <- get_slippy_map(example_raster(),
  image_source = "stamen",
  image_type = "watercolor",
  max_tiles = 5)
```
ggslippy  

* Adds a layer created using slippy_overlay() or slippy_raster() to a `ggplot2` chart

---

**Description**

Adds a layer created using slippy_overlay() or slippy_raster() to a `ggplot2` chart

**Usage**

```r
ggslippy(slippy_raster, alpha = 1, set_coord_equal = TRUE)
```

**Arguments**

- `slippy_raster`: A raster raster returned by either slippy_raster() or slippy_overlay(return_png = FALSE)
- `alpha`: Opacity of the raster in `ggplot2`
- `set_coord_equal`: TRUE returns a square plot

**Value**

a `ggplot` object

**Examples**

```r
library(ggplot2)
library(geoviz)

dem <- example_raster()
dem <- raster::aggregate(dem, 10) # aggregate to speed up ggplot for testing
gg_overlay_image <- slippy_overlay(
  dem,
  image_source = "stamen",
  image_type = "watercolor",
  return_png = FALSE,
  max_tiles = 2
)

ggplot() +
ggslippy(gg_overlay_image, set_coord_equal = FALSE)
```
**latlong_to_rayshader_coords**

*Converts WGS84 lat long points into 'rayshader' coordinates. Useful for adding arbitrary points and text to a 'rayshader' scene.*

**Description**

Converts WGS84 lat long points into 'rayshader' coordinates. Useful for adding arbitrary points and text to a 'rayshader' scene.

**Usage**

```r
latlong_to_rayshader_coords(raster_input, lat, long)
```

**Arguments**

- **raster_input**: a raster
- **lat**: vector of WGS84 latitude points
- **long**: vector of WGS84 longitude points

**Value**

A tibble with x,y in 'rayshader' coordinates

**Examples**

```r
latlong_to_rayshader_coords(example_raster(), example_igc()$lat, example_igc()$long)
```

---

**mapbox_dem**

*Gets Digital Elevation Model (DEM) data from 'mapbox'*

**Description**

Gets Digital Elevation Model (DEM) data from 'mapbox'

**Usage**

```r
mapbox_dem(lat, long, square_km, width_buffer = 1, max_tiles = 10, api_key)
```
Arguments

- **lat**: WGS84 latitude. Either a single point to use as the centre for a square_km sized raster, or a vector of track points.
- **long**: WGS84 longitude. Either a single point to use as the centre for a square_km sized raster, or a vector of track points.
- **square_km**: length of one edge the required square area, in km. Ignored if lat and long have length > 1.
- **width_buffer**: If lat and long have length > 1, used as buffer distance around the provided points in km.
- **max_tiles**: maximum number of map tiles to request. More tiles will give higher resolution scenes but take longer to download. Note that very small numbers of tiles may result in a scene that is not square.
- **api_key**: 'Mapbox' API key

Value

a raster with values corresponding to terrain height in metres

Examples

```r
# Not run:
# NOT RUN
#mapbox_dem() requires a 'mapbox' API key

mapbox_key <- "YOUR_MAPBOX_API_KEY"

lat <- 54.4502651
long <- -3.1767946
square_km <- 20

dem <- mapbox_dem(lat, long, square_km, api_key = mapbox_key)
```

## End(Not run)

mapzen_dem

*Gets Digital Elevation Model (DEM) data from 'mapzen' via 'Amazon Public Datasets'*

Description

Gets Digital Elevation Model (DEM) data from 'mapzen' via 'Amazon Public Datasets'

Usage

`mapzen_dem(lat, long, square_km, width_buffer = 1, max_tiles = 10)`
Arguments

lat  

WGS84 latitude. Either a single point to use as the centre for a square_km sized raster, or a vector of track points

long  

WGS84 longitude. Either a single point to use as the centre for a square_km sized raster, or a vector of track points

square_km  

length of one edge the required square area, in km. Ignored if lat and long have length > 1

width_buffer  

If lat and long have length > 1, used as buffer distance around the provided points in km

max_tiles  

maximum number of map tiles to request. More tiles will give higher resolution scenes but take longer to download. Note that very small numbers of tiles may result in a scene that is not square.

Value

a raster with values corresponding to terrain height in metres

Examples

lat = 54.4502651
long = -3.1767946
square_km = 2

dem <- mapzen_dem(lat, long, square_km, max_tiles = 2)

mosaic_files  

Stitches together files into a single raster Requires a target directory of files that can be read with raster::raster(), e.g. .asc files, or a directory of .zip files containing these files

Description

Stitches together files into a single raster Requires a target directory of files that can be read with raster::raster(), e.g. .asc files, or a directory of .zip files containing these files

Usage

mosaic_files(
  path,
  extract_zip = FALSE,
  file_match = ".*.asc",
  zip_file_match = ".*.zip",
  raster_output_file = "mosaic_out.raster",
  file_crs = NULL,
  raster_todisk = FALSE
)
Arguments

path: path to files that are to be stitched together
extract_zip: FALSE to target .asc files, TRUE if your .asc files are zipped.
file_match: regex pattern to match .asc files, either in path or in zip files.
zip_file_match: regex pattern to match .zip files
raster_output_file: raster file to be created (will overwrite existing files)
file_crs: projection string of the input files. Output will always be WGS84.
raster_todisk: Setting TRUE will set rasterOptions(todisk=TRUE), which can help with memory issues.

Value

TRUE

Examples

# Merge two small example .asc files of LIDAR data
# from https://environment.data.gov.uk (open government licence)

path_to_files <- system.file("extdata/example_asc", package = "geoviz")
path_to_output <- tempdir()
mosaic_files(path_to_files,
    raster_output_file = paste0(path_to_output, "/mosaic_out.raster", sep = ""),
    extract_zip = TRUE, file_crs = "+init=epsg:27700"
)
raster_mosaic <- raster::raster(paste0(path_to_output, "/mosaic_out.gri", sep = ""))

raster_zscale: Approximates the zscale of a raster Digital Elevation Model for 'rayshader'

Description

Approximates the zscale of a raster Digital Elevation Model for 'rayshader'

Usage

raster_zscale(raster, height_units = "m")

Arguments

raster: A raster object of elevation data values
height_units: Elevation units of the raster, c("m", "feet")
read_igc

Value

    a number to be used as zscale in rayshader::plot_3d()

Examples

    raster_zscale(example_raster())

read_igc  Load an IGC file

Description

    Load an IGC file

Usage

    read_igc(path)

Arguments

    path  target IGC file

Value

    a tibble

Examples

    igc <- read_igc(system.file("extdata/example.igc", package = "geoviz"))

slippy_overlay  Creates an overlay image from 'Mapbox' or 'Stamen' Maps using the 'slippymath' package

Description

    Creates an overlay image from 'Mapbox' or 'Stamen' Maps using the 'slippymath' package
Usage

slippy_overlay(
  raster_base,
  image_source = "stamen",
  image_type = "watercolor",
  max_tiles = 10,
  api_key,
  return_png = TRUE,
  png_opacity = 0.9
)

Arguments

raster_base   A raster to use to calculate dimensions for the overlay
image_source  Source for the overlay image. Valid entries are "mapbox", "stamen".
image_type    The type of overlay to request. "satellite", "mapbox-streets-v8", "mapbox-terrain-v2", "mapbox-traffic-v1", "terrain-rgb", "mapbox-incidents-v1" (mapbox), "dem" (mapzen) or "watercolor", "toner", "toner-background", "toner-lite" (stamen). You can also request a custom Mapbox style by specifying image_source = "mapbox", image_type = "username/mapid"
max_tiles     Maximum number of tiles to be requested by slippymath
api_key       API key (required for mapbox)
return_png    TRUE to return a png image. FALSE will return a raster
png_opacity   Opacity of the returned image. Ignored if return_png = FALSE

Value

an overlay image for raster_base

Examples

overlay_image <- slippy_overlay(example_raster(),
  image_source = "stamen",
  image_type = "watercolor",
  max_tiles = 2)

slippy_raster

Creates a square raster centred on any lat long point, or a rectangular raster surrounding a set of lat long points from 'Mapbox', 'Mapzen' or 'Stamen' Maps using the 'slippymath' package

Description

Creates a square raster centred on any lat long point, or a rectangular raster surrounding a set of lat long points from 'Mapbox', 'Mapzen' or 'Stamen' Maps using the 'slippymath' package
Usage

```
slippy_raster(
  lat, 
  long, 
  square_km, 
  width_buffer = 1, 
  image_source = "stamen", 
  image_type = "watercolor", 
  max_tiles = 10, 
  api_key
)
```

Arguments

- `lat` WGS84 latitude. Either a single point to use as the centre for a `square_km` sized raster, or a vector of track points
- `long` WGS84 longitude. Either a single point to use as the centre for a `square_km` sized raster, or a vector of track points
- `square_km` length of one edge the required square area, in km. Ignored if lat and long have length > 1
- `width_buffer` If lat and long have length > 1, used as buffer distance around the provided points in km
- `image_source` Source for the overlay image. Valid entries are "mapbox", "mapzen", "stamen".
- `image_type` The type of overlay to request. "satellite", "mapbox-streets-v8", "mapbox-terrain-v2", "mapbox-traffic-v1", "terrain-rgb", "mapbox-incidents-v1" (mapbox), "dem" (mapzen) or "watercolor", "toner", "terrain" (stamen)
- `max_tiles` Maximum number of tiles to be requested by `slippymath`
- `api_key` API key (required for 'mapbox')

Value

a rasterBrick image

Examples

```
lat <- 54.4502651
long <- -3.1767946
square_km <- 1

overlay_image <- slippy_raster(lat = lat, 
                               long = long, 
                               square_km = square_km, 
                               image_source = "stamen", 
                               image_type = "watercolor", 
                               max_tiles = 5)
```
Index

add_gps_to_rayshader, 2

crop_raster_square, 3
crop_raster_track, 4

dplyr_data, 5
drybrush, 5

elevation_shade, 6
elevation_transparency, 6
example_igc, 8
example_raster, 8

gslippy_map, 9
ggslippy, 10

latlong_to_rayshader_coords, 11

mapbox_dem, 11
mapzen_dem, 12
mosaic_files, 13

raster_zscale, 14
read_igc, 15

slippy_overlay, 15
slippy_raster, 16