

# Package: slippymath (via r-universe)

August 21, 2024

**Title** Slippy Map Tile Tools

**Version** 0.3.1

**Description** Provides functions for performing common tasks when working with slippy map tile service APIs e.g. Google maps, Open Street Map, Mapbox, Stamen, among others. Functionality includes converting from latitude and longitude to tile numbers, determining tile bounding boxes, and compositing tiles to a georeferenced raster image.

**Depends** R (>= 3.5.0)

**License** MIT + file LICENSE

**URL** <https://www.github.com/milesmcbain/slippymath>

**BugReports** <https://www.github.com/milesmcbain/slippymath/issues>

**Encoding** UTF-8

**LazyData** true

**Imports** raster, purrr, stats, png

**Suggests** testthat, knitr, rmarkdown

**RoxygenNote** 6.1.1

**VignetteBuilder** knitr

**Repository** <https://milesmcbain.r-universe.dev>

**RemoteUrl** <https://github.com/milesmcbain/slippymath>

**RemoteRef** master

**RemoteSha** 1e4c8bde183884d63b7db0d726138bc9ac12c26f

## Contents

bbox_tile_extent . . . . .	2
bbox_tile_query . . . . .	3
bbox_to_tile_grid . . . . .	3
compose_tile_grid . . . . .	4
lonlat_to_merc . . . . .	5

lonlat_to_tilenum . . . . .	6
merc_truncate . . . . .	7
raster_to_png . . . . .	7
tilenum_to_lonlat . . . . .	8
tile_bbox . . . . .	9
tile_grid_bboxes . . . . .	9
within_merc_extent . . . . .	10
<b>Index</b>	<b>11</b>

---

bbox_tile_extent	<i>bbox_tile_extent</i>
------------------	-------------------------

---

**Description**

Convert a bounding box from latitude and longitude to tile numbers

**Usage**

```
bbox_tile_extent(bbox, zoom)
```

**Arguments**

- `bbox`            a bbox object created by 'sf::st\_bbox', or a vector with names 'xmin', 'xmax', 'ymin', 'ymax'
- `zoom`            zoom level to calculate the tile grid on.

**Details**

This function creates an analog of a bounding box but in tile numbers. It returns the min and max x and y tile numbers for a tile grid that would fit the bounding box for a given zoom level.

**Value**

a list of 'x\_min', 'y\_min', 'x\_max', 'y\_max'

**Examples**

```
tibrogargan<- c(xmin = 152.938485, ymin = -26.93345, xmax = 152.956467,  
               ymax = -26.921463)  
bbox_tile_extent(tibrogargan, zoom = 15)
```

---

bbox_tile_query	<i>bbox_tile_query</i>
-----------------	------------------------

---

**Description**

Bounding box tile query

**Usage**

```
bbox_tile_query(bbox, zoom_levels = 2:18)
```

**Arguments**

bbox	a bbox object created by 'sf::st_bbox', or a vector with names 'xmin', 'xmax', 'ymin', 'ymax'
zoom_levels	a numeric vector of zoom levels to calculate tile usage for.

**Details**

Determines how many tiles the bounding box would occupy for a range of zooms. Useful for working out what is a reasonable zoom to work at. Each tile is a separate request from the server.

Tiles are typically 256x256 pixels and are tens of Kb in size, you can get some sense of the data from the query also.

**Value**

a data frame containing tile usage information for the bounding box at each zoom level.

**Examples**

```
tibrogargan<- c(xmin = 152.938485, ymin = -26.93345, xmax = 152.956467,
                ymax = -26.921463)

bbox_tile_query(tibrogargan)
```

---

bbox_to_tile_grid	<i>bbox_to_tile_grid</i>
-------------------	--------------------------

---

**Description**

Bounding box to tile grid

**Usage**

```
bbox_to_tile_grid(bbox, zoom = NULL, max_tiles = NULL)
```

**Arguments**

bbox	the bounding box to fit onto a grid of tiles. Must be either a 'bbox' object created with sf::st_bbox or a vector of length 4 with names: 'xmin', 'xmax', 'ymin', 'ymax'.
zoom	Optional. The desired zoom level.
max_tiles	Optional. The maximum number of tiles the grid may occupy.

**Details**

Calculate a slippy map tile grid that will fit a supplied bounding box.

The grid is returned as part of a tile\_grid object that contains a data.frame of x,y tile numbers and zoom level.

The tile grid can be calculated for a given zoom level or for the deepest zoom that ensures the number of tiles is less than or equal to 'max\_tiles'.

If 'zoom' and 'max\_tiles' are supplied together, then the max is still enforced and the function will fail if more tiles are required for the given zoom.

**Value**

a 'tile\_grid' object containing 'tiles' and 'zoom'

**Examples**

```
tibrogargan<- c(xmin = 152.938485, ymin = -26.93345, xmax = 152.956467,
                ymax = -26.921463)

## Get a grid of the minimum number of tiles for a given zoom.
bbox_to_tile_grid(tibrogargan, zoom = 15)

## get a grid of at most 12 tiles, choosing the most detailed zoom possible.
bbox_to_tile_grid(tibrogargan, max_tiles = 12)
```

---

compose\_tile\_grid      *compose\_tile\_grid*

---

**Description**

Compose a list of images using tile\_grid data.

**Usage**

```
compose_tile_grid(tile_grid, images)
```

**Arguments**

tile_grid	a tile_grid object, likely returned from ‘bbox_to_tile_grid’
images	a list of character strings defining paths to images. Matched to tiles in tile_grid based on list position.

**Details**

Given a tile\_grid object and a list of images, compose the images into a single spatially referenced RasterBrick object.

The list of images is assumed to be in corresponding order to the tiles in the tile\_grid object.

The returned object uses the Web Mercator projection, EPSG:3857, which is the native crs of the tiles.

**Value**

a spatially referenced raster.

---

lonlat_to_merc	<i>Transform between spherical Mercator and longitude/latitude</i>
----------------	--

---

**Description**

Transform between spherical Mercator and longitude/latitude

**Usage**

```
lonlat_to_merc(ll)
```

```
merc_to_lonlat(xy)
```

**Arguments**

ll	matrix of longitude / latitude
xy	matrix of x / y Mercator

**Value**

matrix of coordinates transformed forward or inverse

**Examples**

```

uluru_lonlat <- matrix(c(131.0325162,
                        -25.3448562),
                      nrow = 1)

lonlat_to_merc(uluru_lonlat)

uluru_merc <- matrix(c(14586472.958481,
                      -2918162.223463),
                    nrow = 1)

merc_to_lonlat(uluru_merc)

```

---

lonlat_to_tilenum	<i>lonlat_to_tilenum</i>
-------------------	--------------------------

---

**Description**

Convert longitude and latitude to slippy tile numbers

**Usage**

```
lonlat_to_tilenum(lon_deg, lat_deg, zoom)
```

**Arguments**

lon_deg	degrees longitude for point
lat_deg	degrees latitude for point
zoom	zoom level for tile calculation. Increasing zoom increases the number of tiles.

**Details**

Returns the Open Street Map slippy map tile numbers (x, y) the supplied latitude and longitude fall on, for a given zoom level.

The point specified by `lon_deg` and `lat_deg` is assumed to be in EPSG:4326 coordinate reference system.

**Value**

a list containing 'x' and 'y' - the tile numbers.

**Examples**

```

lonlat_to_tilenum(
  lon = 13.37771496361961,
  lat = 52.51628011262304,
  zoom = 17
)

```

---

merc_truncate	<i>merc_truncate</i>
---------------	----------------------

---

**Description**

Truncate coordinate to Mercator extent.

**Usage**

```
merc_truncate(xy)
```

**Arguments**

xy                    a matrix of Mercator XY points.

**Details**

If a point in m lies outside the Mercator extent, this function can be used to truncate it to the boundary of the extent.

**Value**

a matrix of XY points.

**Examples**

```
stray <- matrix(c(20037509,
                  -2918162.223463),
                nrow = 1)

merc_truncate(stray)
```

---

raster_to_png	<i>raster_to_png</i>
---------------	----------------------

---

**Description**

Write a raster to PNG

**Usage**

```
raster_to_png(tile_raster, file_path)
```

**Arguments**

tile\_raster        the raster to write to PNG  
file\_path         the path to write the raster

**Details**

This function is a convenience wrapper for writing rasters in PNG format.

**Value**

nothing.

---

tilenum_to_lonlat	<i>tilenum_to_lonlat</i>
-------------------	--------------------------

---

**Description**

Convert slippy map tiles numbers to latitude and longitude

**Usage**

```
tilenum_to_lonlat(x, y, zoom)
```

**Arguments**

x	slippy map tile number in x domain (left to right)
y	slippy map tile number in y domain (top to bottom)
zoom	the zoom level for the calculation. Increasing zoom increases the number of tiles.

**Details**

Returns the latitude and longitude of the top left corner of a slippy map tile specified by 'x', 'y' for a given zoom level.

**Value**

a list containing 'lat' and 'lon' - latitude and longitude.

**Examples**

```
tilenum_to_lonlat(  
  x = 70406,  
  y = 42987,  
  zoom = 17  
)
```



---

tile_bbox	<i>tile_bbox</i>
-----------	------------------

---

### Description

Calculate the bounding box for a tile in latitude and longitude

### Usage

```
tile_bbox(x, y, zoom)
```

### Arguments

x	slippy map tile x number
y	slippy map tile y number
zoom	zoom level for tile

### Details

Given a slippy maps tile specified by ‘x’, ‘y’, and ‘zoom’, return the an ‘sf’ bounding box object for the tile with units in metres using the EPSG:3857 coordinate reference system (Web Mercator).

### Value

an sf bbox object.

### Examples

```
## return an sf style bbox object in with epsg and proj4string  
tile_bbox(x = 30304, y = 18929, zoom = 15)
```

---

tile_grid_bboxes	<i>tile_grid_bboxes</i>
------------------	-------------------------

---

### Description

Get tile grid bounding boxes

### Usage

```
tile_grid_bboxes(tile_grid)
```

### Arguments

tile_grid	a tile_grid object, likely returned from ‘bbox_to_tile_grid’
-----------	--

**Details**

Given an `tile_grid` object like that returned from `'bbox_to_tile_grid'`, return a list of sf style bounding box objects, one for each tile in the grid, in the same order as tiles in `'tile_grid$tiles'`.

The bounding box units are metres in the EPSG:3857 coordinate reference system (Web Mercator).

**Value**

a list of sf bounding box objects in the corresponding order to the tiles in `'tile_grid'`

**Examples**

```
tibrogargan<- c(xmin = 152.938485, ymin = -26.93345, xmax = 152.956467,
               ymax = -26.921463)

tibrogargan_grid <- bbox_to_tile_grid(tibrogargan, zoom = 15)

tile_grid_bboxes(tibrogargan_grid)
```

---

<code>within_merc_extent</code>	<i>within_mercator_extent</i>
---------------------------------	-------------------------------

---

**Description**

Are points in meters within Mercator extent?

**Usage**

```
within_merc_extent(xy)
```

**Arguments**

`xy` a matrix of Mercator xy coordinates.

**Details**

When doing maths with Mercator coordinates in m, you can end up outside the Mercator extent with an undefined coordinate. This function returns true if all xy lie within the Mercator extent.

**Value**

TRUE or FALSE

**Examples**

```
stray <- matrix(c(20037509,
                 -2918162.223463),
               nrow = 1)

within_merc_extent(stray)
```

# Index

`bbox_tile_extent`, [2](#)  
`bbox_tile_query`, [3](#)  
`bbox_to_tile_grid`, [3](#)  
  
`compose_tile_grid`, [4](#)  
  
`lonlat_to_merc`, [5](#)  
`lonlat_to_tilenum`, [6](#)  
  
`merc_to_lonlat(lonlat_to_merc)`, [5](#)  
`merc_truncate`, [7](#)  
  
`raster_to_png`, [7](#)  
  
`tile_bbox`, [9](#)  
`tile_grid_bboxes`, [9](#)  
`tilenum_to_lonlat`, [8](#)  
  
`within_merc_extent`, [10](#)