Package: quadkeyr (via r-universe)

September 29, 2026

Title Generate Raster Images from QuadKey-Identified Datasets

Version 0.1.0

Description A set of functions of increasing complexity allows users to (1) convert QuadKey-identified datasets, based on 'Microsoft's Bing Maps Tile System', into Simple Features data frames, (2) transform Simple Features data frames into rasters, and (3) process multiple 'Meta' ('Facebook') QuadKey-identified human mobility files directly into raster files. For more details, see D'Andrea et al. (2024) <doi:10.21105/joss.06500>.

License MIT + file LICENSE

URL https://docs.ropensci.org/quadkeyr/, https://github.com/ropensci/quadkeyr

BugReports https://github.com/ropensci/quadkeyr/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Suggests bslib (>= 0.6.1), DT (>= 0.31), ggplot2 (>= 3.4.4), knitr, leaflet (>= 2.2.1), magick (>= 2.8.5), rmarkdown, rnaturalearth (>= 1.0.0), shinytest2 (>= 0.3.2), testthat (>= 3.1.10), tidyr (>= 1.3.0), viridis (>= 0.6.4), withr

VignetteBuilder knitr

Depends R (>= 4.1.0)

LazyData true

Imports dplyr (>= 1.1.2), lubridate (>= 1.9.2), purrr (>= 1.0.1), readr (>= 2.1.4), rlang (>= 1.1.2), sf (>= 1.0.14), shiny (>= 1.7.4), stars (>= 0.6.2)

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev make libuv1-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev libx11-dev zlib1g-dev

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

Date/Publication 2025-03-25 19:31:35 UTC

RemoteUrl https://github.com/ropensci/quadkeyr

RemoteRef main

RemoteSha 07cc343b2d77d25b8e3e802fc3a5ca5e9bd72d0e

Contents

add_regular_polygon_grid . . . 2
apply_weekly_lag . . . 3
create_qk_grid . . . 4
create_stars_raster . . . 5
format_fb_data . . . 7
get_qk_coord . . . 8
get_regular_polygon_grid . . . 9
grid_to_polygon . . . 10
ground_res . . . 10
latlong_to_pixelXY . . . 11
latlong_to_quadkey . . . 12
mapscale . . . 13
mapsize . . . 14
missing_combinations . . . 15
pixelXY_to_latlong . . . 15
pixelXY_to_tileXY . . . 16
polygon_to_raster . . . 17
qkmap_app . . . 19
quadkey_df_to_polygon . . . 19
quadkey_to_latlong . . . 20
quadkey_to_polygon . . . 21
quadkey_to_tileXY . . . 22
read_fb_mobility_files . . . 23
regular_qk_grid . . . 24
tileXY_to_pixelXY . . . 25
tileXY_to_quadkey . . . 26

Index **28**

`add_regular_polygon_grid`

Add the rows needed to complete a regular QuadKey polygon grid derived from the bounding box of the `quadkey` *column of a data.frame.*

Description

This function estimates the bounding box of the quadkeys given in the quadkey column and adds rows to complete the quadkeys and the geometry needed to create a regular grid. All other columns for the introduced QuadKeys will be filled with NAs.

For a detailed explanation on how to use this and other similar quadkeyr functions, read the the vignette: https://docs.ropensci.org/quadkeyr/articles/quadkey_identified_data_to_raster.html

Usage

```
add_regular_polygon_grid(data)
```

Arguments

data	A data.frame with a quadkey column.

Value

A list with three elements:

- data A sf POLYGON data.frame with all the QuadKeys within the bounding box of the ones provided in the quadkey column of the input dataset, and the rest of the original variables. The columns quadkey and geometry are returned for all the grid, The values of the newly added QuadKeys will be NA for the rest of the variables.
- num_rows The number of columns of the regular grid.
- num_cols The number of rows of the regular grid.

Examples

```
# Read the file with the data
path <- paste0(
  system.file("extdata", package = "quadkeyr"),
  "/cityA_2020_04_15_0000.csv"
)
data <- read.csv(path)
data <- format_fb_data(data)

add_regular_polygon_grid(data = data)
```

---

apply_weekly_lag *Apply a 7 day lag to the variable* n_crisis

---

Description

Applying a week lag to the data will create raster images showing the mobility a week before the date of interest. This function works only for QuadKeys reported without NAs for n_crisis and percent_change variables .

Usage

```
apply_weekly_lag(data)
```

Arguments

data	A data.frame with the columns quadkey, day, hour and n_crisis.

Value

A data.frame with the extra columns n_crisis_lag_7 and percent_change_7.

- n_crisis_lag_7, is the same variable defined as n_crisis in the Facebook mobility data.frame with a 7 day lag applied.
- percent_change_7 is the difference between the n_crisis value between weeks expressed as percentage.

Examples

```
files <- read_fb_mobility_files(
  path_to_csvs = paste0(system.file("extdata",
    package = "quadkeyr"
  ), "/"),
  colnames = c(
    "lat",
    "lon",
    "quadkey",
    "date_time",
    "n_crisis",
    "percent_change"
  ),
  coltypes = list(
    lat = "d",
    lon = "d",
    quadkey = "c",
    date_time = "T",
    n_crisis = "c",
    percent_change = "c"
  )
)

apply_weekly_lag(data = files)
```

---

create_qk_grid *Create grid of QuadKeys for a particular zoom or level of detail.*

Description

Generates a grid comprising all the QuadKeys within the area defined by the maximum and minimum coordinates of latitude and longitude along with a specified zoom level.

Usage

```
create_qk_grid(xmin, xmax, ymin, ymax, zoom)
```

Arguments

xmin	Minimum value in the x axis (longitude)
xmax	Maximum value in the y axis (latitude)
ymin	Minimum value in the x axis (longitude)
ymax	Maximum value in the Y axis (latitude)
zoom	Zoom or level of detail, from 1 (lowest detail) to 23 (highest detail).

Value

A list returning the QuadKeys as a data.frame (`data`), the number of rows (`num_rows`) and columns (`num_cols`) of the grid.

Examples

```
grid <- create_qk_grid(
  xmin = -59,
  xmax = -57,
  ymin = -35,
  ymax = -34,
  zoom = 12
)
```

`create_stars_raster` *Create a* `stars` *raster*

Description

The use of a template enables the creation of an accurate raster, even in the presence of NAs.

Usage

```
create_stars_raster(template, nx, ny, data, var)
```

Arguments

template	A `sf` POLYGON data.frame to use as template. Check `stars::st_as_stars()` documentation for more details.
nx	Integer; number of cells in x direction.
ny	Integer; number of cells in y direction.
data	A `sf` POLYGON data.frame with the variable we want to represent in the raster.
var	The column name of the variable to plot.

Value

A stars object.

See Also

st_as_stars, st_rasterize

Examples

```
# Basic workflow
# Read the data
path <- paste0(
  system.file("extdata", package = "quadkeyr"),
  "/cityA_2020_04_15_0000.csv"
)
data <- read.csv(path)
data <- format_fb_data(data)

complete_polygon_grid <- add_regular_polygon_grid(data = data)

stars_object <- create_stars_raster(
  data = complete_polygon_grid$data,
  template = complete_polygon_grid$data,
  var = "percent_change",
  nx = complete_polygon_grid$num_cols,
  ny = complete_polygon_grid$num_rows
)
stars_object

# Other workflow
grid <- create_qk_grid(
  xmin = -59,
  xmax = -57,
  ymin = -35,
  ymax = -34,
  zoom = 12
)

grid_coords <- get_qk_coord(data = grid$data)

polygrid <- grid_to_polygon(grid_coords)

data("data_provided")
data_raster <- polygrid |>
  dplyr::inner_join(data_provided,
    by = c("quadkey")
  )

raster <- create_stars_raster(
  template = data_raster,
  nx = grid$num_cols,
  ny = grid$num_rows,
```

```
  data = data_raster,
  var = "variable"
)
```

`format_fb_data` *Format the Facebook mobility data*

Description

This function removes unnecessary characters such as `\\N` and ensures that the format of the date and QuadKeys is correct.

Usage

```
format_fb_data(data, keep_format = NULL)
```

Arguments

`data`	A data.frame with a `quadkey` and `date_time` columns and other variables
`keep_format`	Vector of column names, besides `date_time`, `day` and `quadkey`, that you don't want to convert to a number.

Value

A data.frame without `\N`, `quadkey` without scientific notation and a new column `day` and `hour`

See Also

`read_fb_mobility_files`

Examples

```
data(result_read_fb_mobility_data)
format_fb_data(data = result_read_fb_mobility_data)
```

`get_qk_coord` *Get lat/long coordinates from the QuadKey*

Description

Reads the QuadKey as a string and extracts the lat/long coordinates of the upper-left corner of the QuadKey.

Usage

```
get_qk_coord(data)
```

Arguments

`data`	A dataframe with a quadkey column.

Value

A `sf` POINT data.frame containing the tiles XY coordinates (`tileX`, `tileY`), the QuadKeys (`quadkey`), and a `geometry` column.

See Also

`quadkey_to_tileXY`

`tileXY_to_pixelXY`

`pixelXY_to_latlong`

Examples

```
grid <- create_qk_grid(
  xmin = -59,
  xmax = -40,
  ymin = -38,
  ymax = -20,
  zoom = 6
)

# quadkey column in grid$data converted to geographic coordinates
grid_coords <- get_qk_coord(data = grid$data)

plot(grid_coords)
```

get_regular_polygon_grid *Get regular QuadKey polygon grid derived from the bounding box of the* quadkey *column of a data.frame.*

Description

This function estimates the bounding box of the QuadKeys given in the quadkey column and adds the rows needed to complete a regular grid.

For a detailed explanation on how to use this and other similar quadkeyr functions, read the the vignette: https://docs.ropensci.org/quadkeyr/articles/facebook_mobility_csvs_to_raster_files.html

Usage

```
get_regular_polygon_grid(data)
```

Arguments

data	A data.frame with a quadkey column.

Value

A list with three elements:

- data An sf POLYGON data.frame with all the QuadKeys within the bounding box of the quadkey column of a data.frame. Only the columns quadkey, tileX, tileY and geometry are returned.
- num_rows The number of columns of the regular grid.
- num_cols The number of rows of the regular grid.

Examples

```
# Data File
path <- paste0(
  system.file("extdata", package = "quadkeyr"),
  "/cityA_2020_04_15_0000.csv"
)
data <- read.csv(path)
data <- format_fb_data(data)

get_regular_polygon_grid(data = data)
```

grid_to_polygon *Convert a grid of QuadKeys to square polygons*

Description

The main argument of this function, the grid of geographic coordinates (lat/long WG84) represents the upper-left corner of the QuadKey. To transform these coordinates into square polygons, the function supplements the grid by adding a row and column of tiles. These points introduce QuadKeys located at the border of the area using the internal function `complete_grid_for_polygons()`. The function builds the polygons using all the points of the grid. Note that it's possible to associate each QuadKey with its square polygon.

Usage

```
grid_to_polygon(data)
```

Arguments

`data`	A `sf` POINT data.frame with a `quadkey` and `geometry` column.

Value

A sf POLYGON data.frame with a quadkey column.

Examples

```
grid <- create_qk_grid(
  xmin = -59,
  xmax = -57,
  ymin = -35,
  ymax = -34,
  zoom = 11
)

grid_coords <- get_qk_coord(data = grid$data)

polygrid <- grid_to_polygon(grid_coords)
polygrid
```

ground_res *Ground resolution at a specified latitude and zoom level*

Description

Determines the ground resolution (in meters per pixel) at a specified latitude and zoom level. For further information, refer to the Microsoft Bing Maps Tile System documentation.

Usage

```
ground_res(latitude, zoom)
```

Arguments

latitude	Latitude (in degrees) at which to measure the ground resolution.
zoom	Zoom or level of detail, from 1 (lowest detail) to 23 (highest detail).

Value

the ground resolution (meters / pixel)

References

https://learn.microsoft.com/en-us/bingmaps/articles/bing-maps-tile-system

Examples

```
ground_res(
  latitude = 0,
  zoom = 6
)
```

---

latlong_to_pixelXY *Convert lat/long coordinates to pixel XY coordinates*

Description

Converts a point from latitude/longitude WGS-84 coordinates (in degrees) into pixel XY coordinates at a specified zoom level. For further information, refer to the Microsoft Bing Maps Tile System documentation.

Usage

```
latlong_to_pixelXY(lat, lon, zoom)
```

Arguments

lat	Latitude of the point, in degrees.
lon	Longitude of the point, in degrees.
zoom	Zoom or level of detail, from 1 (lowest detail) to 23 (highest detail).

Details

Converting latitude/longitude coordinates into a QuadKey and then back to latitude/longitude won't yield identical values, unless the initial latitude/longitude coordinates correspond to the upper-left Quadkey's pixel and tile XY coordinates at the same zoom level.

Understanding this distinction is crucial for the accurate use of these functions in coordinate conversions.

For a detailed explanation on how to use this and other similar `quadkeyr` functions, read the the vignette: https://docs.ropensci.org/quadkeyr/articles/quadkey_to_sf_conversion.html

Value

A list returning pixel X and pixel Y coordinates.

References

https://learn.microsoft.com/en-us/bingmaps/articles/bing-maps-tile-system

Examples

```
latlong_to_pixelXY(
  lat = -35,
  lon = -50,
  zoom = 6
)
```

---

latlong_to_quadkey *Convert latitude/longitude coordinates into QuadKeys*

Description

Converts a point from latitude/longitude WGS-84 coordinates (in degrees) into a Quadkey at a specified zoom level. For further information, refer to the Microsoft Bing Maps Tile System documentation.

Usage

```
latlong_to_quadkey(lat, lon, zoom)
```

Arguments

`lat`	Latitude of the point, in degrees.
`lon`	Longitude of the point, in degrees.
`zoom`	Zoom or level of detail, from 1 (lowest detail) to 23 (highest detail).

Details

Converting latitude/longitude coordinates into a QuadKey and then back to latitude/longitude won't yield identical values, unless the initial latitude/longitude coordinates correspond to the upper-left Quadkey's pixel and tile XY coordinates at the same zoom level.

Understanding this distinction is crucial for the accurate use of these functions in coordinate conversions.

For a detailed explanation on how to use this and other similar `quadkeyr` functions, read the the vignette: https://docs.ropensci.org/quadkeyr/articles/quadkey_to_sf_conversion.html

Value

A dataframe with latitude (lat), longitude (lon), zoom level (zoom) and the QuadKey as a string (quadkey).

References

https://learn.microsoft.com/en-us/bingmaps/articles/bing-maps-tile-system

Examples

```
latlong_to_quadkey(
  lat = 35.63051,
  lon = 139.69116,
  zoom = 20
)
latlong_to_quadkey(
  lat = c(-4, -25.33, -25.66),
  lon = c(-53, -60.33, -70.66),
  zoom = 4
)
```

---

mapscale *Map scale (1 : N)*

Description

Determines the map scale at a specified latitude, zoom level, and screen resolution. For further information, refer to the Microsoft Bing Maps Tile System documentation.

Usage

```
mapscale(latitude, zoom, screen_dpi)
```

Arguments

`latitude`	Latitude (in degrees) at which to measure the map scale.
`zoom`	Zoom or level of detail, from 1 (lowest detail) to 23 (highest detail).
`screen_dpi`	Resolution of the screen, in dots per inch.

Value

The map scale, expressed as the denominator N of the ratio 1 : N.

References

https://learn.microsoft.com/en-us/bingmaps/articles/bing-maps-tile-system

Examples

```
mapscale(
  latitude = 0,
  zoom = 6,
  screen_dpi = 96
)
```

---

mapsize *Map size in pixels*

Description

Determines the map width and height (in pixels) at a specified zoom level. For further information, refer to the Microsoft Bing Maps Tile System documentation.

Usage

```
mapsize(zoom)
```

Arguments

zoom	Zoom or level of detail, from 1 (lowest detail) to 23 (highest detail).

Value

The map width and height in pixels.

References

https://learn.microsoft.com/en-us/bingmaps/articles/bing-maps-tile-system

Examples

```
mapsize(zoom = 6)
```

missing_combinations *Detect dates and hours missing in filenames*

Description

Facebook mobility data is reported daily at 3 different hours (0, 8, 16). This function reads the data extracted from the current files and detects if any file for a particular day or hour is missing.

Usage

```
missing_combinations(data, hour_col = "hour", date_col = "day")
```

Arguments

data	A data.frame with one column for the raster's date and another for the hour. If not explicitly specified in the function's arguments, the column names are day and hour.
hour_col	The name of the column with the hour information.
date_col	The name of the column with the date information.

Value

A data.frame with the missing days and hours, if any.

Examples

```
# Sample dataset
data <- data.frame(
  country = c("US", "MX", "MX"),
  day = c("2023-01-01", "2023-01-03", "2023-01-05"),
  hour = c(0, 8, 16)
)

missing_combinations(data)
```

pixelXY_to_latlong *Convert pixel XY coordinatess into lat/long coordinates.*

Description

Converts a pixel from pixel XY coordinates at a specified zoom level into latitude/longitude WGS-84 coordinates (in degrees). For further information, refer to the Microsoft Bing Maps Tile System documentation.

Usage

```
pixelXY_to_latlong(pixelX, pixelY, zoom)
```

Arguments

pixelX	X coordinate of the point, in pixels.
pixelY	Y coordinates of the point, in pixels.
zoom	Zoom or level of detail, from 1 (lowest detail) to 23 (highest detail).

Details

Converting latitude/longitude coordinates into a QuadKey and then back to latitude/longitude won't yield identical values, unless the initial latitude/longitude coordinates correspond to the upper-left QuadKey's pixel and tile XY coordinates at the same zoom level.

Understanding this distinction is crucial for the accurate use of these functions in coordinate conversions.

For a detailed explanation on how to use this and other similar quadkeyr functions, read the the vignette: https://docs.ropensci.org/quadkeyr/articles/quadkey_to_sf_conversion.html

Value

A list with the longitude and latitude.

References

https://learn.microsoft.com/en-us/bingmaps/articles/bing-maps-tile-system

Examples

```
pixelXY_to_latlong(
  pixelX = 768,
  pixelY = 1280,
  zoom = 3
)
```

---

pixelXY_to_tileXY *Convert pixel XY coordinates into tile XY coordinates*

Description

Converts pixel XY coordinates into tile XY coordinates of the tile containing the specified pixel. For further information, refer to the Microsoft Bing Maps Tile System documentation.

Usage

```
pixelXY_to_tileXY(pixelX, pixelY)
```

Arguments

pixelX	Pixel X coordinate.
pixelY	Pixel Y coordinate.

Details

Converting latitude/longitude coordinates into a QuadKey and then back to latitude/longitude won't yield identical values, unless the initial latitude/longitude coordinates correspond to the upper-left Quadkey's pixel and tile XY coordinates at the same zoom level.

Understanding this distinction is crucial for the accurate use of these functions in coordinate conversions.

For a detailed explanation on how to use this and other similar quadkeyr functions, read the the vignette: https://docs.ropensci.org/quadkeyr/articles/quadkey_to_sf_conversion.html

Value

A list returning the tile X and tile Y coordinates.

References

https://learn.microsoft.com/en-us/bingmaps/articles/bing-maps-tile-system

Examples

```
pixelXY_to_tileXY(
  pixelX = 5916,
  pixelY = 9894
)
```

---

polygon_to_raster *Create and save raster images for different dates and times*

Description

Creates one raster by each date and time reported and saves it as a .tif.

Usage

```
polygon_to_raster(
  data,
  nx,
  ny,
  template,
  var = "percent_change",
  filename,
  path
)
```

Arguments

data	A sf POLYGON data.frame with the variable we want to represent in the raster.
nx	Integer; number of cells in x direction.
ny	Integer; number of cells in y direction.
template	A sf POLYGON data.frame
var	The column name of the variable to plot.
filename	Select a name for the file. The date and time will be included automatically in the name.
path	Path where the files should be stored.

Value

as many .tif files as dates and times in the dataset

See Also

st_as_stars, st_rasterize

missing_combinations

Examples

```
files <- read_fb_mobility_files(
  path_to_csvs = paste0(system.file("extdata",
    package = "quadkeyr"
  ), "/"),
  colnames = c(
    "lat", "lon",
    "quadkey", "date_time",
    "n_crisis", "percent_change"
  ),
  coltypes = list(
    lat = "d",
    lon = "d",
    quadkey = "c",
    date_time = "T",
    n_crisis = "c",
    percent_change = "c"
  )
)

# Get a regular grid and create the polygons
regular_grid <- get_regular_polygon_grid(data = files)

# Keep only the QuadKeys reported
files_polygons <- files |>
  dplyr::inner_join(regular_grid$data,
    by = c("quadkey")
  )
```

```
# Generate the raster files
polygon_to_raster(
  data = files_polygons,
  nx = regular_grid$num_cols,
  ny = regular_grid$num_rows,
  template = files_polygons,
  var = "percent_change",
  filename = "cityA",
  path = paste0(
    system.file("extdata",
      package = "quadkeyr"
    ),
    "/"
  )
)
```

---

qkmap_app *Launch the Shiny App*

Description

This function launches the Shiny app.

Usage

```
qkmap_app()
```

Value

This function does not return any value. It launches the Shiny app.

Examples

```
if(interactive()){
qkmap_app()
}
```

---

quadkey_df_to_polygon *Convert data.frame with* quadkey *column to a* sf *POLYGON data.frame*

Description

Convert data.frame with quadkey column to a sf POLYGON data.frame

Usage

```
quadkey_df_to_polygon(data)
```

Arguments

data	A data.frame with a quadkey column

Value

The same original data.frame with a sf POLYGON data.frame with a geometry column.

See Also

quadkey_df_to_polygon

Examples

```
# Quadkey as string
quadkey_to_polygon(quadkey = "213")

# QuadKeys as column in a data.frame
# get data file
path <- paste0(
  system.file("extdata", package = "quadkeyr"),
  "/cityA_2020_04_15_0000.csv"
)
data <- read.csv(path)
data <- format_fb_data(data)

quadkey_df_to_polygon(data = data)
```

---

quadkey_to_latlong *Convert a string of Quadkey numbers to lat/long coordinates*

Description

This function converts Quadkeys to latitude/longitude WGS-84 coordinates (in degrees). For further information, refer to the Microsoft Bing Maps Tile System documentation.

Usage

```
quadkey_to_latlong(quadkey_data)
```

Arguments

quadkey_data	A single QuadKey as a string or a vector with unique QuadKeys.

Details

Converting latitude/longitude coordinates into a QuadKey and then back to latitude/longitude won't yield identical values, unless the initial latitude/longitude coordinates correspond to the upper-left Quadkey's pixel and tile XY coordinates at the same zoom level.

Understanding this distinction is crucial for the accurate use of these functions in coordinate conversions.

For a detailed explanation on how to use this and other similar `quadkeyr` functions, read the the vignette: https://docs.ropensci.org/quadkeyr/articles/quadkey_to_sf_conversion.html

Value

A sf POINT data.frame with a `quadkey` column. The latitude/longitude coordinates represent the upper-left corner of the QuadKey.

References

https://learn.microsoft.com/en-us/bingmaps/articles/bing-maps-tile-system

See Also

`quadkey_to_tileXY`

`tileXY_to_pixelXY`

`pixelXY_to_latlong`

Examples

```
quadkey_to_latlong(quadkey_data = "213")
quadkey_to_latlong(quadkey_data = c("213", "212", "210"))
```

---

quadkey_to_polygon *Convert a QuadKey into a square polygon*

Description

This functions creates a `sf` POLYGON data.frame from a QuadKey string.

Usage

```
quadkey_to_polygon(quadkey)
```

Arguments

`quadkey`	The QuadKey as a string

Value

A sf POLYGON data.frame with a quadkey and geometry column.

See Also

quadkey_df_to_polygon

Examples

```
# Quadkey as string
quadkey_to_polygon(quadkey = "213")

# QuadKeys as column in a data.frame
# get data file
path <- paste0(
  system.file("extdata", package = "quadkeyr"),
  "/cityA_2020_04_15_0000.csv"
)
data <- read.csv(path)
data <- format_fb_data(data)

quadkey_df_to_polygon(data = data)
```

---

quadkey_to_tileXY *Convert a QuadKey into tile XY coordinates.*

Description

For further information, refer to the Microsoft Bing Maps Tile System documentation.

Usage

```
quadkey_to_tileXY(quadkey)
```

Arguments

quadkey	A QuadKey as a single string.

Details

Converting latitude/longitude coordinates into a QuadKey and then back to latitude/longitude won't yield identical values, unless the initial latitude/longitude coordinates correspond to the upper-left QuadKey's pixel and tile XY coordinates at the same zoom level.

Understanding this distinction is crucial for the accurate use of these functions in coordinate conversions.

For a detailed explanation on how to use this and other similar quadkeyr functions, read the the vignette: https://docs.ropensci.org/quadkeyr/articles/quadkey_to_sf_conversion.html

Value

A list returning the tile X, tile Y coordinates and the zoom level.

References

https://learn.microsoft.com/en-us/bingmaps/articles/bing-maps-tile-system

Examples

```
quadkey_to_tileXY(quadkey = "213")
```

`read_fb_mobility_files` *Read all the .csv files in a folder and format the data.*

Description

This function reads all the `.csv` files in a particular folder. These files consistently contain identical columns, with variations only in location, day, and time. As a result, we can uniformly apply specific formatting to columns across these files.

Usage

```
read_fb_mobility_files(path_to_csvs, colnames, coltypes, keep_format = NULL)
```

Arguments

`path_to_csvs`	Path to the folder where the `.csv` files are stored
`colnames`	Columns to include in the results (as character). For more information go to `readr::read_csv()`
`coltypes`	Column specifications (as strings). See vignette("readr", package = "readr") for more details. documentation.
`keep_format`	Vector of column names, besides `date_time`, `day` and `quadkey`, that you don't want to convert to a number.

Value

A data.frame with the information of all the files read.

See Also

`format_fb_data`

`read_csv`

Examples

```
files <- read_fb_mobility_files(
  path_to_csvs = paste0(system.file("extdata",
    package = "quadkeyr"
  ), "/"),
  colnames = c( # The columns not listed here will be omitted
    "lat",
    "lon",
    "quadkey",
    "date_time",
    "n_crisis",
    "percent_change",
    "day",
    "hour"
  ),
  coltypes = list(
    lat = "d",
    lon = "d",
    quadkey = "c",
    date_time = "T",
    n_crisis = "c",
    percent_change = "c",
    day = "D",
    hour = "i"
  )
)

head(files)
```

`regular_qk_grid` *Convert a incomplete QuadKey* `sf` *POINT data.frame into a regular grid.*

Description

This function completes `sf` POINT data.frame grid of QuadKeys using the bounding box of the data provided.

Usage

```
regular_qk_grid(data)
```

Arguments

`data`	A `sf` POINT data.frame

Value

A list with three elements:

- `data` A `sf` POINT data.frame, with the rows needed to complete the grid.
- `num_rows` The number of columns of the regular grid.
- `num_cols` The number of rows of the regular grid.

See Also

`create_qk_grid`

`quadkey_to_latlong`

Examples

```
quadkey_vector <- c("213", "210", "211")

qtll <- quadkey_to_latlong(quadkey = quadkey_vector)

regular_qk_grid(qtll)
```

---

tileXY_to_pixelXY *Convert tile XY coordinates into pixel XY coordinates*

Description

Converts tile XY coordinates into pixel XY coordinates of the upper-left pixel of the specified tile. For further information, refer to the Microsoft Bing Maps Tile System documentation.

Usage

```
tileXY_to_pixelXY(tileX, tileY)
```

Arguments

`tileX`	Tile X coordinate.
`tileY`	Tile Y coordinate.

Details

Converting latitude/longitude coordinates into a QuadKey and then back to latitude/longitude won't yield identical values, unless the initial latitude/longitude coordinates correspond to the upper-left Quadkey's pixel and tile XY coordinates at the same zoom level.

Understanding this distinction is crucial for the accurate use of these functions in coordinate conversions.

For a detailed explanation on how to use this and other similar `quadkeyr` functions, read the the vignette: https://docs.ropensci.org/quadkeyr/articles/quadkey_to_sf_conversion.html

Value

A list returning the pixel X and pixel Y coordinates.

References

https://learn.microsoft.com/en-us/bingmaps/articles/bing-maps-tile-system

Examples

```
tileXY_to_pixelXY(
  tileX = 3,
  tileY = 5
)
```

---

tileXY_to_quadkey *Convert tile XY coordinates into a QuadKey.*

Description

Converts tile XY coordinates into a QuadKey at a specified zoom level. For further information, refer to the Microsoft Bing Maps Tile System documentation.

Usage

```
tileXY_to_quadkey(tileX, tileY, zoom)
```

Arguments

tileX	Tile X coordinate.
tileY	Tile Y coordinate.
zoom	Zoom or level of detail, from 1 (lowest detail) to 23 (highest detail).

Details

Converting latitude/longitude coordinates into a QuadKey and then back to latitude/longitude won't yield identical values, unless the initial latitude/longitude coordinates correspond to the upper-left Quadkey's pixel and tile XY coordinates at the same zoom level.

Understanding this distinction is crucial for the accurate use of these functions in coordinate conversions.

For a detailed explanation on how to use this and other similar quadkeyr functions, read the the vignette: https://docs.ropensci.org/quadkeyr/articles/quadkey_to_sf_conversion.html

Value

The QuadKey as a string.

References

https://learn.microsoft.com/en-us/bingmaps/articles/bing-maps-tile-system

Examples

```
tileXY_to_quadkey(
  tileX = 23,
  tileY = 38,
  zoom = 6
)
```

Index

add_regular_polygon_grid, 2
apply_weekly_lag, 3

create_qk_grid, 4, *25*
create_stars_raster, 5

format_fb_data, 7, *23*

get_qk_coord, 8
get_regular_polygon_grid, 9
grid_to_polygon, 10
ground_res, 10

latlong_to_pixelXY, 11
latlong_to_quadkey, 12

mapscale, 13
mapsize, 14
missing_combinations, 15, *18*

pixelXY_to_latlong, *8*, 15, *21*
pixelXY_to_tileXY, 16
polygon_to_raster, 17

qkmap_app, 19
quadkey_df_to_polygon, 19, *20*, *22*
quadkey_to_latlong, 20, *25*
quadkey_to_polygon, 21
quadkey_to_tileXY, *8*, *21*, 22

read_csv, *23*
read_fb_mobility_files, *7*, 23
readr::read_csv(), *23*
regular_qk_grid, 24

st_as_stars, *6*, *18*
st_rasterize, *6*, *18*
stars::st_as_stars(), *5*

tileXY_to_pixelXY, *8*, *21*, 25
tileXY_to_quadkey, 26