

Package ‘a5R’

September 8, 2026

Title 'A5' Discrete Global Grid System

Version 0.6.0

Description Bindings for the ``A5 geospatial index"

<<https://a5geo.org/>>. 'A5' partitions the Earth's surface into pentagonal cells across 31 resolution levels using an equal-area projection onto a dodecahedron. Provides functions for indexing coordinates to cells, traversing the cell hierarchy, computing cell boundaries, and compacting/uncompacting cell sets. Powered by the 'A5' 'Rust' crate via 'extendr'.

License Apache License (>= 2)

URL <https://github.com/belian-earth/a5R>,
<https://belian-earth.github.io/a5R/>

BugReports <https://github.com/belian-earth/a5R/issues>

Depends R (>= 4.2)

Imports cli, rlang (>= 1.1.0), units, vctrs (>= 0.6.0), wk (>= 0.9.0)

Suggests arrow, knitr, pillar, rmarkdown, sf, terra, testthat (>= 3.0.0), tibble, withr

VignetteBuilder knitr

Config/testthat/edition 3

Config/rextendr/version 0.5.0

SystemRequirements Cargo (Rust's package manager), rustc

Encoding UTF-8

RoxygenNote 8.0.0

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Hugh Graham [aut, cre],
belian.earth [cph]

Maintainer Hugh Graham <hugh@belian.earth>

Repository CRAN

Date/Publication 2026-09-08 11:50:15 UTC

Contents

a5_cell	2
a5_cell_area	3
a5_cell_distance	4
a5_cell_edge_length_avg	5
a5_cell_from_arrow	5
a5_cell_to_boundary	6
a5_cell_to_children	7
a5_cell_to_lonlat	8
a5_cell_to_parent	9
a5_compact	9
a5_get_num_cells	10
a5_get_num_children	11
a5_get_res0_cells	11
a5_get_resolution	12
a5_grid_disk	12
a5_linestring_to_cells	13
a5_lonlat_to_cell	14
a5_polygon_to_cells	15
a5_set_threads	16
a5_spherical_cap	17
a5_u64_to_hex	17
a5_uncompact	18
wk_methods	19
Index	20

a5_cell	<i>A5 Cell Index Vector</i>
---------	-----------------------------

Description

Create, test, and coerce A5 cell index vectors. Cells are stored as a record with eight raw-byte fields (b1–b8) representing the little-endian bytes of the u64 cell ID. This avoids the precision loss of floating-point storage and keeps memory compact.

Usage

```
a5_cell(x = character())

is_a5_cell(x)

as_a5_cell(x)

a5_is_valid(x)
```

Arguments

`x` A character vector of hex-encoded A5 cell IDs, or an object coercible to one.

Value

An `a5_cell` vector (`a5_cell`, `as_a5_cell`), a logical scalar (`is_a5_cell`), or a logical vector (`a5_is_valid`).

Examples

```
cells <- a5_cell(c("0800000000000006", "0800000000000016"))
cells
a5_is_valid(c("0800000000000006", "not_a_cell", NA))
```

a5_cell_area	<i>Cell area at a given resolution</i>
--------------	--

Description

Returns the area of a single cell in square metres at the given resolution(s). Because A5 is an equal-area DGGS, all cells at the same resolution have identical area.

Usage

```
a5_cell_area(resolution, units = "m^2")
```

Arguments

`resolution` Integer vector of resolutions (0–30).

`units` Character scalar specifying the output area unit (default "m^2"). Any unit convertible from m^2 via `units::set_units()` is accepted (e.g. "km^2", "ha", "acre"). If NULL, the area is returned as a numeric vector in m^2.

Value

A `units::units` vector of areas.

Examples

```
a5_cell_area(0:5)
a5_cell_area(5, units = "km^2")
```

a5_cell_distance	<i>Distance between cell centroids</i>
------------------	--

Description

Computes the distance between the centroids of pairs of A5 cells using the specified method.

Usage

```
a5_cell_distance(
  from,
  to,
  units = "m",
  method = c("haversine", "geodesic", "rhumb")
)
```

Arguments

from, to	a5_cell vectors (recycled to common length).
units	Character scalar specifying the distance unit (default "m"). Any unit convertible from metres via units::set_units() is accepted (e.g. "km", "mi"). If NULL, the distance is returned as a numeric vector in metres.
method	Distance calculation method. One of "haversine" (great-circle, default), "geodesic" (WGS84 ellipsoid via Karney 2013), or "rhumb" (loxodrome / constant-bearing).

Value

A [units::units](#) vector of distances.

See Also

[a5_cell_to_lonlat\(\)](#) for cell centroids, [a5_cell_area\(\)](#) for cell areas.

Examples

```
a <- a5_lonlat_to_cell(-3.19, 55.95, resolution = 24)
b <- a5_lonlat_to_cell(-3.10, 55.90, resolution = 24)
a5_cell_distance(a, b)
a5_cell_distance(a, b, units = "km")
a5_cell_distance(a, b, method = "geodesic")
```

a5_cell_edge_length_avg

Average cell edge length at a given resolution

Description

Returns the average length of a cell edge at the given resolution(s). Individual edge lengths vary from this average by roughly +/-10%, depending on the cell's shape and its position on the globe. Use this for a quick estimate of cell size when choosing a resolution; use [a5_cell_to_boundary\(\)](#) to measure a specific cell.

Usage

```
a5_cell_edge_length_avg(resolution, units = "m")
```

Arguments

resolution	Integer vector of resolutions (0–30).
units	Character scalar specifying the output length unit (default "m"). Any unit convertible from m via units::set_units() is accepted (e.g. "km", "mi"). If NULL, the length is returned as a numeric vector in metres.

Value

A [units::units](#) vector of average edge lengths, or a numeric vector if units = NULL.

See Also

[a5_cell_area\(\)](#)

Examples

```
a5_cell_edge_length_avg(0:5)
a5_cell_edge_length_avg(10, units = "km")
```

a5_cell_from_arrow

Convert between a5_cell and Arrow uint64 arrays

Description

Losslessly convert between [a5_cell](#) vectors and Arrow uint64 arrays. This avoids the precision loss that occurs when Arrow converts uint64 to R's double (which can only represent integers exactly up to 2⁵³, while A5 cell IDs span the full 0–2⁶⁴ range).

Usage

```
a5_cell_from_arrow(x)
```

```
a5_cell_to_arrow(x)
```

Arguments

x For `a5_cell_from_arrow()`, an Arrow Array or ChunkedArray of type uint64. For `a5_cell_to_arrow()`, an [a5_cell](#) vector.

Details

Internally these use Arrow's zero-copy `View()` to reinterpret uint64 bytes as `fixed_size_binary(8)`, then convert to/from the raw-byte representation used by [a5_cell](#). The resulting Arrow arrays can be written directly to Parquet and read correctly by DuckDB, Python, and other Arrow-compatible tools.

Value

`a5_cell_from_arrow()` returns an [a5_cell](#) vector. `a5_cell_to_arrow()` returns an Arrow Array of type uint64.

See Also

[a5_u64_to_hex\(\)](#) for converting to hex strings instead.

Examples

```
cell <- a5_lonlat_to_cell(135, 0, resolution = 10)
arr <- a5_cell_to_arrow(cell)
back <- a5_cell_from_arrow(arr)
identical(format(cell), format(back))
```

```
cells <- a5_lonlat_to_cell(c(-3.19, 135), c(55.95, 0), resolution = 10)
arr <- a5_cell_to_arrow(cells)
arr$type$ToString()
```

`a5_cell_to_boundary` *Get cell boundary polygons*

Description

Returns the boundary of each cell as a `wk::wkt()` or `wk::wkb()` polygon geometry. Boundaries are pentagonal polygons on the WGS 84 ellipsoid.

Usage

```
a5_cell_to_boundary(
  cell,
  format = c("wkb", "wkt"),
  closed = TRUE,
  segments = NULL
)
```

Arguments

cell	An a5_cell vector.
format	Character scalar, either "wkb" (default) or "wkt".
closed	Logical scalar; if TRUE (default) the ring is closed (first vertex repeated at end).
segments	Integer scalar or NULL. Number of interpolation segments per edge for geodesic accuracy. NULL uses the default (straight edges).

Value

A wk_wkt or wk_wkb vector of polygon geometries with wk: :wk_crs_longlat() CRS.

See Also

[a5_cell_to_lonlat\(\)](#) for cell centroids.

Examples

```
cell <- a5_lonlat_to_cell(-3.19, 55.95, resolution = 5)
a5_cell_to_boundary(cell)
a5_cell_to_boundary(cell, format = "wkt")
```

a5_cell_to_children	<i>Get child cells</i>
---------------------	------------------------

Description

Returns the child cells of a single cell. By default returns the 4 immediate children (one resolution finer). Optionally target a specific finer resolution.

Usage

```
a5_cell_to_children(cell, resolution = NULL)
```

Arguments

cell	A single a5_cell value.
resolution	Integer scalar target child resolution, or NULL for immediate children.

Value

An [a5_cell](#) vector of child cells.

See Also

[a5_cell_to_parent\(\)](#), [a5_get_resolution\(\)](#)

Examples

```
cell <- a5_lonlat_to_cell(-3.19, 55.95, resolution = 5)
a5_cell_to_children(cell)
```

a5_cell_to_lonlat	<i>Convert A5 cell indices to coordinates</i>
-------------------	---

Description

Returns the centre-point longitude and latitude of each cell.

Usage

```
a5_cell_to_lonlat(cell, as_dataframe = FALSE)
```

Arguments

cell	An a5_cell vector (or character coercible to one).
as_dataframe	Logical scalar controlling the return container. When FALSE (default), centroids are returned as a wk::xy() vector with WGS 84 CRS, the geographic-typed form that plugs into wk/sf pipelines. When TRUE, centroids are returned as a base <code>data.frame</code> with columns <code>lon</code> and <code>lat</code> .

Value

A [wk::xy\(\)](#) vector (if `as_dataframe = FALSE`) or a `data.frame` with columns `lon` and `lat` (if `as_dataframe = TRUE`).

See Also

[a5_lonlat_to_cell\(\)](#) for the inverse operation, [a5_cell_to_boundary\(\)](#) for full cell polygons.

Examples

```
cell <- a5_lonlat_to_cell(-3.19, 55.95, resolution = 5)
a5_cell_to_lonlat(cell)

# Data frame output
cell2 <- a5_lonlat_to_cell(114.8, 4.1, resolution = 5)
a5_cell_to_lonlat(cell2, as_dataframe = TRUE)
```

a5_cell_to_parent	<i>Navigate to parent cell(s)</i>
-------------------	-----------------------------------

Description

Returns the parent cell of each input cell. By default returns the immediate parent (one resolution coarser). Optionally target a specific coarser resolution.

Usage

```
a5_cell_to_parent(cell, resolution = NULL)
```

Arguments

cell	An a5_cell vector.
resolution	Integer scalar target parent resolution, or NULL for the immediate parent.

Value

An [a5_cell](#) vector of parent cells.

See Also

[a5_cell_to_children\(\)](#), [a5_get_resolution\(\)](#)

Examples

```
cell <- a5_lonlat_to_cell(-3.19, 55.95, resolution = 10)
a5_cell_to_parent(cell)
a5_cell_to_parent(cell, resolution = 5)
```

a5_compact	<i>Compact a set of A5 cells</i>
------------	----------------------------------

Description

Merges complete sibling groups into their common parent, reducing the number of cells while preserving coverage.

Usage

```
a5_compact(cells)
```

Arguments

cells	An a5_cell vector.
-------	------------------------------------

Value

An [a5_cell](#) vector of compacted cells.

See Also

[a5_uncompact\(\)](#)

Examples

```
cell <- a5_lonlat_to_cell(-3.19, 55.95, resolution = 5)
children <- a5_cell_to_children(cell)
a5_compact(children) # back to the parent
```

a5_get_num_cells	<i>Total number of cells at a given resolution</i>
------------------	--

Description

Total number of cells at a given resolution

Usage

```
a5_get_num_cells(resolution)
```

Arguments

resolution Integer scalar resolution (0–30).

Value

A numeric scalar (double) giving the total count. Returned as double because the count can exceed R's integer range.

Examples

```
a5_get_num_cells(0)
a5_get_num_cells(10)
```

a5_get_num_children	<i>Number of children between two resolutions</i>
---------------------	---

Description

Returns the number of child cells each parent cell contains when expanding from one resolution to another.

Usage

```
a5_get_num_children(parent_resolution, child_resolution)
```

Arguments

parent_resolution

Integer scalar (0–30).

child_resolution

Integer scalar (0–30), must be $\geq$ parent_resolution.

Value

A numeric scalar. Returned as double because the count can exceed R's integer range at large resolution deltas.

See Also

[a5_get_num_cells\(\)](#), [a5_cell_to_children\(\)](#), [a5_uncompact\(\)](#)

Examples

```
a5_get_num_children(5, 8) # 4^3 = 64
a5_get_num_children(0, 5)
```

a5_get_res0_cells	<i>Get all resolution-0 root cells</i>
-------------------	--

Description

Returns the 12 root cells corresponding to the 12 faces of the dodecahedron.

Usage

```
a5_get_res0_cells()
```

Value

An [a5_cell](#) vector of length 12.

Examples

```
a5_get_res0_cells()
```

a5_get_resolution	<i>Get the resolution of A5 cell indices</i>
-------------------	--

Description

Extracts the resolution level (0–30) encoded in each cell index.

Usage

```
a5_get_resolution(cell)
```

Arguments

cell An [a5_cell](#) vector.

Value

An integer vector of resolutions.

See Also

[a5_cell_to_parent\(\)](#), [a5_cell_to_children\(\)](#)

Examples

```
cell <- a5_lonlat_to_cell(-3.19, 55.95, resolution = 10)
a5_get_resolution(cell)
```

a5_grid_disk	<i>Cells within k hops of a cell</i>
--------------	--------------------------------------

Description

Returns all cells reachable within k edge hops of a centre cell, including the centre cell itself.

Usage

```
a5_grid_disk(cell, k, vertex = FALSE)
```

Arguments

cell	A single a5_cell value.
k	Integer scalar, number of hops.
vertex	Logical scalar. If FALSE (default), only edge-sharing neighbours (4-connected) are traversed. If TRUE, vertex-sharing neighbours are included (8-connected).

Value

A compacted [a5_cell](#) vector.

See Also

[a5_spherical_cap\(\)](#) for distance-based selection.

Examples

```
cell <- a5_lonlat_to_cell(-3.19, 55.95, resolution = 8)
a5_grid_disk(cell, k = 1)
```

a5_linestring_to_cells

Cells traced by a great-circle linestring

Description

Returns the A5 cells at resolution whose pentagons are intersected by the great-circle polyline connecting the supplied waypoints. Output is uncompact, in discovery order along the path, with duplicates removed first-seen. Multi-feature inputs (a MULTILINESTRING or an sfc of multiple linestrings) are handled natively: per-feature cell sequences are concatenated in feature order with first-seen deduplication.

Usage

```
a5_linestring_to_cells(x, resolution)
```

Arguments

x	A linestring-like geometry. One of: - Any geometry handleable by wk::wk_handle() containing one or more LINESTRING / MULTILINESTRING features. - A SpatVector of linestrings (requires the terra package). - A two-column numeric matrix (cbind(lon, lat)) of waypoints. - A data.frame with columns lon and lat.
resolution	Integer scalar target resolution (0–30).

Details

Consecutive waypoints are connected by great-circle arcs (not rhumb lines or planar segments), so antimeridian-crossing paths work transparently when written in unwrapped lon/lat.

Matrix and `data.frame` inputs are treated as a single linestring; for multi-feature data, pass an `sf`, `sfc`, `wk`, or `SpatVector` geometry instead.

Value

An `a5_cell` vector at resolution.

See Also

`a5_polygon_to_cells()`, `a5_grid_disk()`.

Examples

```
line <- wk::wkt("LINESTRING (2.35 48.86, -0.13 51.51)")
cells <- a5_linestring_to_cells(line, resolution = 6)
length(cells)
```

a5_lonlat_to_cell	<i>Convert coordinates to A5 cell indices</i>
-------------------	---

Description

Maps longitude/latitude coordinates to A5 cell indices at the specified resolution.

Usage

```
a5_lonlat_to_cell(lon, lat, resolution)
```

Arguments

lon	Numeric vector of longitudes in degrees.
lat	Numeric vector of latitudes in degrees.
resolution	Integer scalar or vector of resolutions (0–30).

Value

An `a5_cell` vector of cell indices.

See Also

`a5_cell_to_lonlat()` for the inverse operation.

Examples

```
a5_lonlat_to_cell(-3.19, 55.95, resolution = 5)
```

a5_polygon_to_cells *Cells inside or overlapping a polygon*

Description

Returns A5 cells at resolution that fall inside the polygon. By default a cell is included when its centre lies inside the polygon; set `containment = "overlapping"` to also include every cell that touches the polygon boundary, giving gap-free coverage. Multi-feature inputs (a MULTIPOLYGON, an sfc of multiple polygons, or a POLYGON with holes) are handled natively: per polygon part, the outer ring and its holes are converted together with hole interiors excluded, then the results are unioned across parts. The final cell set is compacted; use `a5_uncompact()` to expand to a uniform-resolution grid.

Usage

```
a5_polygon_to_cells(x, resolution, containment = c("centre", "overlapping"))
```

Arguments

x	A polygon-like geometry. One of: - Any geometry handleable by <code>wk::wk_handle()</code> (e.g. <code>wk::wkt()</code>, <code>wk::wkb()</code>, <code>wk::rct()</code>, <code>sf</code>, <code>sfc</code>) containing one or more POLYGON / MULTIPOLYGON features. - A <code>SpatVector</code> of polygons (requires the <code>terra</code> package). - A two-column numeric matrix (<code>cbind(lon, lat)</code>) of vertices, interpreted as a single outer ring. - A <code>data.frame</code> with columns <code>lon</code> and <code>lat</code>, interpreted as a single outer ring.
resolution	Integer scalar target resolution (0-30).
containment	Character scalar selecting which cells to include. <code>"centre"</code> (the default) includes a cell when its centre lies inside the polygon. <code>"overlapping"</code> additionally includes every cell that overlaps the polygon boundary, so the result fully covers the polygon. The overlapping set is a superset of the centre set.

Details

With `containment = "centre"`, membership is determined by **centre-point containment**: a cell is included if its centroid lies inside the polygon, with hole interiors excluded. Cells straddling the boundary whose centre falls outside are dropped, so the union of the cells does not fully cover the polygon.

With `containment = "overlapping"`, every cell that contains any point of the polygon boundary is kept as well, so the returned cells cover the polygon without gaps. Hole boundaries count as boundary, so cells straddling a hole edge are included while the hole interior is still excluded.

Coordinates must be WGS 84 longitude/latitude in degrees. Rings are closed automatically; a trailing duplicate vertex is dropped if present.

Where no A5 cell centroids at the specified resolution fall within the geometry, an empty `a5_cell` vector is returned.

Matrix and `data.frame` inputs are treated as a single ring; for multi-feature data or polygons with holes, pass an `sf`, `sfc`, `wk`, or `SpatVector` geometry instead.

Value

An `a5_cell` vector at or coarser than resolution.

See Also

`a5_linestring_to_cells()`, `a5_uncompact()`.

Examples

```
poly <- wk::wkt(
  "POLYGON ((-3.3 55.9, -3.1 55.9, -3.1 56, -3.3 56, -3.3 55.9))"
)
cells <- a5_polygon_to_cells(poly, resolution = 8)
length(cells)

# Gap-free coverage: every cell touching the polygon
covering <- a5_polygon_to_cells(poly, resolution = 8,
                               containment = "overlapping")
length(covering)
```

`a5_set_threads`

Set the number of threads used by a5R

Description

Controls the number of threads used for parallel processing in vectorised functions. Set to 1 (the default) for sequential processing with zero overhead, or higher for parallel execution via rayon.

Usage

```
a5_set_threads(n = 1L)
```

```
a5_get_threads()
```

Arguments

`n` Integer scalar. Number of threads. Must be ≥ 1 .

Value

Invisibly returns the previous thread count.

Integer scalar.

a5_spherical_cap	<i>Cells within a great-circle radius</i>
------------------	---

Description

Returns all cells whose centres fall within a great-circle distance of a given cell's centre.

Usage

```
a5_spherical_cap(cell, radius)
```

Arguments

cell	A single a5_cell value.
radius	Numeric scalar, great-circle radius in metres.

Value

A compacted [a5_cell](#) vector.

See Also

[a5_grid_disk\(\)](#) for hop-based selection.

Examples

```
cell <- a5_lonlat_to_cell(-3.19, 55.95, resolution = 8)
a5_spherical_cap(cell, radius = 1000)
```

a5_u64_to_hex	<i>Coerce between hex strings and A5 cell vectors</i>
---------------	---

Description

`a5_u64_to_hex()` converts an [a5_cell](#) vector to 16-character zero-padded hex strings. `a5_hex_to_u64()` converts hex strings to an [a5_cell](#) vector.

Usage

```
a5_u64_to_hex(x)
```

```
a5_hex_to_u64(x)
```

Arguments

x	For <code>a5_u64_to_hex()</code> , an a5_cell vector (or object coercible to one). For <code>a5_hex_to_u64()</code> , a character vector of hex-encoded cell IDs.
---	---

Details

These are named to match `u64_to_hex` / `hex_to_u64` in the upstream Python, JavaScript, and DuckDB A5 bindings. In those languages the functions convert between a native 64-bit unsigned integer and its hex representation. Because R has no native `uint64` type, `a5_u64_to_hex()` accepts an `a5_cell` (which stores the `u64` internally as eight raw bytes) instead of a bare integer.

Value

`a5_u64_to_hex()` returns a character vector. `a5_hex_to_u64()` returns an `a5_cell` vector.

See Also

`a5_cell_from_arrow()` and `a5_cell_to_arrow()` for lossless conversion between `a5_cell` and Arrow `uint64` arrays.

Examples

```
cell <- a5_lonlat_to_cell(-3.19, 55.95, resolution = 5)
hex <- a5_u64_to_hex(cell)
hex

a5_hex_to_u64(hex)
```

a5_uncompact

Uncompact a set of A5 cells to a target resolution

Description

Expands each cell to its descendants at the target resolution.

Usage

```
a5_uncompact(cells, resolution)
```

Arguments

<code>cells</code>	An <code>a5_cell</code> vector.
<code>resolution</code>	Integer scalar target resolution (0–30).

Value

An `a5_cell` vector of uncompact cells.

See Also

`a5_compact()`

Examples

```
cell <- a5_lonlat_to_cell(-3.19, 55.95, resolution = 5)
a5_uncompact(cell, resolution = 7)
```

wk_methods

wk methods for a5_cell

Description

Integration with the [wk](#) geometry framework. Allows `a5_cell` vectors to be handled as geometry (via their boundary polygons) and to report their CRS.

Usage

```
## S3 method for class 'a5_cell'
wk_handle(handleable, handler, ...)

## S3 method for class 'a5_cell'
wk_crs(x)
```

Arguments

<code>handleable, x</code>	An a5_cell vector.
<code>handler</code>	A wk handler .
<code>...</code>	Passed to underlying methods.

Value

- `wk_handle()`: the result of the handler.
- `wk_crs()`: a [wk::wk_crs](#) object (WGS 84 lon/lat).

Index

a5_cell, [2](#), [4–14](#), [16–19](#)
a5_cell_area, [3](#)
a5_cell_area(), [4](#), [5](#)
a5_cell_distance, [4](#)
a5_cell_edge_length_avg, [5](#)
a5_cell_from_arrow, [5](#)
a5_cell_from_arrow(), [18](#)
a5_cell_to_arrow (a5_cell_from_arrow), [5](#)
a5_cell_to_arrow(), [18](#)
a5_cell_to_boundary, [6](#)
a5_cell_to_boundary(), [5](#), [8](#)
a5_cell_to_children, [7](#)
a5_cell_to_children(), [9](#), [11](#), [12](#)
a5_cell_to_lonlat, [8](#)
a5_cell_to_lonlat(), [4](#), [7](#), [14](#)
a5_cell_to_parent, [9](#)
a5_cell_to_parent(), [8](#), [12](#)
a5_compact, [9](#)
a5_compact(), [18](#)
a5_get_num_cells, [10](#)
a5_get_num_cells(), [11](#)
a5_get_num_children, [11](#)
a5_get_res0_cells, [11](#)
a5_get_resolution, [12](#)
a5_get_resolution(), [8](#), [9](#)
a5_get_threads (a5_set_threads), [16](#)
a5_grid_disk, [12](#)
a5_grid_disk(), [14](#), [17](#)
a5_hex_to_u64 (a5_u64_to_hex), [17](#)
a5_is_valid (a5_cell), [2](#)
a5_linestring_to_cells, [13](#)
a5_linestring_to_cells(), [16](#)
a5_lonlat_to_cell, [14](#)
a5_lonlat_to_cell(), [8](#)
a5_polygon_to_cells, [15](#)
a5_polygon_to_cells(), [14](#)
a5_set_threads, [16](#)
a5_spherical_cap, [17](#)
a5_spherical_cap(), [13](#)
a5_u64_to_hex, [17](#)
a5_u64_to_hex(), [6](#)
a5_uncompact, [18](#)
a5_uncompact(), [10](#), [11](#), [15](#), [16](#)
as_a5_cell (a5_cell), [2](#)

is_a5_cell (a5_cell), [2](#)

units::set_units(), [3–5](#)
units::units, [3–5](#)

wk, [19](#)
wk handler, [19](#)
wk::rct(), [15](#)
wk::wk_crs, [19](#)
wk::wk_handle(), [13](#), [15](#)
wk::wkb(), [6](#), [15](#)
wk::wkt(), [6](#), [15](#)
wk::xy(), [8](#)
wk_crs.a5_cell (wk_methods), [19](#)
wk_handle.a5_cell (wk_methods), [19](#)
wk_methods, [19](#)