Package: mapbaltimore (via r-universe)

August 30, 2024

Type Package

Title Make maps for Baltimore City with open data

Version 0.1.1.9001

Maintainer Eli Pousson <eli.pousson@gmail.com>

Description This package provides data from the Baltimore City, the state of Maryland, and other sources, functions to access additional data, and function to create and modify simple maps of Baltimore neighborhoods using sf and ggplot2.

License MIT + file LICENSE

URL https://github.com/elipousson/mapbaltimore, https://elipousson.github.io/mapbaltimore/

BugReports https://github.com/elipousson/mapbaltimore/issues

Depends R (>= 2.10)

Imports cli, dplyr, esri2sf (>= 0.2.0), getdata (>= 0.1.0.9004), glue, janitor, lifecycle (>= 1.0.3), lubridate, magrittr, pkgconfig, purrr, rappdirs, rlang (>= 1.1.0), sf, sfext (>= 0.1.1), snakecase, stringr, tibble, tidyr, tidyselect, utils

Suggests bcpss, covr, forcats, ggplot2, ggrepel, knitr, maplayer, naniar, osmdata, progress, readr, rmarkdown, RSocrata, testthat (>= 3.0.0)

VignetteBuilder knitr

Remotes Chicago/RSocrata, elipousson/bcpss, elipousson/esri2sf, elipousson/getdata, elipousson/maplayer, elipousson/sfext

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

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

RemoteUrl https://github.com/elipousson/mapbaltimore

RemoteRef HEAD

RemoteSha 18d599c91c0c66a5d94615b73eadbad075824822

Contents

adopted_plans . . . 4
baltimore_bbox . . . 4
baltimore_blocks . . . 5
baltimore_block_groups . . . 6
baltimore_census_xwalk . . . 6
baltimore_city . . . 7
baltimore_city_detailed . . . 8
baltimore_gis_index . . . 8
baltimore_mihp . . . 9
baltimore_msa_counties . . . 10
baltimore_msa_water . . . 11
baltimore_pumas . . . 12
baltimore_tracts . . . 13
baltimore_water . . . 13
balt_tbl_labs . . . 14
bcps_programs . . . 15
bcps_zones . . . 15
buildings_21stc . . . 16
cache_baltimore_data . . . 17
chap_districts . . . 18
check_area . . . 19
circulator_routes . . . 19
circulator_stops . . . 20
congressional_districts . . . 21
council_districts . . . 22
csas . . . 22
explore_baltimore . . . 23
filter_streets . . . 24
get_area . . . 25
get_area_911_calls . . . 27
get_area_bcps_programs . . . 28
get_area_census_geography . . . 29
get_area_citations . . . 29
get_area_crime . . . 30
get_area_data . . . 32
get_area_permits . . . 33
get_area_property . . . 34
get_area_requests . . . 35
get_area_streets . . . 38
get_area_vacants . . . 39
get_area_zoning . . . 40

get_baltimore_esri_data . . . 41
get_baltimore_worker_flows . . . 42
get_batch . . . 43
get_intersection . . . 46
get_nearby_areas . . . 47
get_streets . . . 47
hmt_2017 . . . 49
inspire_plans . . . 50
layer_area_property . . . 51
layer_area_streets . . . 52
legislative_districts . . . 53
legislative_districts_2012 . . . 54
main_streets . . . 55
map_area_bcps_programs . . . 55
map_area_highlighted . . . 56
map_area_in_areas . . . 57
map_area_in_city . . . 57
map_area_mta_services . . . 58
map_area_parks . . . 59
map_area_property . . . 60
map_area_zoning . . . 61
maryland_open_data_api_key . . . 61
mta_bus_lines . . . 62
mta_bus_stops . . . 63
mta_light_rail_lines . . . 64
mta_light_rail_stations . . . 65
mta_marc_lines . . . 66
mta_marc_stations . . . 66
mta_subway_lines . . . 67
mta_subway_stations . . . 68
named_intersections . . . 69
neighborhoods . . . 69
neighborhoods_2020 . . . 70
parks . . . 71
park_districts . . . 72
planning_districts . . . 72
police_districts . . . 73
police_districts_2023 . . . 73
public_art . . . 74
rec_centers . . . 75
request_types . . . 76
respagency_codes . . . 77
scale_mapbaltimore . . . 77
schools_21stc . . . 79
set_map_theme . . . 81
streets . . . 81
wards_1797_1918 . . . 82
xwalk_block2tract . . . 83

xwalk_csa2nsa 83
xwalk_neighborhood2tract 84
xwalk_zip2csa 84
zoning 85

Index **86**

`adopted_plans` *Adopted city plans, accepted community-initiated plans, and LINCS corridors*

Description

Combined area plans and LINCS corridor data from the Baltimore City Department of Planning.

Usage

```
adopted_plans
```

Format

A data frame with 59 rows and 5 variables:

`plan_name` Plan or area name

`year_adopted` Year adopted or initiated

`program` Planning program

`url` URL of plan website or document

`geometry` MULTIPOLYGON geometry for plan areas and MULTILINESTRING geometry for LINCS corridors

Source

<...>

`baltimore_bbox` *Baltimore City WGS84 Bounding Box*

Description

A generalized boundary for Baltimore City, Maryland (`baltimore_city`) converted to a bounding box object using a EPSG:4326 coordinate reference system.

Usage

```
baltimore_bbox
```

Format

A bbox class object.

Source

https://www.census.gov/geo/maps-data/data/tiger-line.html

`baltimore_blocks` *U.S. Census Blocks - 2020*

Description

U.S. Census Blocks for Baltimore city, Maryland downloaded from the U.S. Census Bureau API with the tigris package.

Usage

```
baltimore_blocks
```

Format

A data frame with 13,598 rows and 9 variables:

`tractce10` Tract FIPS

`blockce10` Block FIPS

`geoid10` Block GeoID

`name10` Block name

`aland10` Land area

`awater10` Water area

`intptlat10` Interior center point latitude

`intptlon10` Interior center point longitude

`geometry` MULTIPOLYGON geometry for block boundary

Source

https://www.census.gov/geo/maps-data/data/tiger-line.html

`baltimore_block_groups`

U.S. Census Block Groups - 2020

Description

U.S. Census Block Groups for Baltimore city, Maryland downloaded from the U.S. Census Bureau API with the tigris package.

Usage

```
baltimore_block_groups
```

Format

A data frame with 618 rows and 9 variables:

`tractce` Census tract code

`blkgrpce` Census block group number

`geoid` Census block group identifier; a concatenation of the state FIPS code, county FIPS code, census tract code, and block group number

`namelsad` translated legal/statistical area description and the block group number

`aland` land area (square meters)

`awater` water area (square meters)

`intptlat` latitude of the internal point

`intptlon` longitude of the internal point

`geometry` POLYGON geometry for block group boundary

Source

https://www.census.gov/geo/maps-data/data/tiger-line.html

`baltimore_census_xwalk`

Crosswalk for Baltimore areas and Census geography

Description

A pre-built crosswalk data frame that can be filtered by geography and then used with the `getACS::use_area_xwalk()` function. Crosswalk uses the 2010 City Council district boundaries, 2010 and 2020 neighborhood (neighborhood statistical area) boundaries, and current city planning districts.

Usage

```
baltimore_census_xwalk
```

Format

A data frame with 1024 rows and 8 variables:

`geography` Geography/area type

`GEOID` 2020 Census GeoID

`TRACTCE20` 2020 Census Tract ID

`name` Area name

`POP20` Population in area and tract

`perc_POP20` Percent of population in area and tract

`HOUSING20` Households/occupied housing units in area and tract

`perc_HOUSING20` Percent of households in area and tract

Details

Created using the `getACS::make_area_xwalk()` data.

---

baltimore_city *Generalized political boundary for Baltimore City*

Description

A generalized boundary for Baltimore City, Maryland using TIGER/Line Shapefiles data from the U.S. Census Bureau downloaded with `tigris::county_subdivisions()`.

Usage

```
baltimore_city
```

Format

A data frame with 1 row and 3 variables:

`name` County name

`countyfp` 3-character county FIPS code

`geoid` county identifier; a concatenation of state FIPS code and county FIPS code

`aland` land area (square meters)

`awater` water area (square meters)

`intptlat` latitude of the internal point

`intptlon` longitude of the internal point

`geometry` MULITPOLYGON boundary geometry

Source

https://www.census.gov/geo/maps-data/data/tiger-line.html

`baltimore_city_detailed` *Detailed physical boundary for Baltimore City*

Description

A detailed physical boundary of Baltimore City filtered from statewide detailed boundary data available through Maryland iMap.

Usage

```
baltimore_city_detailed
```

Format

A data frame with 1 row and 3 variables:

`name` County name

`countyfp` 3-character county FIPS code

`geometry` MULITPOLYGON boundary geometry

Source

Maryland Physical Boundaries - County Boundaries (Detailed)

`baltimore_gis_index` *Baltimore ArcGIS Server index data*

Description

A data.frame indexing the layers, services, and folders on four ArcGIS Servers maintained by the Baltimore City Mayor's Office of Information Technology (MOIT) Enterprise GIS (EGIS) program. A limited number of potential sensitive and unresponsive server layers have been excluded. Used by the `get_baltimore_esri_data()` function. Updated 2023 May 26.

Usage

```
baltimore_gis_index
```

Format

A data frame with 1,324 rows and 17 variables:

`name` Name

`nm` Name with snake case

`type` Service/layer type

`url` Folder/service/layer URL

`urlType` URL type

`folderPath` Index type

`serviceName` Service name

`serviceType` Service type

`id` integer Layer ID number

`parentLayerId` integer Parent layer ID number

`serviceItemId` integer Service item ID number

`defaultVisibility` logical Layer default visibility

`minScale` double Minimum scale

`maxScale` integer Maximum scale

`geometryType` Geometry type

`subLayerIds` list Sublayer ID numbers

`supportsDynamicLegends` logical Supports dynamic legends

`baltimore_mihp` *Maryland Inventory of Historic Properties in Baltimore City*

Description

Baltimore City properties included in the Maryland Inventory of Historic Properties (MIHP). The MIHP is an administrative inventory maintained by the Maryland Historical Trust, Maryland's statewide historic preservation office and an agency within the Maryland Department of Planning. The boundaries represent property boundaries and district boundaries depending on the type of MIHP record. Updated 2023 March 29.

Usage

```
baltimore_mihp
```

Format

A data frame with 5,203 rows and 14 variables:

`num_polys` Number of polygons

`mihp_id` MIHP ID

`property_id` Property ID

`mihp_num` MIHP Number

`name` Property name

`alternate_name` Alternate property name

`full_address` Full street address

`town` Town name

`county` County

`pdflink` URL for PDF MIHP form

`xcoord` Longitude

`ycoord` Latitude

`do_erecord` Indicator for electronic records.

`geometry` MULTIPOLYGON geometry for property/district boundaries.

Source

Maryland Inventory Historic Properties (MD iMap)

`baltimore_msa_counties` *County boundaries for the Baltimore–Columbia–Towson MSA*

Description

Counties boundaries in the Baltimore–Columbia–Towson Metropolitan Statistical Area (MSA) include Baltimore City, Baltimore County, Carroll County, Anne Arundel County, Howard County, Queen Anne's County, and Harford County.

Usage

```
baltimore_msa_counties
```

Format

A data frame with 7 rows and 13 variables:

`countyfp` County FIPS code

`countyns` County GNIS code

`geoid` Unique county FIPS code (concatenation of state and county FIPS codes)

`name` County name

`namelsad` Concatenated variable length geographic area name and legal/statistical area description (LSAD)

`lsad` Legal/statistical area description (LSAD)

`classfp` FIPS class code

`funcstat` Functional status

`aland` Land area (square meters)

`awater` Water area (square meters)

`intptlat` Latitude of the internal point

`intptlon` Longitude of the internal point

`geometry` MULTIPOLYGON geometry for county boundary

Source

https://www.census.gov/geo/maps-data/data/tiger-line.html

`baltimore_msa_water` *Baltimore Metropolitan Statistical Area (MSA) Water Polygons*

Description

Downloaded using tigris package.

Usage

```
baltimore_msa_water
```

Format

A data frame with 3,491 rows and 9 variables:

`ansicode` American National Standards Institute codes (ANSI codes)

`hydroid` Unique key for hydrographic features

`fullname` Full name

`mtfcc` MAF/TIGER Feature Class Code

`aland` land area (square meters)

awater water area (square meters)

intptlat latitude of the internal point

intptlon longitude of the internal point

geometry POLYGON geometry for water areas

---

baltimore_pumas *Baltimore PUMAS (Public Use Microdata Areas) - 2010*

Description

The U.S. Census Bureau explains that "Public Use Microdata Areas (PUMAs) are non-overlapping, statistical geographic areas that partition each state or equivalent entity into geographic areas containing no fewer than 100,000 people each... The Census Bureau defines PUMAs for the tabulation and dissemination of decennial census and American Community Survey (ACS) Public Use Microdata Sample (PUMS) data."

Usage

```
baltimore_pumas
```

Format

A data frame with 5 rows and 8 variables:

pumace10 PUMA code

geoid10 GeoID

namelsad10 name and the translated legal/statistical area description code for census tract

aland10 land area (square meters)

awater10 water area (square meters)

intptlat10 latitude of the internal point

intptlon10 longitude of the internal point

geometry Polygon with PUMA boundary

Source

https://www.census.gov/geo/maps-data/data/tiger-line.html

baltimore_tracts *U.S. Census Tracts - 2020*

Description

U.S. Census Tracts for Baltimore city, Maryland downloaded from the U.S. Census Bureau API with the tigris package.

Usage

```
baltimore_tracts
```

Format

A data frame with 199 rows and 9 variables:

`tractce` census tract code

`geoid` nation-based census tract identifier; a concatenation of state FIPS code, county FIPS code, and census tract number

`name` Variable length geographic area name

`namelsad` name and the translated legal/statistical area description code for census tract

`aland` land area (square meters)

`awater` water area (square meters)

`intptlat` latitude of the internal point

`intptlon` longitude of the internal point

`geometry` Polygon with tract boundary

Source

https://www.census.gov/geo/maps-data/data/tiger-line.html

baltimore_water *Baltimore City Water*

Description

Detailed MULTIPOLYGON data for area of streams, lakes, and other water bodies in Baltimore City.

Usage

```
baltimore_water
```

Format

A data frame with 468 rows and 6 variables:

`name` Water feature name, if available

`type` Water type

`subtype` Water subtype

`symbol` Symbol

`water` Water indicator

`acres` Feature area in acres

`geometry` MULTIPOLYGON geometry

Source

https://dotgis.baltimorecity.gov/arcgis/rest/services/DOT_Map_Services/DOT_Basemap/MapServer/7

---

balt_tbl_labs *Baltimore Data Table Labels*

Description

A data.frame with labels to use with tables created using mapbaltimore data. The Housing Market Typology 2017 data is the only set of labels included and the preset table functions are not yet implemented.

Usage

```
balt_tbl_labs
```

Format

A data frame with 9 rows and 7 variables:

`fn_name` Function name

`table` Table name

`col` Column name

`label` Column label

`definition` Column variable definition (logical)

`source` logical Column variable data source

`fmt` Column data format

Source

https://docs.google.com/spreadsheets/d/1FXEJlhccnhoQmSO2WydBidXIw-f2lpomURDGy9KBgJw/edit?usp=sharing

bcps_programs *Baltimore City Public School Programs (SY 2021-2022)*

Description

Locations of school buildings/school programs from SY 2021-2022 joined by location to OpenStreetMap polygons tagged with "amenity:school".

Usage

```
bcps_programs
```

Format

A data frame with 164 rows and 7 variables:

`program_name_short` Program or school name (short)

`program_number` Program number

`osm_name` OpenStreetMap name

`osm_id` OpenStreetMap identifier

`type` Program type

`category` Program category or grade band, e.g. E, EM, H, etc.

`swing_space` Program located in a temporary swing space; logical

`geometry` MULTIPOLYGON geometry for school program location

Source

https://services3.arcgis.com/mbYrzb5fKcXcAMNi/ArcGIS/rest/services/SY2122_Ezones_and_Programs/FeatureServer/11

bcps_zones *Baltimore City Public Schools School Zones or School Attendance Zones (SY 2021-2022)*

Description

Baltimore City Public Schools School Zones also known as School Attendance Zones.

Usage

```
bcps_zones
```

Format

A data frame with 96 rows and 4 variables:

`zone_name` Program name with zone appended

`program_number` Program number

`program_name_short` Program or school name (short)

`type` Program type

`category` Program category or grade band, e.g. E, EM, H, etc.

`geometry` MULTIPOLYGON geometry for school zone boundary

Source

https://services3.arcgis.com/mbYrzb5fKcXcAMNi/ArcGIS/rest/services/SY2122_Ezones_and_Programs/FeatureServer/15

---

buildings_21stc *Baltimore 21st Century Schools*

Description

Buildings constructed or renovated through the 21st Century Schools Program. See `schools_21stc` for school-level information.

Usage

```
buildings_21stc
```

Format

A data frame with 28 rows and 20 variables:

`bldg_name` Building name

`name` Name (identical to build name)

`bldg_name_short` Short building name

`project_type` Project type

`project_url` Project URL

`building_occupied` Building occupied year/season (or scheduled occupation date)

`inspire_plan` INSPIRE Plan

`inspire_plan_url` INSPIRE Plan URL

`school_names` School/program names

`school_names_short` Short school/program names

`school_numbers` School numbers

`grade_bands` Schools grade bands

grades_served Grades served

address Street address

city City

state State

zip Zip code

lon Longitude (EPSG 4326)

lat Latitude (EPSG 4326)

geometry POINT geometry for building locations

---

cache_baltimore_data *Cache data for mapbaltimore package*

Description

Cache data to rappdirs::user_cache_dir("mapbaltimore")

Usage

```
cache_baltimore_data(data = NULL, filename = NULL, overwrite = FALSE)

cache_msa_streets(
  url =
   "https://geodata.md.gov/imap/rest/services/Transportation/MD_HighwayPerformanceMonitoringSystem/M
  filename = "baltimore_msa_streets.gpkg",
  crs = pkgconfig::get_config("mapbaltimore.crs", 2804),
  overwrite = FALSE
)

cache_edge_of_pavement(
  url =
   "https://gisdata.baltimorecity.gov/egis/rest/services/OpenBaltimore/Edge_of_Pavement/FeatureServe
  filename = "edge_of_pavement.gpkg",
  crs = pkgconfig::get_config("mapbaltimore.crs", 2804),
  overwrite = FALSE
)

cache_baltimore_property(
  url =
   "https://geodata.baltimorecity.gov/egis/rest/services/Housing/dmxOwnership/MapServer/0",
  location = NULL,
  filename = "baltimore_property.gpkg",
  crs = pkgconfig::get_config("mapbaltimore.crs", 2804),
  overwrite = FALSE
)

show_cached_files()
```

Arguments

data	Data to cache.
filename	File name to use for cached file. Defaults to name of data. If the data is an sf object make sure to include the file type, e.g. "data.gpkg", supported by `sf::write_sf()`. All other data is written to rda with `readr::write_rds()`.
overwrite	Logical. Default `FALSE`. If `TRUE`, overwrite any existing cached files that use the same filename.
url	URL
crs	Coordinate reference system.
location	`sf`, `sfc`, or `bbox` object (or other object convertible with `as_bbox()`. Optional.

Details

- Use `cache_msa_streets()` to download and cache street centerline data for all counties in the Baltimore metropolitan area.
- Use `cache_edge_of_pavement()` to download and cache edge of pavement data for Baltimore city.

Value

`show_cached_files()` returns a tibble with the columns:

- `file`, the name of the file,
- `size_MB`, file size in MB,
- `modified`, date and time last modified

`chap_districts` *CHAP Historic Districts*

Description

Historic districts designated by the Baltimore City Commission on Historical and Architectural Preservation (CHAP) which is the local historic preservation office for Baltimore City, Maryland. Updated 2023 February 10.

Usage

```
chap_districts
```

Format

A data frame with 39 rows and 7 variables:

`name` Historic district name

`contact_name` CHAP Staff contact name

`url` URL for CHAP website

`deed_covenant` Design review required under deed covenants

`overlaps_nr_district` District is also designated as or overlaps some or entirely with a designated National Register Historic District

`acres` Acreage

`geometry` MULTIPOLYGON boundary geometry

---

check_area *Validate area provided to mapping or charting function.*

Description

Validate an area for a mapping function or another mapbaltimore function.

Usage

```
check_area(area)
```

Arguments

`area`	`sf` object with a column named "name."

---

circulator_routes *Charm City Circulator Routes*

Description

The Baltimore City Department of Transportation describes the Charm City Circulator (CCC) as "a fleet of 24 free shuttles that travel four routes in the central business district of Baltimore City, Maryland." The Harbor Connector (HC) is "an extension of the CCC and is the City's free maritime transit service connecting 6 piers through four vessels."

Usage

```
circulator_routes
```

Format

A data frame with 6 rows and 3 variables:

`route_name` Route name

`alt_route_name` Alternate route name

`geometry` MULTILINESTRING geometry for routes

Source

Baltimore CityView - Charm City Circulator Routes

---

circulator_stops *Charm City Circulator and Harbor Connector Stops*

Description

The Baltimore City Department of Transportation describes the Charm City Circulator (CCC) as "a fleet of 24 free shuttles that travel four routes in the central business district of Baltimore City, Maryland." The Harbor Connector (HC) is "an extension of the CCC and is the City's free maritime transit service connecting 6 piers through four vessels."

Usage

```
circulator_stops
```

Format

A data frame with 111 rows and 5 variables:

`stop_num` Stop number as integer

`stop_location` Intersection location (address, intersection, or landmark)

`corner` Intersection corner

`route_name` Route name

`geometry` POINT geometry for stop location

Source

Baltimore CityView - Charm City Circulator Stops

`congressional_districts` *U.S. Congressional Districts for Baltimore City*

Description

U.S. Congressional Districts overlapping with Baltimore City. Downloaded with the tigris package.

Usage

```
congressional_districts
```

Format

A data frame with 3 rows and 15 variables:

`statefp` 2-character state FIPS code

`cd116fp` 116th congressional district FIPS code

`geoid` GeoID

`namelsad` concatenated variable length geographic area name and legal/statistical area description (LSAD)

`lsad` Legal/statistical area description (LSAD)

`cdsessn` Congressional session code

`mtfcc` MAF/TIGER Feature Class Code (MTFCC)

`funcstat` functional status

`aland` land area (square meters)

`awater` water area (square meters)

`intptlat` latitude of the internal point

`intptlon` longitude of the internal point

`label` Congressional District label

`name` Congressional District name

`geometry` MULTIPOLYGON geometry for Congressional district boundary

Source

<...>

council_districts *Baltimore City Council Districts*

Description

Boundaries for the Baltimore City Council Districts used since 2012 (following boundary revisions completed in 2011 based on the 2010 Decennial Census).

Usage

```
council_districts
```

Format

A data frame with 14 rows and 2 variables:

`id` Number of the City Council district

`name` Name of the City Council district

`geometry` MULTIPOLYGON geometry fpr Council district boundary

Source

https://geodata.baltimorecity.gov/egis/rest/services/CityView/City_Council_Districts/MapServer/0

csas *Community Statistical Areas (2010)*

Description

Community Statistical Areas (CSAs) are clusters of neighborhoods and are organized around U.S. Census tract boundaries by the Baltimore Neighborhood Indicators Alliance. In some cases, CSA boundaries may cross neighborhood boundaries. There are 55 CSAs in Baltimore City. Neighborhood lines often do not fall along CSA boundaries. The CSAs were originally created in 2002 and were revised for the publication of Vital Signs 10 using new 2010 Census Tract boundaries. There are no anticipated boundary revisions in 2020.

Usage

```
csas
```

Format

A data frame with 55 rows and 3 variables:

`id` Community Statistical Area id number

`name` Community Statistical Area name

`url` URL to BNIA-JFI webpage on Community Statistical Area

`geometry` MULITPOLYGON boundary geometry

Source

https://bniajfi.org/mapping-resources/

`explore_baltimore` *Explore Baltimore Heritage Stories*

Description

A table of public stories on the Explore Baltimore Heritage website published by Baltimore Heritage. The text of stories on Explore Baltimore Heritage is licensed under a CC BY 4.0 license. Updated on 2023 March 29.

Usage

```
explore_baltimore
```

Format

A data frame with 491 rows and 10 variables:

`id` Story identifier

`featured` Featured indicator

`modified` Modified date/time

`title` Story title

`address` Street address for story location

`thumbnail` URL for thumbnail-size featured image

`fullsize` URL for full-size featured image

`url` URL for story

`geometry` POINT for story location

Source

https://explore.baltimoreheritage.org/

`filter_streets` *Filter streets*

Description

Internal function for filtering streets by multiple parameters

Usage

```
filter_streets(
  x,
  sha_class = NULL,
  street_type = NULL,
  block_num = NULL,
  union = FALSE,
  bbox = NULL,
  call = caller_env()
)
```

Arguments

`x`	sf object with streets to filter
`sha_class`	selected SHA classifications to include. "all" selects all streets with an assigned SHA classification (around one-quarter of all street segments). Additional options include c("COLL", "LOC", "MART", "PART", "FWY", "INT")
`street_type`	selected street subtypes to include. By default, the returned data includes all subtypes except alleys ("STRALY"). Options include c("STRALY", "STRPRD", "STRR", "STREX", "STRFIC", "STRNDR", "STRURD", "STCLN", "STRTN"). Not supported for
`block_num`	Integer vector with block number, e.g. 300, or range of block numbers (e.g. `c(100, 500)`) to filter streets.
`union`	Logical. Default `TRUE`. Union geometry based on `fullname` of streets.
`bbox`	Bounding box to filter passed to location parameter of `getdata::get_location_data()`.

Value

streets filtered by parameters

get_area *Get area of selected type*

Description

Get a sf object with one or more neighborhoods, Baltimore City Council districts, Maryland Legislative Districts, U.S. Congressional Districts, Baltimore Planning Districts, Baltimore Police Districts, or Community Statistical Areas, park districts, or Census blocks, block groups, or tracts. Area type is required and can be used in combination with area name, area id (not supported by all data sets), or location (as an address or sf object). Name and id are not supported for U.S. Census geogrpahies. Use the location parameter to return any areas of the selected type that intersect with the specified location. `get_baltimore_area()` has different parameter names (more consistent with `getdata::get_location()`) and is now recommended over `get_area()` to avoid a name conflict with the `sfext::get_area()` function.

Usage

```
get_area(
  type = c("neighborhood", "council district", "legislative district",
    "congressional district", "planning district", "police district", "csa",
    "park district", "block", "block group", "tract"),
  area_name = NULL,
  area_id = NULL,
  location = NULL,
  union = FALSE,
  area_label = NULL
)

get_baltimore_area(
  type = NULL,
  name = NULL,
  id = NULL,
  location = NULL,
  union = FALSE,
  label = NULL
)

get_neighborhood(name, location = NULL, union = FALSE, ...)
```

Arguments

type	Required. Area type matching one of the boundary datasets included with mapbaltimore. Supported values include "neighborhood", "council district", "legislative district", "congressional district", "planning district", "police district", "csa", "park district". U.S. Census geographies including "block", "block group", and "tract" are supported when using the location parameter only.
area_name	name or names matching id column in data of selected dataset. Character.

area_id	identifier or identifiers matching id column of selected dataset. Not all supported datasets have an id column and the id may be an integer or character depending on the dataset.
location	Location supports to types of values: an address that can be geocoded using tidygeocoder::geo() *or* an sf object that intersects with the selected area types. If using an sf object, the CRS for the object must be EPSG:2804.
union	If TRUE and multiple area names are provided, the area geometry is combined with sf::st_union(). Defaults to FALSE.
area_label	Label to use as name for area if union is TRUE or as additional label column if union is FALSE. If union is TRUE and area_label is not provided, the original area names are concatenated into a single string.
name	Passed to area_name by get_baltimore_area()
id	Passed to area_id by get_baltimore_area()
label	Passed to area_label by get_baltimore_area()
...	Additional parameters passed by get_neighborhood() to get_area()

See Also

neighborhoods,council_districts,legislative_districts, congressional_districts,planning_districts,police_districts,csas, park_districts tidygeocoder::geo()

Examples

```
# Get the Harwood neighborhood by name
get_area(
  type = "neighborhood",
  area_name = "Harwood"
)

# Get City Council District 12 and 14 by id
get_area(
  type = "council district",
  area_id = c(12, 14)
)

# Get the east and southeast planning districts and combine them
get_area(
  type = "planning district",
  area_id = c("East", "Southeast"),
  union = TRUE,
  area_label = "East and Southeast Planning Districts"
)

# Get legislative district for Walters Art Museum (600 N Charles St)
get_area(
  type = "legislative district",
  location = "600 N Charles St, Baltimore, MD 21201"
)
```

```
# Get Census tracts for the Edmondson Village neighborhood
get_area(
  type = "tract",
  location = get_area("neighborhood", "Edmondson Village")
)
```

`get_area_911_calls` *Get area 911 calls for service from Open Baltimore*

Description

`get_area_911_calls()` can return public records on 911 calls for service from 2017 through the present year.

Usage

```
get_area_911_calls(
  area_type = NULL,
  area_name = NULL,
  description = NULL,
  year = 2023,
  start_date = NULL,
  end_date = NULL,
  where = NULL,
  ...
)
```

Arguments

`area_type`	Area type. Requires area_name is also provided. Options include "neighborhood", "council district", or "police district"
`area_name`	Area name. Requires area_type is also provided.
`description`	String matching call description, e.g. "DISORDERLY", "BURGLARY", "DISCHRG FIREARM", etc.
`year`	numeric. Year of calls for service. Currently only one year at a time is supported (except for years since 2021). If `NULL`, the oldest year from the start_date and end_date is used.
`start_date`	Character string in format YYYY-MM-DD. Filters calls by date.
`end_date`	Character string in format YYYY-MM-DD. Filters calls by date.
`where`	string for where condition. Ignored if area_type, area_name, start_date, or end_date are provided.
`...`	Additional parameters passed to `getdata::get_esri_data()` excluding url, where, crs, and .name_repair.

`get_area_bcps_programs` *Get BCPS programs and attendance zones for a local area*

Description

Get BCPS programs and attendance zones for a local area

Usage

```
get_area_bcps_programs(
  area,
  dist = NULL,
  diag_ratio = NULL,
  asp = NULL,
  crop = TRUE,
  trim = FALSE,
  type = c("all", "zones", "programs", "other")
)
```

Arguments

`area`	`sf` object. If multiple areas are provided, they are unioned into a single sf object using `sf::st_union()`
`dist`	buffer distance in meters. Optional.
`diag_ratio`	ratio to set map extent based diagonal distance of area's bounding box. Ignored when `dist` is provided.
`asp`	Aspect ratio of width to height as a numeric value (e.g. 0.33) or character (e.g. "1:3").
`crop`	If TRUE, data cropped to area or bounding box `sf::st_crop()` adjusted by the `dist`, `diag_ratio`, and `asp` parameters provided. Default `TRUE`.
`trim`	If TRUE, data trimmed to area with `sf::st_intersection()`. This option is not supported for any adjusted areas that use the `dist`, `diag_ratio`, or `asp` parameters. Default `FALSE`.
`type`	Type of BCPS data to return. "all" returns a named list with all of the following spatial data. "zones" returns attendance zones, "programs" returns locations of programs (schools) with zones intersecting area (even if the program is located outside the area), "other" returns charter schools and other special schools located within the specified area.

Details

Returns a named list with overlapping BCPS attendance zones, program locations associated with those zones, and any additional programs located within the area.

get_area_census_geography *Get U.S. Census geography overlapping with an area.*

Description

Return an sf object with the U.S. Census blocks, block groups, or tracts overlapping with an area. By default, at least 25% of the tract area or 30% of the block group area, or 50% of the block area must be within the provided area to be returned. Returned sf object includes new columns with the combined land and water area of the Census geography, the Census geography area within the provided area, the percent of Census geography area within the provided area, and the percent of the provided area within the Census geography area.

Usage

```
get_area_census_geography(
  area,
  geography = c("block", "block group", "tract"),
  area_overlap = NULL
)
```

Arguments

area	sf object.
geography	Character vector with type of U.S. Census
area_overlap	Optional. A numeric value less than 1 and greater than 0 representing the physical area of the geography that should be within the provided area to return.

get_area_citations *Get area citations from Open Baltimore*

Description

Get Environmental Control Board (ECB) citations from 2007 to 2021.

Usage

```
get_area_citations(
  area_type = NULL,
  area_name = NULL,
  description = NULL,
  start_date = NULL,
  end_date = NULL,
  where = "1=1",
  geometry = TRUE,
```

```
  crs = pkgconfig::get_config("mapbaltimore.crs", 2804),
  ...
)
```

Arguments

area_type	Area type. Requires area_name is also provided. Options include "neighborhood", "council district", or "police district"
area_name	Area name. Requires area_type is also provided.
description	String matching description of citations, e.g. "SIGNS" filters citations to "PROHIBITED POSTING OF SIGNS ON PUBLIC PROPERTY"
start_date	Character string in format YYYY-MM-DD. Filters citations by violation date.
end_date	Character string in format YYYY-MM-DD. Filters citations by violation date.
where	string for where condition. Ignore where condition if area_type and area_name are provided.
geometry	Return sf object based on lat/lon. Default TRUE. Set to FALSE to return citations with missing coordinates.
crs	Coordinate reference system (CRS) to return. Default 2804
...	Additional parameters passed to getdata::get_esri_data() excluding url, where, crs, and .name_repair.

Examples

```
# Get bulk trash citations for Council District 5
get_area_citations(
   area_type = "council district",
   area_name = "5",
   description = "BULK TRASH")
```

---

get_area_crime *Get area crimes from Open Baltimore*

Description

Get reported crimes since 2014 for a specific area.

Usage

```
get_area_crime(
  area,
  description = NULL,
  date_range = NULL,
  where = NULL,
  dist = NULL,
  diag_ratio = NULL,
```

```
  asp = NULL,
  unit = "m",
  trim = FALSE,
  crs = pkgconfig::get_config("mapbaltimore.crs", 2804)
)
```

Arguments

area	sf, sfc, or bbox object. If multiple areas are provided, they are unioned into a single sf object using `sf::st_union()`.
description	Crime type or description. Supported options include "AGG. ASSAULT", "ARSON", "AUTO THEFT", "BURGLARY", "COMMON ASSAULT", "HOMICIDE", "LARCENY", "LARCENY FROM AUTO", "RAPE", "ROBBERY - CARJACKING", "ROBBERY - COMMERCIAL", "ROBBERY - RESIDENCE", "ROBBERY - STREET", or "SHOOTING"
date_range	Date range as character vector in format of c("YYYY-MM-DD", "YYYY-MM-DD"). Minimum and maximum values are used if length is greater than 1.
where	where query string passed to esri2sf, Default: `NULL`
dist	buffer distance in units. Optional.
diag_ratio	ratio of diagonal distance of area's bounding box used as buffer distance. e.g. if the diagonal distance is 3000 meters and the "diag_ratio = 0.1" a 300 meter will be used. Ignored when `dist` is provided.
asp	Aspect ratio of width to height as a numeric value (e.g. 0.33) or character (e.g. "1:3"). If numeric, `get_asp()` returns the same value without modification.
unit	Units for buffer. Supported options include "meter", "foot", "kilometer", and "mile", "nautical mile" Common abbreviations (e.g. "km" instead of "kilometer") are also supported. Distance in units is converted to units matching GDAL units for x; defaults to "meter"
trim	If `TRUE`, x is trimmed to y with `st_trim()`.
crs	Cordinate reference system to return, Default: 4326 for `sf_to_df()` and `NULL` for `df_to_sf()`.

Examples

```
## Not run:
# Get shootings for the Lauraville area
area <- get_area("neighborhood", "Barclay")
crimes <- get_area_crime(
  area = area,
  date_range = c("2022-01-01", "2022-12-31"),
  description = "SHOOTING"
)

## End(Not run)
```

get_area_data *Get local or cached data for an area*

Description

[Deprecated] Returns data for a selected area or areas with an optional buffer. If both crop and trim are FALSE, the function uses `sf::st_intersects()` to filter data to include the full geometry of anything that overlaps with the area or bbox (if the area is not provided).

Usage

```
get_area_data(
  area = NULL,
  bbox = NULL,
  data = NULL,
  extdata = NULL,
  cachedata = NULL,
  path = NULL,
  url = NULL,
  fn = NULL,
  diag_ratio = NULL,
  dist = NULL,
  asp = NULL,
  crop = TRUE,
  trim = FALSE,
  crs = NULL
)
```

Arguments

area	`sf` object. If multiple areas are provided, they are unioned into a single sf object using `sf::st_union()`
bbox	bbox object defining area used to filter data. If an area is provided, the bounding box is ignored.
data	`sf` object including data in area
extdata	Character. Name of an external geopackage (.gpkg) file included with the package where selected data is available. Available data includes "trees", "unimproved_property", and "vegetated_area"
cachedata	Character. Name of a cached geopackage (.gpkg) file where selected data is available. Running `cache_mapbaltimore_data()` caches data for "real_property", "baltimore_msa_streets", and "edge_of_pavement"
path	Character. Path to local or remote spatial data file supported by `sf::st_read()`
url	Character. URL for FeatureServer or MapServer layer to pass to get_area_esri_data.
fn	Function to apply to area data before returning.

diag_ratio	ratio to set map extent based diagonal distance of area's bounding box. Ignored when dist is provided.
dist	buffer distance in meters. Optional.
asp	Aspect ratio of width to height as a numeric value (e.g. 0.33) or character (e.g. "1:3").
crop	If TRUE, data cropped to area or bounding box sf::st_crop() adjusted by the dist, diag_ratio, and asp parameters provided. Default TRUE.
trim	If TRUE, data trimmed to area with sf::st_intersection(). This option is not supported for any adjusted areas that use the dist, diag_ratio, or asp parameters. Default FALSE.
crs	Coordinate Reference System (CRS) to use for the returned data. The CRS of the provided data and bounding box or area must match one another but are not required to match the CRS provided by this parameter.

---

get_area_permits *Get area building permits from Open Baltimore*

Description

Get building permits from 2019 through the present.

Usage

```
get_area_permits(
  area,
  year = 2022,
  date_range = NULL,
  permit_type = NULL,
  where = NULL,
  dist = NULL,
  diag_ratio = NULL,
  unit = "m",
  asp = NULL,
  trim = FALSE,
  crs = pkgconfig::get_config("mapbaltimore.crs", 2804),
  ...
)
```

Arguments

area	sf, sfc, or bbox object. If multiple areas are provided, they are unioned into a single sf object using sf::st_union().
year	Year. Must be 2019 or later.
date_range	Date range as character vector in format of c("YYYY-MM-DD", "YYYY-MM-DD"). Minimum and maximum values are used if length is greater than 1.

permit_type	Optional. Supported values include "USE", "DEM", "COM", or "BMZ".
where	string for where condition. permit_type and year are ignored if a custom where is provided. Set where to "1=1" to return data for all years since 2019.
dist	buffer distance in units. Optional.
diag_ratio	ratio of diagonal distance of area's bounding box used as buffer distance. e.g. if the diagonal distance is 3000 meters and the "diag_ratio = 0.1" a 300 meter will be used. Ignored when dist is provided.
unit	Units for buffer. Supported options include "meter", "foot", "kilometer", and "mile", "nautical mile" Common abbreviations (e.g. "km" instead of "kilometer") are also supported. Distance in units is converted to units matching GDAL units for x; defaults to "meter"
asp	Aspect ratio of width to height as a numeric value (e.g. 0.33) or character (e.g. "1:3"). If numeric, get_asp() returns the same value without modification.
trim	If TRUE, x is trimmed to y with st_trim().
crs	Cordinate reference system to return, Default: 4326 for sf_to_df() and NULL for df_to_sf().
...	Additional parameters passed to getdata::get_esri_data().

`get_area_property` *Get real property data*

Description

Get showing parcels described as owner occupied, non-owner occupied, vacant, and unimproved. Real property or parcel data is from the Maryland State Department of Assessment and Taxation and may include errors.

Usage

```
get_area_property(
  area = NULL,
  bbox = NULL,
  dist = NULL,
  diag_ratio = NULL,
  unit = "m",
  asp = NULL,
  crop = TRUE,
  trim = FALSE,
  cache = FALSE,
  filename = NULL,
  overwrite = FALSE,
  ...
)

format_property_data(data)
```

Arguments

area	sf object. If multiple areas are provided, they are unioned into a single sf object using sf::st_union()
bbox	bbox object defining area used to filter data. If an area is provided, the bounding box is ignored.
dist	buffer distance in units. Optional.
diag_ratio	ratio of diagonal distance of area's bounding box used as buffer distance. e.g. if the diagonal distance is 3000 meters and the "diag_ratio = 0.1" a 300 meter will be used. Ignored when dist is provided.
unit	Units for buffer. Supported options include "meter", "foot", "kilometer", and "mile", "nautical mile" Common abbreviations (e.g. "km" instead of "kilometer") are also supported. Distance in units is converted to units matching GDAL units for x; defaults to "meter"
asp	Aspect ratio of width to height as a numeric value (e.g. 0.33) or character (e.g. "1:3"). If numeric, get_asp() returns the same value without modification.
crop	If TRUE, data cropped to area or bounding box sf::st_crop() adjusted by the dist, diag_ratio, and asp parameters provided. Default TRUE.
trim	If TRUE, data trimmed to area with sf::st_intersection(). This option is not supported for any adjusted areas that use the dist, diag_ratio, or asp parameters. Default FALSE.
cache	If TRUE, cache data to mapbaltimore cache folder. Defaults to FALSE.
filename	File name to use for cached file. Defaults to name of data. If the data is an sf object make sure to include the file type, e.g. "data.gpkg", supported by sf::write_sf(). All other data is written to rda with readr::write_rds().
overwrite	Logical. Default FALSE. If TRUE, overwrite any existing cached files that use the same filename.
...	Additional parameters passed to getdata::get_esri_data().
data	sf object including data in area

Examples

```
get_area_property(
  area = neighborhoods[1, ],
  dist = -150,
  unit = "m"
)
```

---

get_area_requests *Get area 311 service requests from Open Baltimore*

Description

Get 311 service requests for a specific area. Service requests from 2017 to 2020 area available but only a single year can be requested at a time. Duplicate requests are removed from the returned data. Requests can be filtered by request type, responsible city agency, or both. You can return multiple types or agencies, by using a custom where query parameter or by calling each type/agency separately.

Usage

```
get_area_requests(
  area = NULL,
  year = 2022,
  date_range = NULL,
  request_type = NULL,
  agency = NULL,
  where = NULL,
  dist = NULL,
  diag_ratio = NULL,
  unit = "m",
  asp = NULL,
  trim = FALSE,
  geometry = TRUE,
  crs = pkgconfig::get_config("mapbaltimore.crs", 2804),
  duplicates = FALSE,
  ...
)
```

Arguments

area	sf, sfc, or bbox object. If multiple areas are provided, they are unioned into a single sf object using sf::st_union().
year	Year for service requests. Default 2021. 2017 to 2022 supported.
date_range	Date range as character vector in format of c("YYYY-MM-DD", "YYYY-MM-DD"). Minimum and maximum values are used if length is greater than 1.
request_type	Service request type.
agency	City agency responsible for request. Options include "Transportation", "BGE", "Solid Waste", "Housing", "Water Wastewater", "Health", "Call Center", "Finance", "Liquor Board", "Recreation & Parks", "Fire Department", "Parking Authority", and "General Services"
where	string for where condition. This parameter is ignored if a request_type or agency are provided.
dist	buffer distance in units. Optional.
diag_ratio	ratio of diagonal distance of area's bounding box used as buffer distance. e.g. if the diagonal distance is 3000 meters and the "diag_ratio = 0.1" a 300 meter will be used. Ignored when dist is provided.

unit	Units for buffer. Supported options include "meter", "foot", "kilometer", and "mile", "nautical mile" Common abbreviations (e.g. "km" instead of "kilometer") are also supported. Distance in units is converted to units matching GDAL units for x; defaults to "meter"
asp	Aspect ratio of width to height as a numeric value (e.g. 0.33) or character (e.g. "1:3"). If numeric, `get_asp()` returns the same value without modification.
trim	If TRUE, x is trimmed to y with `st_trim()`.
geometry	Default TRUE. If FALSE, return requests with missing latitude/longitude (for years prior to 2021 only).
crs	Cordinate reference system to return, Default: 4326 for `sf_to_df()` and NULL for `df_to_sf()`.
duplicates	If TRUE, return 311 service requests marked as "Duplicate". If FALSE, filter duplicate requests out of results.
...	Arguments passed on to `esri2sf::esri2sf` `outFields` vector of fields you want to include. default is NULL for all fields. `replaceDomainInfo` If TRUE, add domain information to the return data frame. Default FALSE.

Examples

```
# Get boundary for Edmondson Village
area <- get_area("neighborhood", "Edmondson Village")

# Get fallen limb requests for 2022
get_area_requests(
  area = area,
  date_range = c("2022-11-01", "2022-12-31"),
  request_type = "FOR-Fallen Limb"
)

# Get dirty alley service requests for multiple years using purrr::map_dfr()
purrr::list_rbind(
  purrr::map(
    c(2021, 2020),
    ~ get_area_requests(
      area = area,
      year = .x,
      request_type = "SW-Dirty Alley"
    )
  )
)
```

`get_area_streets` *Get selected area streets*

Description

Get streets within an area or areas.

Usage

```
get_area_streets(
  area = NULL,
  street_type = NULL,
  sha_class = NULL,
  bbox = NULL,
  dist = NULL,
  diag_ratio = NULL,
  asp = NULL,
  trim = FALSE,
  msa = FALSE,
  union = TRUE
)
```

Arguments

`area`	sf object with area of streets to return.
`street_type`	selected street subtypes to include. By default, the returned data includes all subtypes except alleys ("STRALY"). Options include c("STRALY", "STRPRD", "STRR", "STREX", "STRFIC", "STRNDR", "STRURD", "STCLN", "STRTN")
`sha_class`	selected SHA classifications to include. "all" selects all streets with an assigned SHA classification (around one-quarter of all street segments). Additional options include c("COLL", "LOC", "MART", "PART", "FWY", "INT")
`bbox`	bbox object defining area used to filter data. If an area is provided, the bounding box is ignored.
`dist`	buffer distance in meters. Optional.
`diag_ratio`	ratio to set map extent based diagonal distance of area's bounding box. Ignored when `dist` is provided.
`asp`	Aspect ratio of width to height as a numeric value (e.g. 0.33) or character (e.g. "1:3").
`trim`	Logical. Default `FALSE`. Trim streets to area using `sf::st_intersection()`.
`msa`	Logical. Default `FALSE`. Get streets from cached `baltimore_msa_streets.gpkg` file using `cachedata` parameter of `get_area_data` function.
`union`	Logical. Default `TRUE`. Union geometry based on `fullname` of streets.

`get_area_vacants` *Get vacant building notices*

Description

Parcel boundaries for all properties with an active vacant building notice. If a building is unoccupied and unsafe or unfit for people to live or work inside the building, or has two code violations that have not been fixed, or has six code violations in the past year, then the building may receive a vacant building notice in Baltimore City.

Usage

```
get_area_vacants(
  area = NULL,
  bbox = NULL,
  dist = NULL,
  diag_ratio = NULL,
  asp = NULL,
  crop = TRUE,
  trim = FALSE,
  rehabbed = FALSE
)
```

Arguments

`area`	`sf` object. If multiple areas are provided, they are unioned into a single sf object using `sf::st_union()`
`bbox`	bbox object defining area used to filter data. If an area is provided, the bounding box is ignored.
`dist`	buffer distance in meters. Optional.
`diag_ratio`	ratio to set map extent based diagonal distance of area's bounding box. Ignored when `dist` is provided.
`asp`	Aspect ratio of width to height as a numeric value (e.g. 0.33) or character (e.g. "1:3").
`crop`	If TRUE, data cropped to area or bounding box `sf::st_crop()` adjusted by the `dist`, `diag_ratio`, and `asp` parameters provided. Default `TRUE`.
`trim`	If TRUE, data trimmed to area with `sf::st_intersection()`. This option is not supported for any adjusted areas that use the `dist`, `diag_ratio`, or `asp` parameters. Default `FALSE`.
`rehabbed`	If TRUE, return building permits pulled on properties with vacant building notices. Default FALSE.

Details

If the rehabbed parameter is TRUE, the returned data is use and occupancy permits that were pulled on properties with vacant building notices. DHCD uses this data as proxy for vacant building rehabs.

`get_area_zoning` *Get zoning data for an area*

Description

Get zoning codes for an area within a provided sf or bbox object.

Usage

```
get_area_zoning(
  area = NULL,
  bbox = NULL,
  category = c("all", "residential", "commercial", "industrial"),
  diag_ratio = NULL,
  dist = NULL,
  asp = NULL,
  crop = TRUE,
  trim = FALSE,
  crs = NULL,
  union = FALSE
)
```

Arguments

`area`	sf, sfc, or bbox object. If multiple areas are provided, they are unioned into a single sf object using `sf::st_union()`.
`bbox`	bbox object defining area used to filter data. If an area is provided, the bounding box is ignored.
`category`	Zoning category to return. "all", "residential", "commercial", "industrial"
`diag_ratio`	ratio to set map extent based diagonal distance of area's bounding box. Ignored when `dist` is provided.
`dist`	buffer distance in meters. Optional.
`asp`	Aspect ratio of width to height as a numeric value (e.g. 0.33) or character (e.g. "1:3").
`crop`	If TRUE, data cropped to area or bounding box `sf::st_crop()` adjusted by the `dist`, `diag_ratio`, and `asp` parameters provided. Default `TRUE`.
`trim`	If TRUE, data trimmed to area with `sf::st_intersection()`. This option is not supported for any adjusted areas that use the `dist`, `diag_ratio`, or `asp` parameters. Default `FALSE`.
`crs`	Coordinate Reference System (CRS) to use for the returned data. The CRS of the provided data and bounding box or area must match one another but are not required to match the CRS provided by this parameter.
`union`	Logical. Default FALSE. If true, group zoning by label and combine geometry with `sf::st_union()`.

Details

This 2017 zoning data does not include any exemptions granted by the Baltimore City BMZA (Board of Municipal Zoning Appeals).

Value

`sf` object with zoning and overlay data for area.

`get_baltimore_esri_data` *Get Baltimore data*

Description

A wrapper for `getdata::get_esri_data()`

Usage

```
get_baltimore_esri_data(area = NULL, nm = NULL, type = NULL, crs = NULL, ...)
```

Arguments

`area`	Area (passed to location), Default: NULL
`nm`	nm (should match a single value from baltimore_gis_index$nm), Default: NULL
`type`	Type used as an alias for a nm value, Default: NULL
`crs`	Coordinate reference system, Default: NULL
`...`	Arguments passed on to `getdata::get_esri_data`

`url` FeatureServer or MapServer url to retrieve data from. Passed to `url` parameter of `esri2sf::esri2sf()` or `esri2sf::esri2df()` functions. For `get_esri_layers()`, the optional url must be a service url which is the base url for one or more layer urls.

`location` `sf`, `sfc`, or `bbox` object (or other object convertible with `as_bbox()`. Optional.

`dist` buffer distance in units. Optional.

`diag_ratio` ratio of diagonal distance of area's bounding box used as buffer distance. e.g. if the diagonal distance is 3000 meters and the "diag_ratio = 0.1" a 300 meter will be used. Ignored when `dist` is provided.

`unit` Units for buffer. Supported options include "meter", "foot", "kilometer", and "mile", "nautical mile" Common abbreviations (e.g. "km" instead of "kilometer") are also supported. Distance in units is converted to units matching GDAL units for x; defaults to "meter"

`asp` Aspect ratio of width to height as a numeric value (e.g. 0.33) or character (e.g. "1:3"). If numeric, `get_asp()` returns the same value without modification.

`where` where query string passed to esri2sf, Default: `NULL`

`name,name_col` Name value and name column found in the ArcGIS FeatureServer or MapServer data.

`coords` Coordinate columns for input data.frame or output sf object (if geometry is 'centroid' or 'point') Default: c("lon", "lat").

`from_crs` For `df_to_sf()`, coordinate reference system used by coordinates or well known text in data frame.

`clean_names` If `TRUE`, set .name_repair to `janitor::make_clean_names()` Ignored when `get_esri_metadata()` is not returning a data.frame, e.g. `meta = "id"`.

`token` string for authentication token. defaults to `NULL`.

`progress` Show progress bar from `cli::cli_progress_along()` if `TRUE`. Default `FALSE`.

`quiet` If `TRUE`, use `suppressMessages()` to prevent the printing of messages about the requested layer. Defaults to `FALSE`.

`.name_repair` Defaults to "check_unique"

Value

A dataframe or simple feature object

See Also

`getdata::get_esri_data()`

`get_baltimore_worker_flows` *Get Baltimore metro area worker flows from the Census Transportation Planning data (2012-2016 ACS)*

Description

Use FeatureLayers provided by the Baltimore Metropolitan Council.

Usage

```
get_baltimore_worker_flows(
  area,
  tracts = baltimore_tracts,
  min_estimate = 10,
  geometry = TRUE,
  crs = 2804
)
```

Arguments

area	A sf or sfc object that intersects with tracts.
tracts	Data from tigris::tracts() for one or more county in the Balitmore metro area. Defaults to baltimore_tracts.
min_estimate	Minimum number of workers or residents a tract must have to include in results. Tracts with fewer than the min_estimate values are filtered out of results. Defaults to 10.
geometry	If TRUE, return a list of sf objects. If FALSE, return a list of data.frame objects. Defaults to TRUE.
crs	Coordinate reference system to use for returned data when geometry = TRUE. Defaults to 2804.

Value

A list of two data.frames or sf objects named "to" and "from".

---

get_batch *Batch load or save data for an area, street, or intersection*

Description

This batch loading/saving function is less flexible than get_area_data() can reduce the need for repetitive calls to get_area_data() when gathering area-level data for mapping.

- get_data_batch() calls get_area_data().
- get_area_batch() calls get_area() using the provided area as the location parameter.

Usage

```
get_data_batch(
  get = NULL,
  area = NULL,
  label = get,
  adj = list(dist = 15, diag_ratio = NULL, asp = "6:4"),
  fn = NULL,
  batch = NULL,
  crop = TRUE,
  trim = FALSE,
  load = TRUE,
  cache = FALSE,
  save = FALSE,
  filetype = "geojson",
  crs = pkgconfig::get_config("mapbaltimore.crs", 2804),
  ...
)
```

```
get_area_batch(
  get = NULL,
  area = NULL,
  label = get,
  adj = list(dist = 15, diag_ratio = NULL, asp = "6:4"),
  fn = NULL,
  batch = c("neighborhood", "council district", "csa", "tract"),
  trim = FALSE,
  load = TRUE,
  save = FALSE,
  cache = FALSE,
  filetype = "geojson",
  crs = pkgconfig::get_config("mapbaltimore.crs", 2804),
  ...
)
```

Arguments

get	Type of geography to use in setting the area of data to load or save. Supported values area "area", "street", or "intersection". Default: NULL
area	An sf object to use instead of getting an area, street, or intersection. Only used if get is NULL.
label	Label to use for the loaded objects or saved files, Defaults to the same as the get parameter.
adj	Named list with parameters used by `adjust_bbox()` to create a bounding box for the area, street, or intersection. Set to NULL if to use the area as is (or to use another sf object with the other_area parameter) Default: list(dist = 15, diag_ratio = NULL, asp = "6:4").
fn	Function to apply to area after returning it. Useful for applying a buffer to a street or creating a walking distance isochrone to use as the bounding box for an intersection.
batch	A character string or named list. • If using `get_area_batch()`, batch must be a character vector or list with the type(s) of area supported by `get_area()`. Any area intersecting with the area or adjusted area is returned. Default: "neighborhood", "council district", "csa", "tract" • If using `get_data_batch()`, batch must be a character vector matching one of the spatial datasets included with the mapbaltimore package or cached in advance. "osm_buildings" is a special supported parameter that calls `get_area_osm_buildings()` to return all building footprints in the bounding box. Default: c("streets", "parks", "zoning", "hmt_2017", "mta_bus_lines", "mta_bus_stops", "trees", "vegetated_area", "unimproved_property"). A named list where list items are sf objects, supported character strings, or valid URLs for ArcGIS FeatureServer or MapServer layers is also supported. Default: NULL

crop	If FALSE, return data that intersects with the bounding box of the area, street, or intersection but do not crop to the bounding box. This parameter is not supported for get_area_batch(). Default: TRUE.
trim	If TRUE (and if adj is NULL), trim the data to the area, street, or intersection. Default: FALSE.
load	If TRUE, load the datasets to the global environment, Default: TRUE
cache	If TRUE, cache the datasets to the package cache folder with cache_baltimore_data(). Default FALSE.
save	If TRUE, save the selected areas and datasets locally as a file (using the filetype parameter as a file extension)., Default: FALSE
filetype	File extension supported by sf::write_sf(), Default: 'geojson'
crs	Coordinate reference system
...	Parameters passed to get_area(), get_streets(), or get_intersection() depending on the value of the get parameter.

Examples

```
## Not run:
if (interactive()) {
  # Load streets and cached edge of pavement data for the Harwood neighborhood
  get_data_batch(
    get = "area",
    label = "harwood",
    type = "neighborhood",
    area_name = "Harwood",
    batch = c("streets", "edge_of_pavement"),
    load = TRUE,
    save = FALSE
  )

  # Save parks, trees, and vegetated area w/in 800 meters
  # of the intersection of E. Pratt and Light Sts. to GeoJSON files
  get_data_batch(
    get = "intersection",
    street_names = "E PRATT ST & LIGHT ST",
    adj = list(dist = 0, diag_ratio = NULL, asp = "1:1"),
    dist = 800,
    batch = c("parks", "trees", "vegetated_area")
  )
}

## End(Not run)
```

`get_intersection` *Get intersections*

Description

Get intersections by name and id with option to apply buffer and return streets or edgement of pavement instead of the intersection.

Usage

```
get_intersection(
  street_names = NULL,
  id = NULL,
  dist = 25,
  type = c("area", "edge_of_pavement", "streets"),
  trim = TRUE
)
```

Arguments

`street_names`	street names matching one or more of the names from the `named_intersections` data.
`id`	id values corresponding to one or more id values from the `named_intersections` data.
`dist`	buffer distance in meters. Optional.
`type`	Type of data to return. "area" returns the intersection center if `dist` is 0 or a circle centered on the intersection center with any positive dist value. "edge_of_pavement" or "streets" return what either the cached edge of pavement data or street center line data.
`trim`	If type is "edge_of_pavement" or "streets" and `trim` is TRUE return data trimmed to the buffered intersection, otherwise return data within bounding box, Default: TRUE

Value

Intersection center point, buffered area around intersection center, streets, or edge of pavement data.

Examples

```
get_intersection(street_names = "Overton St & S Chapelgate Lane", dist = 30)

get_intersection(id = "41758", dist = 425, type = "streets", trim = FALSE)
```

get_nearby_areas *Get nearby areas*

Description

Return areas of a selected type within a set distance of another area.

Usage

```
get_nearby_areas(
  area,
  type = c("neighborhood", "council district", "legislative district",
    "congressional district", "planning district", "police district", "csa",
    "park district"),
  dist = 1,
  exclude_area = TRUE,
  residential = FALSE
)
```

Arguments

area	sf object. Must have a name column for exclude_area to work.
type	Required. Supported values include "neighborhood", "council district", "legislative district", "congressional district", "planning district", "police district", "csa", and "park district". The type may be different than the type of the area provided.
dist	Distance in meters for matching nearby areas. Default is 1 meter.
exclude_area	Logical. Default TRUE. If FALSE, include the same areas provided to area (assuming the areas provide are the same type as the parameter provided to get_nearby_areas).
residential	Logical. Default FALSE. If the type is neighborhood, set TRUE to only return residential neighborhoods (excluding industrial areas, business parks, and parks/reservoirs).

get_streets *Get streets*

Description

Get streets in Baltimore City by name with option to exclude streets by name, crop to a bounding box, or to filter to selected street types or functional classifications.

Usage

```
get_streets(
  street_name,
  exclude_name = NULL,
  street_type = NULL,
  sha_class = NULL,
  block_num = NULL,
  bbox = NULL,
  union = TRUE
)
```

Arguments

street_name	Street names to return. Required.
exclude_name	Street names to exclude
street_type	selected street subtypes to include. By default, the returned data includes all subtypes except alleys ("STRALY"). Options include c("STRALY", "STRPRD", "STRR", "STREX", "STRFIC", "STRNDR", "STRURD", "STCLN", "STRTN"). Not supported for
sha_class	selected SHA classifications to include. "all" selects all streets with an assigned SHA classification (around one-quarter of all street segments). Additional options include c("COLL", "LOC", "MART", "PART", "FWY", "INT")
block_num	Integer vector with block number, e.g. 300, or range of block numbers (e.g. `c(100, 500)`) to filter streets.
bbox	bbox to crop returned streets. Optional.
union	Logical. If `TRUE`, use `st_union` to combine geometry by `fullname` of the streets.

Details

DETAILS

Value

OUTPUT_DESCRIPTION

See Also

get_area_streets

streets `get_area_streets()`

Examples

```
get_streets(street_name = "UNIVERSITY PKWY")

get_streets(street_name = c("E FAYETTE", "ORLEANS"), block_num = c(1700, 3600))
```

hmt_2017 *Housing Market Typology 2017*

Description

The 2017 update of the City's Housing Market Typology was jointly developed by the Baltimore City Planning Department, Department of Housing & Community Development, and The Reinvestment Fund.

Usage

```
hmt_2017
```

Format

A data frame with 663 rows and 15 variables:

`geoid` U.S. Census Block Group GeoID

`geoid_part` Identifier for U.S. Census Block Group GeoID including part identifier

`part` Part identifier

`cluster` Housing market cluster

`cluster_group` Housing market cluster

`median_sales_price` Median sales price, Q3 2015 - Q2 2017

`sales_price_variation` Sales price variation, Q3 2015 - Q2 2017

`num_sales` Number of residential sales, Q3 2015 - Q2 2017

`num_foreclosure_filings` Number of foreclosure filings, Q3 2015 - Q2 2017

`perc_foreclosure_sales` Percent of sales through foreclosure, Q3 2015 - Q2 2017

`perc_homeowners` Percent owner occupied, July 2017

`perc_permits_over10k` Percent of residential building permits over $10,000, Q3 2015 - Q2 2017

`vacant_lots_bldgs_per_acre_res` Vacant lots and buildings per residential acre, July 2017

`units_per_acre_res` Housing units per residential acre, July 2017

`geometry` MULTIPOLYGON geometry matching Census blocks groups or parts of block groups

Source

https://opendata.baltimorecity.gov/egis/rest/services/Hosted/Housing_Market_Typology_2017/FeatureServer/0

`inspire_plans` *INSPIRE Plans*

Description

Data frame and boundary geometry for INSPIRE Plans adopted and in progress.

Usage

```
inspire_plans
```

Format

A data frame with 24 rows and 23 variables:

`plan_name` Plan name

`plan_name_short` Plan name (short)

`overall_status` Overall status

`inspire_lead_planner` Lead INSPIRE Planner

`plan_url` Baltimore City Department of Planning plan webpage url

`year_launched` Year launched

`year_adopted` Year adopted by Planning Commission

`adoption_status` Planning Commission adoption status

`adoption_date` Planning Commission adoption data

`document_url` Adopted plan PDF url

`recommendation_report_status` Recommendation report status

`recommendation_report_url` Draft recommendation report PDF url

`kick_off_presentation_date` Kick-off presentation date

`launch_date_target` Target launch date

`walking_route_id_target_date` Primary walking route identification date

`recommendations_date_target` Target draft recommendation report publication date

`commission_review_date_target` Target Planning Commission review date

`implementation_status` Plan implementation status

`program_numbers` School program numbers

`planning_districts` Planning Districts

`neighborhoods` Neighborhoods

`council_districts` Baltimore City Council Districts

`geometry` MULTIPOLYGON boundary geometry

Details

Last updated: 2024-03-29

layer_area_property *Add a layer to a gpplot2 map with area property categorized by type*

Description

Real property or parcel data is from the Maryland State Department of Assessment and Taxation and may include outdated or inaccurate information.

Usage

```
layer_area_property(
  area = NULL,
  bbox = NULL,
  data = NULL,
 type = c("improved", "vacant", "principal residence", "use", "building type", "value"),
  asis = FALSE,
  diag_ratio = NULL,
  dist = NULL,
  asp = NULL,
  crop = TRUE,
  trim = FALSE,
  show_area = FALSE,
  show_mask = FALSE,
  crs = pkgconfig::get_config("mapbaltimore.crs", 2804),
  ...
)
```

Arguments

area	sf object. If multiple areas are provided, they are unioned into a single sf object using sf::st_union()
bbox	bbox object defining area used to filter data. If an area is provided, the bounding box is ignored.
data	sf object including data in area
type	Real property variable to map. Options include c("improved", "vacant", "principal residence", "value"). Currently supports only one variable at a time.
asis	Logical. Default FALSE. If TRUE, use inherited data as is without cropping to area.
diag_ratio	ratio to set map extent based diagonal distance of area's bounding box. Ignored when dist is provided.
dist	buffer distance in meters. Optional.
asp	Aspect ratio of width to height as a numeric value (e.g. 0.33) or character (e.g. "1:3").
crop	If TRUE, data cropped to area or bounding box sf::st_crop() adjusted by the dist, diag_ratio, and asp parameters provided. Default TRUE.

trim	If TRUE, data trimmed to area with sf::st_intersection(). This option is not supported for any adjusted areas that use the dist, diag_ratio, or asp parameters. Default FALSE.
show_area	Logical. Default FALSE. If TRUE, add an outline of the area to the layer.
show_mask	Logical. Default FALSE. If TRUE, add a mask using layer_area_mask
crs	Coordinate Reference System (CRS) to use for the returned data. The CRS of the provided data and bounding box or area must match one another but are not required to match the CRS provided by this parameter.
...	passed to ggplot2::geom_sf() for data layer.

See Also

layer_area_data

Examples

```
## Not run:
area <- get_area("neighborhood", "West Forest Park")

property <- get_area_property(area = area)

ggplot2::ggplot() +
  layer_area_property(area = area, data = property, type = "principal residence")

## End(Not run)
```

---

layer_area_streets *Add a layer to a gpplot2 map with area streets, street names, or both*

Description

Add a layer to a gpplot2 map with area streets, street names, or both.

Usage

```
layer_area_streets(
  area = NULL,
  street_type = NULL,
  sha_class = NULL,
  dist = NULL,
  diag_ratio = NULL,
  asp = NULL,
  trim = FALSE,
  msa = FALSE,
  show_streets = TRUE,
  show_names = FALSE,
  name_location = NULL,
```

```
  edge_dist = 10,
  color = "gray40",
  size = 1,
  ...
)
```

Arguments

area	sf object. Returns streets within this area (after adjustment by dist, diag_ratio, and asp parameters)
street_type	selected street subtypes to include. By default, the returned data includes all subtypes except alleys ("STRALY"). Options include c("STRALY", "STRPRD", "STRR", "STREX", "STRFIC", "STRNDR", "STRURD", "STCLN", "STRTN")
sha_class	selected SHA classifications to include. "all" selects all streets with an assigned SHA classification (around one-quarter of all street segments). Additional options include c("COLL", "LOC", "MART", "PART", "FWY", "INT")
dist	buffer distance in meters. Optional.
diag_ratio	ratio to set map extent based diagonal distance of area's bounding box. Ignored when dist is provided.
asp	Aspect ratio of width to height as a numeric value (e.g. 0.33) or character (e.g. "1:3").
trim	Logical. Default FALSE. Trim streets to area using sf::st_intersection().
msa	Logical. Default FALSE. Get streets from cached baltimore_msa_streets.gpkg file using cachedata parameter of get_area_data function.
show_streets	Logical. Default TRUE. If FALSE, hides street center lines.
show_names	Logical. Default FALSE. If TRUE, shows street names.
name_location	Options include c("area", "edge", "top", "left", "bottom", "right", "topleft", "topright", "bottomleft", "bottomright"). Defaults to NULL.
edge_dist	Distance buffer to use for placing street names.
color	Color of streets and/or text of street name labels.
size	Size of the streets and/or street name labels.
...	Other parameters to pass along to ggplot2::geom_sf() that maps the streets.

---

legislative_districts *Maryland Legislative Districts for Baltimore City (2022)*

Description

A subset of Maryland legislative districts from Maryland iMap.

Usage

```
legislative_districts
```

Format

A data frame with 6 rows and 4 variables:

`name` District name

`id` District number

`label` District label

`geometry` MULTIPOLYGON geometry for district boundary

Source

https://geodata.md.gov/imap/rest/services/Boundaries/MD_ElectionBoundaries_2022/FeatureServer/1

---

legislative_districts_2012

Maryland Legislative Districts for Baltimore City (2012)

Description

A subset of Maryland legislative districts from Maryland iMap.

Usage

```
legislative_districts_2012
```

Format

A data frame with 6 rows and 4 variables:

`name` District name

`id` District number

`label` District label

`geometry` MULTIPOLYGON geometry for district boundary

Source

https://geodata.md.gov/imap/rest/services/Boundaries/MD_ElectionBoundaries/FeatureServer/1

main_streets *Baltimore City Main Streets*

Description

Boundaries for Baltimore City Main Street programs, including two programs that are not currently funded but formerly participated in the program.

Usage

```
main_streets
```

Format

A data frame with 10 rows and 7 variables:

`id` Main Street ID from source FeatureLayer

`name` Main Street name

`name_abb` Name abbreviation

`url` Main Street partner organization URL

`funding_status` Funding status (active or inactive)

`name_short` Short name

`geometry` sfc list column with MULTIPOLYGON boundary geometry

Source

https://services1.arcgis.com/43Lm3JYE3nM91DAF/arcgis/rest/services/MainStreets/FeatureServer/0

map_area_bcps_programs *Map BCPS programs and attendance zones for a local area*

Description

Map showing BCPS school zones that overlap with a provided area or areas. If the area sf tibble includes multiple areas, a separate map is created for each area provided.

Usage

```
map_area_bcps_programs(area)
```

Arguments

`area`	sf object

Examples

```
## Not run:
## Map school attendance boundary zones for the Madison Park neighborhood
madisonpark <- get_area(
  area_type = "neighborhood",
  area_name = "Madison Park"
)
map_area_bcps_programs(area = madisonpark)

## End(Not run)

## Not run:
## Map school attendance boundary zones for City Council District 2
district9 <- get_area(
  type = "council district",
  area_name = "9"
)
map_area_bcps_programs(area = district9)

## End(Not run)
```

---

map_area_highlighted *Maps a highlighted area within the context of multiple areas*

Description

Map highlighting the location of an area the context of multiple areas.

Usage

```
map_area_highlighted(area, highlight_name = "all")
```

Arguments

area	Required sf object with a 'name' column.
highlight_name	Character vector. Required. Use "all" to create a grid of maps highlighting each area in the provided sf object or provide the name of one or more areas to highlight.

map_area_in_areas *Map area within selected overlapping areas*

Description

Map an area or areas within selected overlapping areas.

Usage

```
map_area_in_areas(
  area,
  type = c("neighborhood", "council district", "legislative district",
    "congressional district", "planning district", "police district", "csa"),
  show_area = TRUE,
  show_label = FALSE,
  background = NULL
)
```

Arguments

area	sf object. Required
type	Type of area to map. Supports the same types as the get_area function.
show_area	Logical. Default TRUE.
show_label	Logical. Default FALSE. If TRUE, label areas with ggplot2::geom_sf_label()
background	ggplot layer. Default NULL. Passing a ggplot2 layer may be necessary to have an appropriate background for the congressional district maps.

map_area_in_city *Map area in the context of city boundaries*

Description

Map showing the location of an area within the city.

Usage

```
map_area_in_city(area, area_label = NULL)
```

Arguments

area	sf object with a 'name' column. Required.
area_label	area label to replace area name. Optional.

Examples

```
## Not run:
## Area with a defined label
district2 <- get_area(
  type = "council district",
  area_id = "2"
)

map_area_in_city(
  area = district2,
  area_label = "Baltimore's Second Council District"
)

## End(Not run)

## Not run:
## Multiple areas in a single map
selected_se_neighborhoods <- get_area(
  type = "neighborhood",
  area_name = c("Upper Fells Point", "Fells Point", "Canton")
)

map_area_in_city(
  area = selected_se_neighborhoods,
  area_label = "Southeast Baltimore neighborhoods"
)

## End(Not run)

## Not run:
## Area with a defined map title
canton_industrial <- get_area(
  type = "neighborhood",
  area_name = "Canton Industrial Area"
)

map_area_in_city(area = canton_industrial)

## End(Not run)
```

`map_area_mta_services` *Map MTA services*

Description

Map MTA services. MTA bus lines are currently the only supported service.

Usage

```
map_area_mta_services(
  area,
  mta_services = "bus_lines",
  diag_ratio = 0.166,
  asp = NULL
)
```

Arguments

area	sf object. Required.
mta_services	Character vector. Default is "bus_lines" to use mta_bus_lines data.
diag_ratio	ratio to set map extent based diagonal distance of area's bounding box. Ignored when dist is provided.
asp	Aspect ratio of width to height as a numeric value (e.g. 0.33) or character (e.g. "1:3").

---

map_area_parks *Map area parks and open spaces*

Description

Return a ggplot map showing parks in and around a selected area.

Usage

```
map_area_parks(
  area,
  type = c("parks", "vacant lots"),
  label = c("parks"),
  dist = NULL,
  diag_ratio = 0.125,
  asp = NULL
)
```

Arguments

area	sf object. Required.
type	layers to show on map ("parks" or "vacant lots"). Defaults to both.
label	layers to label. Only "parks" is supported. Use any other value to exclude labels.
dist	buffer distance in meters. Optional.
diag_ratio	ratio to set map extent based diagonal distance of area's bounding box. Ignored when dist is provided.
asp	Aspect ratio of width to height as a numeric value (e.g. 0.33) or character (e.g. "1:3").

map_area_property	*Real property or parcel data is from the Maryland State Department of Assessment and Taxation and may include outdated or inaccurate information.*

Description

Real property or parcel data is from the Maryland State Department of Assessment and Taxation and may include outdated or inaccurate information.

Usage

```
map_area_property(
  area,
  property = c("improved", "vacant", "principal residence", "use", "building type",
    "value"),
  dist = NULL,
  diag_ratio = 0.1,
  asp = NULL,
  trim = FALSE,
  show_mask = FALSE
)
```

Arguments

area	Simple features object. Function currently supports only a single area at a time.
property	Real property variable to map. Options include c("improved", "vacant", "principal residence", "value"). Currently supports only one variable at a time.
dist	buffer distance in meters. Optional.
diag_ratio	ratio to set map extent based diagonal distance of area's bounding box. Ignored when dist is provided.
asp	Aspect ratio of width to height as a numeric value (e.g. 0.33) or character (e.g. "1:3").
trim	If TRUE, data trimmed to area with sf::st_intersection(). This option is not supported for any adjusted areas that use the dist, diag_ratio, or asp parameters. Default FALSE.
show_mask	If TRUE, apply a white, 0.6 alpha mask over property located outside the provided area. Default FALSE.

map_area_zoning *Map zoning for an area (not working)*

Description

Map zoning/zoning overlay codes for an area within the city. The 2017 zoning data does not include any exemptions granted by the BMZA (Board of Municipal Zoning Appeals).

Usage

```
map_area_zoning(
  area,
  category = c("all", "residential", "commercial", "industrial"),
  diag_ratio = 0.125,
  asp = NULL,
  crs = pkgconfig::get_config("mapbaltimore.crs", 2804)
)
```

Arguments

area	sf, sfc, or bbox object. If multiple areas are provided, they are unioned into a single sf object using `sf::st_union()`.
category	Zoning category to return. "all", "residential", "commercial", "industrial"
diag_ratio	ratio to set map extent based diagonal distance of area's bounding box. Ignored when `dist` is provided.
asp	Aspect ratio of width to height as a numeric value (e.g. 0.33) or character (e.g. "1:3").
crs	Coordinate Reference System (CRS) to use for the returned data. The CRS of the provided data and bounding box or area must match one another but are not required to match the CRS provided by this parameter.

maryland_open_data_api_key *Install a Maryland Open Data Portal API Key in Your* `.Renviron` *File for Repeated Use*

Description

This function will add your Maryland Open Data Portal API key to your `.Renviron` file so it can be called securely without being stored in your code. After you have installed your key, it can be called any time by typing `Sys.getenv("MARYLAND_OPEN_DATA_API_KEY")` and can be used in package functions by simply typing MARYLAND_OPEN_DATA_API_KEY If you do not have an `.Renviron` file, the function will create on for you. If you already have an `.Renviron` file, the function will append the key to your existing file, while making a backup of your original file for disaster recovery purposes.

Usage

```
maryland_open_data_api_key(key, overwrite = FALSE, install = FALSE)
```

Arguments

key	The API key provided to you from Maryland Open Data Portal formatted in quotes. A key be be created after signing up https://imap.maryland.gov/Pages/open-data-portal-signup.aspx
overwrite	If this is set to TRUE, it will overwrite an existing MARYLAND_OPEN_DATA_API_KEY that you already have in your .Renviron file.
install	if TRUE, will install the key in your .Renviron file for use in future sessions. Defaults to FALSE.

Examples

```
## Not run:
MARYLAND_OPEN_DATA_API_KEY("111111abc", install = TRUE)
# First time, reload your environment so you can use the key without restarting R.
readRenviron("~/.Renviron")
# You can check it with:
Sys.getenv("MARYLAND_OPEN_DATA_API_KEY")

## End(Not run)

## Not run:
# If you need to overwrite an existing key:
MARYLAND_OPEN_DATA_API_KEY("111111abc", overwrite = TRUE, install = TRUE)
# First time, relead your environment so you can use the key without restarting R.
readRenviron("~/.Renviron")
# You can check it with:
Sys.getenv("MARYLAND_OPEN_DATA_API_KEY")

## End(Not run)
```

---

mta_bus_lines *Maryland Transit Administration (MTA) Bus Routes (2022)*

Description

Maryland Department of Transportation's Maryland Transit Administration Summer 2022 Bus Routes including CityLink, LocalLink, Express BusLink and Commuter Bus services and reflects bus route changes as of June 19, 2022. For full details of service change visit: https://www.mta.maryland.gov/servicechanges/summer2022

Usage

```
mta_bus_lines
```

Format

A data frame with 103 rows and 4 variables:

`route_name` Bus route name

`route_type` Route type (CityLink, LocalLink, or Commuter Bus)

`route_number` Unique route number or color identifier

`route_abb` Route abbreviation (only different from `route_number` for color CityLink routes)

`frequent` Logical indicator of route inclusion in MTA BaltimoreLink's Frequent Transit Network.

`school` Indicator for school routes

`geometry` MULTILINESTRING bus route geometry

Source

Maryland Transit - MTA Bus Lines (MD iMap)

---

mta_bus_stops *Maryland Transit Administration (MTA) Bus Stops (2023)*

Description

Maryland Department of Transportation's Maryland Transit Administration Bus Stops including CityLink, LocalLink, Express BusLink, and Commuter Bus. This data is based on the Winter 2023 schedule and reflects bus stop changes as of February 5, 2023 Ridership data is based on Automatic Passenger Counting (APC) system average daily weekday bus stop ridership (boarding, alighting, and total) from the Fall 2022 period and does not exclude outliers. For full details of service change visit: https://www.mta.maryland.gov/servicechanges/winter2023

Usage

```
mta_bus_stops
```

Format

A data frame with 4536 rows and 14 variables:

`stop_id` Stop identification number

`stop_name` Stop name

`rider_on` Average daily weekday count of riders boarding transit at stop

`rider_off` Average daily weekday count of riders alighting transit at stop

`rider_total` Average daily weekday count of total riders served at stop

`stop_ridership_rank` Stop rank for ridership

`routes_served` Routes served at stop

`mode` Mode served at stop

`shelter` Logical indicator of bus shelter availability

`county` County where stop is located

`direction` Route direction

`stop_location` Stop location

`frequent` Indicator for stop serving frequent transit network

`geometry` POINT stop location geometry

Details

Last updated from the Maryland iMap Source on August 23, 2023.

Source

Maryland Transit - MTA Bus Stops (MD iMap)

`mta_light_rail_lines` *Maryland Transit Administration (MTA) Light RailLink Stations*

Description

Location of MTA Light Rail Stations.

Usage

```
mta_light_rail_lines
```

Format

A data frame with 84 rows and 8 variables:

`id` Feature ID

`rail_name` Line name (Light Rail Line)

`mode` Facility mode (Light Rail)

`tunnel` Tunnel indicator

`direction` Travel direction

`miles` Section mileage

`status` Section status

`geometry` LINESTRING line geometry

Source

Maryland Transit - Light Rail Lines (MD iMap)

`mta_light_rail_stations` *Maryland Transit Administration (MTA) Light RailLink Stations*

Description

Locations for stations on the Baltimore Light RailLink (Baltimore Light Rail) line operated by the Maryland Transit Administration.

Usage

```
mta_light_rail_stations
```

Format

A data frame with 33 rows and 11 variables:

`id` Feature ID

`name` Station name

`address` Station address

`city` City

`state` State

`zipcode` Zipcode

`mode` Facility mode (Light Rail)

`avg_wkdy` Average weekday passengers

`avg_wknd` Average weekend passengers

`facility_type` Facility type

`geometry` POINT geometry for station locations

Source

Maryland Transit - Light RailLink Stations (MD iMap)

mta_marc_lines *Maryland Transit Administration (MTA) MARC Train Lines*

Description

MARC (Maryland Area Regional Commuter) Rail system lines operated by the Maryland Transit Administration.

Usage

```
mta_marc_lines
```

Format

A data frame with 162 rows and 8 variables:

`id` Feature ID

`rail_name` Rail line name

`mode` Facility mode and line name (MARC)

`tunnel` Tunnel indicator

`direction` Travel direction

`miles` Section mileage

`status` Section status

`geometry` LINESTRING geometry for rail lines

Source

Maryland Transit - MARC Train Lines (MD iMap)

mta_marc_stations *Maryland Transit Administration (MTA) MARC Train Stations*

Description

Locations of MARC (Maryland Area Regional Commuter) Rail stations operated by the Maryland Transit Administration.

Usage

```
mta_marc_stations
```

Format

A data frame with 44 rows and 12 variables:

`id` Feature ID

`name` Station name

`address` Station address

`city` City

`state` State

`zipcode` Zipcode

`line_name` Line name

`mode` Facility mode and line name (MARC)

`avg_wkdy` Average weekday passengers

`avg_wknd` Average weekend passengers

`facility_type` Facility type (Station)

`geometry` POINT geometry for station location

Source

Maryland Transit - MARC Trains Stations (MD iMap)

---

mta_subway_lines *Maryland Transit Administration (MTA) SubwayLink Metro Lines*

Description

Route of MTA SubwayLink Metro Line.

Usage

```
mta_subway_lines
```

Format

A data frame with 34 rows and 8 variables:

`id` Feature id number as integer

`rail_name` Subway line name (Metro Line)

`mode` Travel mode (Metro)

`tunnel` Section tunnel indicator

`direction` Travel direction

`miles` Section mileage

`status` Section status

`geometry` MULTILINESTRING geometry for lines

Source

Baltimore Metro Subway Line

`mta_subway_stations` *Maryland Transit Administration (MTA) SubwayLink Metro Stations*

Description

Location of MTA SubwayLink Metro Stations.

Usage

```
mta_subway_stations
```

Format

A data frame with 14 rows and 10 variables:

`id` Station identification number as integer

`name` Station name

`address` Station street address

`city` City

`state` State

`mode` Travel mode (Metro)

`avg_wkdy` Average weekday passengers

`avg_wknd` Average weekend passengers

`facility_type` Facility type (Station)

`geometry` POINT station location geometry

Source

Baltimore Metro SubwayLink Stations

named_intersections *Baltimore City Street Intersection Names*

Description

Index of Baltimore City intersections using names from street centerlines within 20 meters of the intersection boundaries. Data supports the for `get_intersection()` function. Updated 2022 October 13.

Usage

```
named_intersections
```

Format

A data frame with 11506 rows and 3 variables:

`id` Intersection identifier matching id in `edge_of_pavement` data

`name` Intersection name

`geometry` POINT geometry for intersection center

neighborhoods *Neighborhood Boundaries for Baltimore City (2010)*

Description

Baltimore City neighborhoods (officially known as Neighborhood Statistical Areas) established by the Baltimore City Department of Planning based on the 2010 U.S. Decennial Census. Note that these boundaries may or may not be used by local community or neighborhood associations as an area of responsibility or membership recruitment.

Usage

```
neighborhoods
```

Format

A data frame with 278 rows and 6 variables:

`name` Neighborhood name

`type` Type of area, with options including residential, industrial area, park/open space, institutionl area and business park)

`acres` Area of the neighborhood (acres)

`osm_id` Open Street Map (OSM) relation identifier

`wikidata` Wikidata entity identifier

`geometry` MULITPOLYGON boundary geometry

Source

Maryland Baltimore City Neighborhoods (MD iMap)

`neighborhoods_2020` *Neighborhood Boundaries for Baltimore City (2020)*

Description

Baltimore City neighborhoods (officially known as Neighborhood Statistical Areas) established by the Baltimore City Department of Planning based on the 2020 U.S. Decennial Census. This is an updated version of the 2010 Neighborhood Statistical Areas.

Usage

```
neighborhoods_2020
```

Format

A data frame with 279 rows and 8 variables:

`name` Neighborhood name

`name_alt` 2010 neighborhood name

`type` Type of area, with options including residential, industrial area, park/open space, institutionl area and business park)

`acres` Area of the neighborhood (acres)

`osm_id` Open Street Map (OSM) relation identifier

`wikidata` Wikidata entity identifier

`color_id` Color identifier

`geometry` MULITPOLYGON boundary geometry

Source

NSA_Feb2023_service

parks *Baltimore City Parks*

Description

Spatial data for parks and public recreation centers in Baltimore City from the Baltimore City Department of Recreation and Parks. A few names have been updated to use common names or recent new official names so the package version may not match the city data in all cases. The parks have been matched to corresponding entities on OpenStreetMap indicated by the osm_id column.

Usage

```
parks
```

Format

A data frame with 321 rows and 9 variables:

`name` Park name

`id` Identification number from city data

`address` Primary street address

`name_alt` Alternate name

`operator` Park operator, Baltimore City Department of Recreation and Parks (BCRP) or other

`management` Park management/owner name (column name may change)

`class` Park classification

`park_district` Park maintenance district for BCRP

`acres` Area of the park property (acres)

`osm_id` OpenStreetMap ID (node, way, or relation)

`geometry` MULTIPOLYGON geometry for park edges

Details

Updated 2023-10-16 with change to more recently updated city FeatureLayer as source for geometry.

Source

https://services1.arcgis.com/UWYHeuuJISiGmgXx/arcgis/rest/services/Map_WFL1/FeatureServer/16

park_districts *Baltimore Park Districts*

Description

Park districts for the Baltimore City Department of Recreation and Parks. District boundaries are used for park maintenance administration.

Usage

```
park_districts
```

Format

A data frame with 5 rows and 2 variables:

`name` Park district name

`geometry` MULTIPOLYGON geometry for park district boundary

planning_districts *Baltimore City Planning Districts*

Description

Administrative boundaries set by the Baltimore City Department of Planning. District planning staff are assigned to each of the planning districts.

Usage

```
planning_districts
```

Format

A data frame with 11 rows and 4 variables:

`id` Planning district area identifier

`name` Full name of the planning district

`abb` Planning district area abbreviation

`geometry` MULTIPOLYGON geometry for the planning district

Source

https://geodata.baltimorecity.gov/egis/rest/services/CityView/PlanningDistricts/MapServer/0

police_districts *Baltimore City Police Districts (1959-2022)*

Description

Baltimore City Police Districts established in 1959 and used through 2022. Note this data will be moved to a separate object for historic district boundaries in 2023.

Usage

```
police_districts
```

Format

A data frame with 9 rows and 3 variables:

`number` Police district number

`name` Police district name

`geometry` MULITPOLYGON geometry for district boundary

Source

https://geodata.baltimorecity.gov/egis/rest/services/Planning/Boundaries/MapServer/7

police_districts_2023 *Baltimore City Police Districts (2023-Current)*

Description

Baltimore City Police Districts boundaries updated in 2023.

Usage

```
police_districts_2023
```

Format

A data frame with 9 rows and 4 variables:

`id` Police district number

`name` Police district name

`name_abb` District name abbreviation

`geometry` MULTIPOLYGON geometry for district boundary

Source

https://services1.arcgis.com/UWYHeuuJISiGmgXx/arcgis/rest/services/Police_District/FeatureServer/0

`public_art` *Baltimore public art works and monuments*

Description

Data created by Eli Pousson and C. Ryan Patterson with contributions from staff and volunteers at Baltimore City Commission on Historical and Architectural Preservation, Baltimore Heritage, and the Baltimore Office of Promotion and the Arts. Updated January 18, 2023. See https://publicartbaltimore.github.io/inventory/ for more information.

Usage

```
public_art
```

Format

A data frame with 1140 rows and 35 variables:

`id` incomplete unique id column

`osm_id` OpenStreetMap identifier

`title` Artwork title

`location` Location name

`type` Artwork type

`medium` Artwork medium

`status` Artwork status

`year` Artwork status

`year_accuracy` Artwork status

`creation_dedication_date` Creation/dedication date

`primary_artist` Primary artist

`primary_artist_gender` Primary artist gender (based on name and biographical information if available)

`street_address` Street address

`city` City

`state` State

`zipcode` Zipcode

`dimensions` Artwork dimensions

`program` Commissioning program

`funding` Primary funding source

`artist_assistants` Artist assistants

`architect` Architect

`fabricator` Fabricator

`neighborhood` Neighborhood

`csa` Community Statistical Area

`council_district` Baltimore City Council District

`legislative_district` character Maryland State Legislative District

`location_desc` character Location description

`indoor_outdoor_access` Indoor/outdoor accessible

`subject_person` Subject of artworks (if work depicts a person)

`related_property` Related property name

`property_ownership` Related property ownership

`agency_or_insitution` Agency/institution responsible

`wikipedia_url` Wikipedia URL

`geometry` POINT location

`rec_centers` *Baltimore City Recreation Centers*

Description

Currently includes only publicly operated (BCRP) rec centers. Expect to add private operator facilities. Added 2023-10-19.

Usage

```
rec_centers
```

Format

A data frame with 48 rows and 18 variables:

`id` ID

`name` Center name

`name_short` Short name

`street_address` Street address

`address` Full address

`center_amenities` Center amenities

`center_assets` Center facility assets

`center_category` Center category

`center_type` Center type

`school_name` School name (school-based centers only)

`operator` Operator (BCRP only)

`recreation_district` BCRP Recreation district

`park_district` BCRP Park Maintenance District

`council_district` City Council district

`legislative_district` Maryland legislative district

`police_district` BPD Police district (maybe outdated)

`notes` Notes

`geometry` POINT geometry with center location

Source

https://services1.arcgis.com/UWYHeuuJISiGmgXx/arcgis/rest/services/recreationCenter2023/FeatureServer

---

`request_types` *311 Service Request Types for Baltimore City*

Description

A list of request types based on unique request types used between January 2019 and October 2020.

Usage

```
request_types
```

Format

A data frame with 320 rows and 1 variable:

`request_type` Service request type

Source

https://data.baltimorecity.gov/

respagency_codes *Baltimore City Real Property Responsible Agency Codes*

Description

A reference table of responsible agency codes appearing in the Baltimore City real property data used by `get_area_property()`. Updated 2023 March 29.

Usage

```
respagency_codes
```

Format

A data frame with 37 rows and 7 variables:

`name` Responsible agency name

`code` Responsible agency code

`agency_name` Baltimore City agency/commission name

`agency_abb` Baltimore City agency/commission abbreviation

`division_name` Agency division name

`active_code` Active code indicator (`FALSE` for codes that do not appear in data)

`notes` Notes on code/agency

Source

https://docs.google.com/spreadsheets/d/1Dnyp4-AZxvFPpt5Vci4NRWR9tGP99R8RaHuPCbzcGCA/edit?usp=sharing

scale_mapbaltimore *Scales for Baltimore data*

Description

Custom palettes for two package datasets: `mta_bus_lines` and `hmt_2017` (both for cluster and cluster group).

Usage

```
scale_mapbaltimore(
  palette = NULL,
  values = NULL,
  na.value = "grey50",
  aesthetics = c("color", "fill"),
  error_call = caller_env(),
  ...
)

scale_color_mapbaltimore(palette = NULL, na.value = "grey50", ...)

scale_fill_mapbaltimore(palette = NULL, na.value = "grey50", ...)
```

Arguments

palette	Options include "mta_bus", "hmt_2017", "hmt_cluster", "cluster", "hmt_cluster_group", or "cluster_group", Default: NULL
values	a set of aesthetic values to map data values to. The values will be matched in order (usually alphabetical) with the limits of the scale, or with breaks if provided. If this is a named vector, then the values will be matched based on the names instead. Data values that don't match will be given na.value.
na.value	Defaults to "grey50"
aesthetics	Character string or vector of character strings listing the name(s) of the aesthetic(s) that this scale works with. This can be useful, for example, to apply colour settings to the colour and fill aesthetics at the same time, via aesthetics = c("colour", "fill").
error_call	The execution environment of a currently running function, e.g. caller_env(). The function will be mentioned in error messages as the source of the error. See the call argument of abort() for more information.
...	Arguments passed on to discrete_scale

palette A palette function that when called with a single integer argument (the number of levels in the scale) returns the values that they should take (e.g., scales::hue_pal()).

limits One of:

- NULL to use the default scale values
- A character vector that defines possible values of the scale and their order
- A function that accepts the existing (automatic) values and returns new ones. Also accepts rlang lambda function notation.

drop Should unused factor levels be omitted from the scale? The default, TRUE, uses the levels that appear in the data; FALSE uses all the levels in the factor.

na.translate Unlike continuous scales, discrete scales can easily show missing values, and do so by default. If you want to remove missing values from a discrete scale, specify na.translate = FALSE.

`scale_name` The name of the scale that should be used for error messages associated with this scale.

`name` The name of the scale. Used as the axis or legend title. If `waiver()`, the default, the name of the scale is taken from the first mapping used for that aesthetic. If `NULL`, the legend title will be omitted.

`labels` One of:

- `NULL` for no labels
- `waiver()` for the default labels computed by the transformation object
- A character vector giving labels (must be same length as `breaks`)
- An expression vector (must be the same length as breaks). See ?plotmath for details.
- A function that takes the breaks as input and returns labels as output. Also accepts rlang lambda function notation.

`guide` A function used to create a guide or its name. See `guides()` for more information.

`super` The super class to use for the constructed scale

Examples

```
## Not run:
if (interactive()) {
  library(ggplot2)

  ggplot(data = dplyr::filter(mta_bus_lines, frequent)) +
    geom_sf(aes(color = route_abb), alpha = 0.5, size = 2) +
    scale_mapbaltimore(palette = "bus") +
    theme_minimal()

  ggplot(data = hmt_2017) +
    geom_sf(aes(fill = cluster_group, color = cluster_group)) +
    scale_mapbaltimore(palette = "cluster_group") +
    theme_minimal()
}

## End(Not run)
```

---

schools_21stc *Baltimore 21st Century Schools*

Description

Schools with buildings in the 21st Century Schools Program. Updated 2022 October 13. This data may contain some out-dated or inaccurate information. See `buildings_21stc` for building-level information (including more accurate locations).

Usage

```
schools_21stc
```

Format

A data frame with 29 rows and 24 variables:

`school_name` School name

`school_number` School number

`nces_number` NCES number

`grade_band` Grade bane

`url` School website URL

`year` 21st Century School renovation/replacement complete

`type` 21st Century School project type

`bldg_budget_approx` Approximate building budget

`status_21c` 21st Century School project status

`status_inspire` INSPRE Plan status

`inspire_plan` Related INSPIRE Plan

`occupancy_month` Building occupancy month

`occupancy_year` Building occupancy year

`address` Street address

`city` City

`state` State

`zip` Zipcode

`phone` School phone number

`alt_school_name` Alternate school name

`bldg_name` Building name (if applicable)

`alt_name` Alternate/former names (if applicable)

`lon` Longitude

`lat` Latitude

`geometry` POINT geometry for school locations

Details

https://baltimore21stcenturyschools.org/school-projects

set_map_theme *Set default map theme*

Description

Set a map theme using ggplot2::theme_set() and default for geom_label using ggplot2::update_geom_defaults(). Optionally hides axis text and labels.

Usage

```
set_map_theme(map_theme = NULL, show_axis = FALSE)
```

Arguments

map_theme	ggplot2 theme. Optional. Defaults to ggplot2::theme_minimal()
show_axis	Logical. If TRUE, keep theme axis formatting. If FALSE, hide the panel grid, axis title, and axis text.

streets *Baltimore City Street Center Lines*

Description

Street center line data for public streets in Baltimore City, Maryland. Data is used by the get_streets() function.

Usage

```
streets
```

Format

Simple feature collection with 48,473 features and 23 fields.

type ...

subtype ...

subtype_label ...

dirpre ...

feanme ...

featype ...

dirsuf ...

fraddl ...

toaddl ...

fraddr ...

toaddr ...

fraddla ...

toaddla ...

fraddra ...

toaddra ...

leftzip ...

rightzip ...

fullname ...

sha_class ...

sha_class_label ...

blocktext ...

block_num ...

geometry ...

Source

https://dotgis.baltimorecity.gov/arcgis/rest/services/DOT_Map_Services/DOT_Basemap/MapServer/7

---

wards_1797_1918 *Historic Ward Boundaries, 1797-1918 for Baltimore City*

Description

Historic ward boundary data from 1797 to 1918. Derived from KML data provided by the Baltimore City Archives.

Usage

```
wards_1797_1918
```

Format

A data frame with 245 rows and 4 variables:

`year` Earliest effective year of ward boundary

`name` Ward name

`number` Ward number

`geometry` MULTIPOLYGON geometry for ward boundary

Source

https://msa.maryland.gov/bca/wards/index.html

xwalk_block2tract — *U.S. Census Block-to-Tract Crosswalk with 2010 Block Household Population*

Description

A crosswalk file used to generate `xwalk_neighborhood2tract`.

Usage

```
xwalk_block2tract
```

Format

A data frame with 13598 rows and 3 variables:

`block` Block GeoID

`tract` Tract GeoID

`households` Block household population

xwalk_csa2nsa — *Community Statistical Area (CSA)-to-Neighborhood Statistical Area (NSA) Crosswalk*

Description

A crosswalk to match Community Statistical Areas to Neighborhood Statistical Areas. Both a Neighborhood Statistical Area name and neighborhood name are provided, with the NSA name matching the crosswalk file provided by BNIA-JFI and the neighborhood name matching the neighborhoods data included with the mapbaltimore package. NSA boundaries may overlap over several CSAs. When more than 50% of a NSA falls within a particular community it is assigned to that community. No NSAs in these files are assigned to more than one community.

Usage

```
xwalk_csa2nsa
```

Format

A data frame with 278 rows and 4 variables:

`id` Community Statistical Area id number

`csa` Community Statistical Area name

`nsa` Neighborhood Statistical Area name

`neighborhood` Neighborhood name

Source

https://bniajfi.org/mapping-resources/

`xwalk_neighborhood2tract` *Neighborhood-to-U.S. Census Tract Crosswalk*

Description

Share of total households is based on the proportion of U.S. Census tract population within the named neighborhood based on overlapping U.S. Census Block groups.

Usage

```
xwalk_neighborhood2tract
```

Format

A data frame with 551 rows and 4 variables:

`name` Neighborhood name

`geoid` GeoID for U.S. Census tract

`tract` Tract number

`weight_households` Share of total households in neighborhood and U.S. Census tract (based on 2010 decennial Census). Variable code is "H013001".

`weight_units` Share of occupied housing units in neighborhood and U.S. Census tract (based on 2020 decennial Census PL-94171 redistricting data). Variable code is "H1_002N".

`xwalk_zip2csa` *Zipcode-to-Community Statistical Area (NSA) Crosswalk*

Description

A crosswalk to match zipcodes to Community Statistical Areas.

Usage

```
xwalk_zip2csa
```

Format

A data frame with 119 rows and 3 variables:

`zip` Zipcode

`csa` Community Statistical Area name

`id` Community Statistical Area id number

Source

https://bniajfi.org/mapping-resources/

`zoning` *Baltimore City Zoning Code*

Description

The Baltimore City Zoning Code is administered by the Baltimore City Department of Housing and Community Development (HCD) Office of the Zoning Administrator. This office supports the Board of Municipal Zoning Appeals (BMZA).

Usage

```
zoning
```

Format

A data frame with 2,406 rows and 4 variables:

`zoning` Zoning designation code

`overlay` Overlay zone designation

`label` Label combining zoning and overlay zoning codes

`category_zoning` Zoning code category

`name_zoning` Zoning code name

`category_overlay` Overlay code category

`name_overlay` Overlay zoning name

`geometry` MULTIPOLYGON geometry for zoning areas

Source

https://geodata.baltimorecity.gov/egis/rest/services/Planning/Boundaries_and_Plans/MapServer/20

Index

∗ **datasets**
adopted_plans, 4
balt_tbl_labs, 14
baltimore_bbox, 4
baltimore_block_groups, 6
baltimore_blocks, 5
baltimore_census_xwalk, 6
baltimore_city, 7
baltimore_city_detailed, 8
baltimore_gis_index, 8
baltimore_mihp, 9
baltimore_msa_counties, 10
baltimore_msa_water, 11
baltimore_pumas, 12
baltimore_tracts, 13
baltimore_water, 13
bcps_programs, 15
bcps_zones, 15
buildings_21stc, 16
chap_districts, 18
circulator_routes, 19
circulator_stops, 20
congressional_districts, 21
council_districts, 22
csas, 22
explore_baltimore, 23
hmt_2017, 49
inspire_plans, 50
legislative_districts, 53
legislative_districts_2012, 54
main_streets, 55
mta_bus_lines, 62
mta_bus_stops, 63
mta_light_rail_lines, 64
mta_light_rail_stations, 65
mta_marc_lines, 66
mta_marc_stations, 66
mta_subway_lines, 67
mta_subway_stations, 68
named_intersections, 69
neighborhoods, 69
neighborhoods_2020, 70
park_districts, 72
parks, 71
planning_districts, 72
police_districts, 73
police_districts_2023, 73
public_art, 74
rec_centers, 75
request_types, 76
respagency_codes, 77
schools_21stc, 79
streets, 81
wards_1797_1918, 82
xwalk_block2tract, 83
xwalk_csa2nsa, 83
xwalk_neighborhood2tract, 84
xwalk_zip2csa, 84
zoning, 85

abort(), *78*
adopted_plans, 4
as_bbox(), *18*, *41*

balt_tbl_labs, 14
baltimore_bbox, 4
baltimore_block_groups, 6
baltimore_blocks, 5
baltimore_census_xwalk, 6
baltimore_city, 7
baltimore_city_detailed, 8
baltimore_gis_index, 8
baltimore_mihp, 9
baltimore_msa_counties, 10
baltimore_msa_water, 11
baltimore_pumas, 12
baltimore_tracts, 13
baltimore_water, 13
bcps_programs, 15

bcps_zones, 15
buildings_21stc, 16

cache_baltimore_data, 17
cache_baltimore_property
(cache_baltimore_data), 17
cache_edge_of_pavement
(cache_baltimore_data), 17
cache_msa_streets
(cache_baltimore_data), 17
chap_districts, 18
check_area, 19
circulator_routes, 19
circulator_stops, 20
cli::cli_progress_along(), *42*
congressional_districts, 21, *26*
council_districts, 22, *26*
csas, 22, *26*

df_to_sf(), *31*, *34*, *37*, *42*
discrete_scale, *78*

esri2sf::esri2df(), *41*
esri2sf::esri2sf, *37*
esri2sf::esri2sf(), *41*
explore_baltimore, 23

filter_streets, 24
format_property_data
(get_area_property), 34

get_area, 25
get_area(), *25, 26*
get_area_911_calls, 27
get_area_911_calls(), *27*
get_area_batch (get_batch), 43
get_area_bcps_programs, 28
get_area_census_geography, 29
get_area_citations, 29
get_area_crime, 30
get_area_data, 32
get_area_permits, 33
get_area_property, 34
get_area_property(), *77*
get_area_requests, 35
get_area_streets, 38
get_area_streets(), *48*
get_area_vacants, 39
get_area_zoning, 40
get_asp(), *31*, *34, 35*, *37*, *41*
get_baltimore_area (get_area), 25
get_baltimore_area(), *25, 26*
get_baltimore_esri_data, 41
get_baltimore_esri_data(), *8*
get_baltimore_worker_flows, 42
get_batch, 43
get_data_batch (get_batch), 43
get_esri_layers(), *41*
get_esri_metadata(), *42*
get_intersection, 46
get_intersection(), *69*
get_nearby_areas, 47
get_neighborhood (get_area), 25
get_neighborhood(), *26*
get_streets, 47
get_streets(), *81*
getACS::make_area_xwalk(), *7*
getACS::use_area_xwalk(), *6*
getdata::get_esri_data, *41*
getdata::get_esri_data(), *27*, *30*, *34, 35*, *41, 42*
getdata::get_location(), *25*
getdata::get_location_data(), *24*
ggplot2::geom_sf(), *52*
ggplot2::theme_minimal(), *81*
ggplot2::theme_set(), *81*
ggplot2::update_geom_defaults(), *81*
guides(), *79*

hmt_2017, 49

inspire_plans, 50

janitor::make_clean_names(), *42*

lambda, *78, 79*
layer_area_property, 51
layer_area_streets, 52
legislative_districts, *26*, 53
legislative_districts_2012, 54

main_streets, 55
map_area_bcps_programs, 55
map_area_highlighted, 56
map_area_in_areas, 57
map_area_in_city, 57
map_area_mta_services, 58
map_area_parks, 59

map_area_property, 60
map_area_zoning, 61
maryland_open_data_api_key, 61
mta_bus_lines, 62
mta_bus_stops, 63
mta_light_rail_lines, 64
mta_light_rail_stations, 65
mta_marc_lines, 66
mta_marc_stations, 66
mta_subway_lines, 67
mta_subway_stations, 68

named_intersections, 69
neighborhoods, *26*, 69
neighborhoods_2020, 70

park_districts, *26*, 72
parks, 71
planning_districts, *26*, 72
police_districts, *26*, 73
police_districts_2023, 73
public_art, 74

rec_centers, 75
request_types, 76
respagency_codes, 77

scale_color_mapbaltimore
(scale_mapbaltimore), 77
scale_fill_mapbaltimore
(scale_mapbaltimore), 77
scale_mapbaltimore, 77
scales::hue_pal(), *78*
schools_21stc, 79
set_map_theme, 81
sf::st_crop(), *28*, *33*, *35*, *39*, *40*, *51*
sf::st_intersection(), *28*, *33*, *35*, *39*, *40*, *52*, *60*
sf::st_intersects(), *32*
sf::st_read(), *32*
sf::st_union(), *26*, *28*, *31–33*, *35*, *36*, *39*, *40*, *51*, *61*
sf_to_df(), *31*, *34*, *37*
sfext::get_area(), *25*
show_cached_files
(cache_baltimore_data), 17
st_trim(), *31*, *34*, *37*
streets, *48*, 81
suppressMessages(), *42*

tidygeocoder::geo(), *26*
tigris::county_subdivisions(), *7*
tigris::tracts(), *43*

wards_1797_1918, 82

xwalk_block2tract, 83
xwalk_csa2nsa, 83
xwalk_neighborhood2tract, 84
xwalk_zip2csa, 84

zoning, 85