

Package: tilemath (via r-universe)

October 15, 2026

Title Tile Math for Map Tile Pyramids

Version 0.0.0.9000

Description Pure tile math for map tile pyramids. Provides coordinate conversions between longitude/latitude, Mercator metres, pixel coordinates, and tile numbers; quadkey encoding and decoding; TMS y-axis flipping; URL templating; and convenience helpers for selecting tiles from an extent or a point with automatic zoom selection. The default pyramid is web Mercator but the internal structure leaves room for longlat and arbitrary raster pyramids.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

URL <https://github.com/hypertidy/tilemath>

BugReports <https://github.com/hypertidy/tilemath/issues>

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

Date/Publication 2026-04-10 04:11:50 UTC

RemoteUrl <https://github.com/hypertidy/tilemath>

RemoteRef HEAD

RemoteSha fb938c619f671b30c9f428268eb33a2c8be2d928

Contents

lonlat_to_merc	2
lonlat_to_pixelxy	3
lonlat_to_tilenum	3
quadkey_parent	4

quadkey_to_tilenum	5
tile_extent	5
tile_url	6
tilenum_to_lonlat	7
tilenum_to_quadkey	7
tiles_at	8
tiles_near	8
xyz_to_tms	9
Index	11

lonlat_to_merc	<i>Spherical Mercator projection</i>
----------------	--------------------------------------

Description

Convert between longitude/latitude (degrees, WGS84) and spherical Mercator metres (EPSG:3857).

Usage

```
lonlat_to_merc(lon, lat)

merc_to_lonlat(x, y)
```

Arguments

- lon, lat numeric vectors of longitude and latitude in degrees
- x, y numeric vectors of Mercator coordinates in metres

Details

Latitudes are clipped to the Mercator limit of approximately +/-85.05113 degrees.

Value

a data.frame with two columns

Examples

```
lonlat_to_merc(147.33, -42.88)
merc_to_lonlat(16395637, -5300240)
```

lonlat_to_pixelxy	<i>Convert longitude/latitude to global pixel coordinates</i>
-------------------	---

Description

Returns the (px, py) pixel coordinates in the global pixel space at a given zoom level. The global pixel space is $2^{\text{zoom}} * \text{tile_size}$ pixels on each side.

Usage

```
lonlat_to_pixelxy(lon, lat, zoom, tile_size = 256)
```

```
pixelxy_to_lonlat(px, py, zoom, tile_size = 256)
```

Arguments

lon, lat	numeric vectors of longitude and latitude in degrees
zoom	integer zoom level
tile_size	pixels per tile side
px, py	numeric vectors of global pixel coordinates

Value

a data.frame with columns px and py

Examples

```
lonlat_to_pixelxy(147.33, -42.88, 12)
```

lonlat_to_tilenum	<i>Convert longitude/latitude to tile numbers</i>
-------------------	---

Description

Returns the (x, y) tile coordinates containing the given lon/lat at a given zoom level. Uses the standard OSM/XYZ convention where (0, 0) is the top-left tile.

Usage

```
lonlat_to_tilenum(lon, lat, zoom)
```

Arguments

lon, lat	numeric vectors of longitude and latitude in degrees
zoom	integer zoom level

Value

a data.frame with integer columns x and y

Examples

```
lonlat_to_tilenum(147.33, -42.88, 12)
```

quadkey_parent	<i>Quadkey tree operations</i>
----------------	--------------------------------

Description

quadkey_parent() returns the quadkey of the containing tile at one lower zoom level. quadkey_children() returns the four child quadkeys at one deeper zoom level. quadkey_siblings() returns the three other quadkeys sharing the same parent.

Usage

```
quadkey_parent(quadkey)
```

```
quadkey_children(quadkey)
```

```
quadkey_siblings(quadkey)
```

Arguments

quadkey character vector of quadkeys

Value

for quadkey_parent(), a character vector. For quadkey_children() and quadkey_siblings(), a list of character vectors, one per input quadkey.

Examples

```
quadkey_parent("1203")
quadkey_children("12")
quadkey_siblings("1203")
```

quadkey_to_tilenum	<i>Convert a quadkey to tile numbers</i>
--------------------	--

Description

Decode a Bing Maps quadkey string back to (x, y, zoom). Vectorized.

Usage

```
quadkey_to_tilenum(quadkey)
```

Arguments

quadkey	character vector of quadkeys
---------	------------------------------

Value

a data.frame with integer columns x, y, zoom

Examples

```
quadkey_to_tilenum("213")  
quadkey_to_tilenum(c("120", "121", "122", "123"))
```

tile_extent	<i>Tile extent</i>
-------------	--------------------

Description

Compute the bounding extent of one or more tiles. Vectorized over x, y, and zoom.

Usage

```
tile_extent(x, y, zoom, units = c("native", "lonlat"))
```

Arguments

x, y	integer vectors of tile coordinates
zoom	integer zoom level
units	one of "native" or "lonlat"

Details

For the Mercator pyramid, "native" units are Mercator metres (EPSG:3857). The "lonlat" units are the inverse projection of the tile corners to longitude and latitude in degrees.

Value

a data.frame with columns xmin, xmax, ymin, ymax

Examples

```
tile_extent(3724, 2589, 12)
tile_extent(3724, 2589, 12, units = "lonlat")
```

tile_url	<i>Format tile URLs from a template</i>
----------	---

Description

Substitute tile coordinates into a URL or path template. Vectorized over x, y, and zoom. Recognised placeholders are {x}, {y}, {z}, {-y} for the TMS y-flipped value, and {q} for the quadkey.

Usage

```
tile_url(x, y, zoom, template)
```

Arguments

- x, y integer vectors of tile coordinates (XYZ convention)
- zoom integer zoom level
- template a character scalar template string

Value

a character vector of formatted URLs

Examples

```
tile_url(3698, 2518, 12,
        "https://tile.openstreetmap.org/{z}/{x}/{y}.png")
tile_url(3, 5, 3,
        "https://ecn.t0.tiles.virtualearth.net/tiles/a{q}.jpeg?g=587")
```

tilenum_to_lonlat	<i>Convert tile numbers to longitude/latitude</i>
-------------------	---

Description

Returns the longitude/latitude of the top-left (north-west) corner of the given tile.

Usage

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

Arguments

x, y	integer vectors of tile coordinates
zoom	integer zoom level

Value

a data.frame with columns lon and lat

Examples

```
tilenum_to_lonlat(3724, 2589, 12)
```

tilenum_to_quadkey	<i>Convert tile numbers to a quadkey</i>
--------------------	--

Description

Encode tile (x, y, zoom) as a Bing Maps quadkey string. Vectorized.

Usage

```
tilenum_to_quadkey(x, y, zoom)
```

Arguments

x, y	integer vectors of tile coordinates
zoom	integer zoom level (length 1 or matching x and y)

Value

a character vector of quadkeys

Examples

```
tilenum_to_quadkey(3, 5, 3)
tilenum_to_quadkey(c(3698, 3699), c(2518, 2518), 12)
```

tiles_at	<i>Tiles covering an extent</i>
----------	---------------------------------

Description

Returns the tiles covering a given longitude/latitude extent. If zoom is NULL, picks the deepest zoom level for which the number of tiles is at most max_tiles. If zoom is supplied, uses it directly and max_tiles is ignored.

Usage

```
tiles_at(extent, max_tiles = 16, zoom = NULL, tile_size = 256)
```

Arguments

extent	numeric vector of length 4 in lonlat: c(xmin, xmax, ymin, ymax)
max_tiles	maximum number of tiles allowed when auto-picking zoom
zoom	optional integer zoom level. If supplied, overrides automatic zoom selection.
tile_size	pixels per tile side (default 256)

Details

The extent is expanded ("snapped") to the union of the tile extents. The snapped extent in native (Mercator) units is attached as the extent_snapped attribute on the returned data.frame.

Value

a data.frame with one row per tile and columns x, y, zoom, xmin, xmax, ymin, ymax (the last four in Mercator metres). The extent_snapped attribute holds the union extent.

Examples

```
tiles_at(c(147.2, 147.5, -43.0, -42.7), max_tiles = 16)
tiles_at(c(147.2, 147.5, -43.0, -42.7), zoom = 12)
```

tiles_near	<i>Tiles around a point</i>
------------	-----------------------------

Description

Returns the tiles covering a buffered region around a longitude/ latitude point. The buffer is applied in Mercator metres around the projected point, then converted back to a lonlat extent and passed to [tiles_at\(\)](#).

Usage

```

tiles_near(
  lon,
  lat,
  buffer = 5000,
  max_tiles = 16,
  zoom = NULL,
  tile_size = 256
)

```

Arguments

lon, lat	numeric scalars of the point coordinates in degrees
buffer	half-width of the region in Mercator metres
max_tiles	maximum number of tiles allowed when auto-picking zoom
zoom	optional integer zoom level
tile_size	pixels per tile side (default 256)

Details

Note that "metres" here means Mercator metres, not great-circle metres on the ellipsoid. At high latitudes the two diverge, but for tile selection the difference is rarely material.

Value

as for `tiles_at()`

Examples

```

tiles_near(147.33, -42.88, buffer = 20000)
tiles_near(147.33, -42.88, buffer = 20000, zoom = 12)

```

xyz_to_tms

Flip tile y between XYZ and TMS conventions

Description

XYZ tile schemes (OSM, Google, Mapbox) place $y = 0$ at the top of the grid. TMS places $y = 0$ at the bottom. These functions flip a y value between the two conventions. They are mathematically identical but named for clarity.

Usage

```

xyz_to_tms(y, zoom)

tms_to_xyz(y, zoom)

```

Arguments

<code>y</code>	integer vector of tile y coordinates
<code>zoom</code>	integer zoom level

Value

integer vector of flipped y coordinates

Examples

```
xyz_to_tms(2518, 12)
tms_to_xyz(1577, 12)
```

Index

`lonlat_to_merc`, [2](#)
`lonlat_to_pixelxy`, [3](#)
`lonlat_to_tilenum`, [3](#)

`merc_to_lonlat (lonlat_to_merc)`, [2](#)

`pixelxy_to_lonlat (lonlat_to_pixelxy)`, [3](#)

`quadkey_children (quadkey_parent)`, [4](#)
`quadkey_parent`, [4](#)
`quadkey_siblings (quadkey_parent)`, [4](#)
`quadkey_to_tilenum`, [5](#)

`tile_extent`, [5](#)
`tile_url`, [6](#)
`tilenum_to_lonlat`, [7](#)
`tilenum_to_quadkey`, [7](#)
`tiles_at`, [8](#)
`tiles_at()`, [8](#), [9](#)
`tiles_near`, [8](#)
`tms_to_xyz (xyz_to_tms)`, [9](#)

`xyz_to_tms`, [9](#)