

---

# **politico-civic-geography Documentation**

*Release 0.5.1*

**POLITICO**

**Jan 20, 2019**



---

Read the docs:

---

<b>1</b>	<b>Why this?</b>	<b>3</b>
<b>2</b>	<b>Quickstart</b>	<b>5</b>
2.1	Requirements . . . . .	5
2.2	Install . . . . .	5
2.3	Configure . . . . .	5
2.4	Bootstrap . . . . .	6
2.5	Run . . . . .	7
<b>3</b>	<b>Models</b>	<b>9</b>
3.1	Division . . . . .	9
3.2	DivisionLevel . . . . .	10
3.3	Geometry . . . . .	10
3.4	IntersectRelationship . . . . .	10
3.5	Point . . . . .	11
3.6	PointLabelOffset . . . . .	11
<b>4</b>	<b>Management</b>	<b>13</b>
4.1	bootstrap_geography . . . . .	13
4.2	Overriding bootstrapped geometry . . . . .	14
4.3	bake_geography . . . . .	14
<b>5</b>	<b>Links</b>	<b>17</b>





Home › Geography › Geometries

Select geometry to change

ADD GEOMETRY +

Q

Search







51 results (2066 total)

Action:

-----

Go

0 of 51 selected

<input type="checkbox"/> DIVISION	SMALL PREVIEW	MAP LEVEL	SERIES
<input type="checkbox"/> Washington		county	2017
<input type="checkbox"/> Arkansas		county	2017
<input type="checkbox"/> Alabama		county	2017
<input type="checkbox"/> Minnesota		county	2017
<input type="checkbox"/> Vermont		county	2017
<input type="checkbox"/> Virginia		county	2017

FILTER

By series

All

2016

2017

By division level

All

country

state

county

district

township

precinct

By map level

All

country

state

county

district

township

precinct



# CHAPTER 1

---

## Why this?

---

Geography is the foundation of almost all civic data. This app models geographic data for U.S. political divisions, including states, congressional districts, counties and townships. It also includes loaders to bootstrap your database, sourced from U.S. Census cartographic boundary files, and exporters to build out a full set of boundary files in TopoJSON format, which we use at POLITICO to create data maps.

In short, politico-civic-geography is a one-stop shop for creating and managing political geography and maps. That's why we use it as the basis for our election rig, dataviz apps and other political projects. With it we keep a consistent source of geographic data from the Census and can easily update across our applications each year when new boundary files are released.

While politico-civic-geography is focused primarily on U.S. political divisions, its data model borrows heavily from more generic specifications, especially the [Open Civic Data](#) project. (We welcome contributions of loaders for other political contexts!)

This app is part of our larger [politico-civic](#) project. [Read the docs](#) for more information.





### 2.1 Requirements

- topojson
- PostgreSQL 9.4
- Django 2.0

### 2.2 Install

```
$ pip install politico-civic-geography
```

### 2.3 Configure

1. Add the app and Django Rest Framework to the installed apps in your project settings and configure app specific settings.

```
# settings.py

INSTALLED_APPS = [
    # ...
    "rest_framework",
    "geography",
]

CENSUS_API_KEY = os.getenv("CENSUS_API_KEY")
GEOGRAPHY_AWS_ACCESS_KEY_ID = os.getenv("AWS_ACCESS_KEY_ID")
GEOGRAPHY_AWS_SECRET_ACCESS_KEY = os.getenv("AWS_SECRET_ACCESS_KEY")
GEOGRAPHY_AWS_S3_BUCKET = os.getenv("AWS_S3_BUCKET")
```

(continues on next page)

(continued from previous page)

```
GEOGRAPHY_AWS_REGION = "us-east-1" # default
GEOGRAPHY_AWS_S3_UPLOAD_ROOT = "elections" # default
GEOGRAPHY_AWS_ACL = "public-read" # default
GEOGRAPHY_AWS_CACHE_HEADER = "max-age=3600" # default
GEOGRAPHY_API_AUTHENTICATION_CLASS = "rest_framework.authentication.
↳BasicAuthentication" # default
GEOGRAPHY_API_PERMISSION_CLASS = "rest_framework.permissions.IsAdminUser" #
↳default
GEOGRAPHY_API_PAGINATION_CLASS = "geography.pagination.ResultsPagination" #
↳default
```

2. Add the app to your project's `urlconf`.

```
# urls.py

urlpatterns = [
    # ...
    path('geography', include('geography.urls')),
]
```

- ### 3. Migrate you DB.

```
$ python manage.py migrate
```

## 2.4 Bootstrap

Bootstrap your database with geographic data from the U.S. Census Bureau. Running this command will create Geography and Geometry fixtures for states, counties, congressional districts and townships.

```
$ python manage.py bootstrap_geography
```

See [Management](#) for more details on using this command and on baking geometry TopoJSON files to AWS S3.

[illegible]

## 2.5 Run

Start the server to see your new fixtures in Django's admin.

```
$ python manage.py runserver
```

Home › Geography › Geometries







Select geometry to change

ADD GEOMETRY +

51 results (2066 total)

Action: 

0 of 51 selected

<input type="checkbox"/>	DIVISION	SMALL PREVIEW	MAP LEVEL	SERIES
<input type="checkbox"/>	Washington		county	2017
<input type="checkbox"/>	Arkansas		county	2017
<input type="checkbox"/>	Alabama		county	2017
<input type="checkbox"/>	Minnesota		county	2017
<input type="checkbox"/>	Vermont		county	2017
<input type="checkbox"/>	Virginia		county	2017

**FILTER**

By series

- All
- 2016
- 2017

By division level

- All
- country
- state
- county
- district
- township
- precinct

By map level

- All
- country
- state
- county
- district
- township
- precinct



Geography models represent all geographic political divisions in the United States, their relationships and their geometric boundaries.

## 3.1 Division

**class** geography.models.Division(\*args, \*\*kwargs)

A political or administrative geography.

For example, a particular state, county, district, precinct or municipality.

### Parameters

- **id** (*UUIDField*) – Id
- **uid** (*CharField*) – Uid
- **slug** (*SlugField*) – Slug
- **name** (*CharField*) – Name
- **label** (*CharField*) – Label
- **short\_label** (*CharField*) – Short label
- **parent** (ForeignKey to Division) – Parent
- **level** (ForeignKey to DivisionLevel) – Level
- **code** (*CharField*) – Code representing a geography: FIPS code for states and counties, district number for districts, precinct number for precincts, etc.
- **code\_components** (*JSONField*) – Component parts of code
- **effective** (*BooleanField*) – Effective
- **effective\_start** (*DateTimeField*) – Effective start
- **effective\_end** (*DateTimeField*) – Effective end

- **intersecting** (*ManyToManyField*) – Intersecting divisions intersect this one geographically but do not necessarily have a parent/child relationship. The relationship between a congressional district and a precinct is an example of an intersecting relationship.

## 3.2 DivisionLevel

**class** geography.models.**DivisionLevel** (\*args, \*\*kwargs)

Level of government or administration at which a division exists.

For example, federal, state, district, county, precinct, municipal.

### Parameters

- **id** (*UUIDField*) – Id
- **uid** (*CharField*) – Uid
- **slug** (*SlugField*) – Slug
- **name** (*CharField*) – Name
- **parent** (*ForeignKey* to *DivisionLevel*) – Parent

## 3.3 Geometry

**class** geography.models.**Geometry** (\*args, \*\*kwargs)

The spatial representation (in topoJSON) of a Division.

### Parameters

- **id** (*UUIDField*) – Id
- **division** (*ForeignKey* to *Division*) – Division
- **subdivision\_level** (*ForeignKey* to *DivisionLevel*) – Subdivision level
- **simplification** (*FloatField*) – Minimum quantile of planar triangle areas for simplifying topojson.
- **topojson** (*JSONField*) – Topojson
- **source** (*URLField*) – Link to the source of this geography data.
- **series** (*CharField*) – Year of boundary series, e.g., 2016 TIGER/Line files.
- **effective** (*BooleanField*) – Effective
- **effective\_start** (*DateField*) – Effective start
- **effective\_end** (*DateField*) – Effective end

## 3.4 IntersectRelationship

**class** geography.models.**IntersectRelationship** (\*args, \*\*kwargs)

Each IntersectRelationship instance represents one side of a paired relationship between intersecting divisions.

The intersection field represents the decimal proportion of the to\_division that intersects with the from\_division. It's useful for apportioning counts between the areas, for example, population statistics from census data.

**Parameters**

- **id** (*AutoField*) – Id
- **from\_division** (*ForeignKey to Division*) – From division
- **to\_division** (*ForeignKey to Division*) – To division
- **intersection** (*DecimalField*) – The portion of the to\_division that intersects this division.

## 3.5 Point

```
class geography.models.Point(*args, **kwargs)
```

A point is a city.

**Parameters**

- **id** (*AutoField*) – Id
- **geometry** (*ForeignKey to Geometry*) – Geometry
- **lat** (*FloatField*) – Latitude coordinate in decimal degrees.
- **lon** (*FloatField*) – Longitude coordinate in decimal degrees.
- **attributes** (*JSONField*) – Miscellaneous attributes on the point.
- **threshold** (*PositiveSmallIntegerField*) – A threshold in pixels above which to display this point.
- **label** (*CharField*) – Label

## 3.6 PointLabelOffset

```
class geography.models.PointLabelOffset(*args, **kwargs)
```

Offsets used to display a Point's label.

**Parameters**

- **id** (*AutoField*) – Id
- **point** (*ForeignKey to Point*) – Point
- **x** (*SmallIntegerField*) – Lateral offset in pixels.
- **y** (*SmallIntegerField*) – Vertical offset in pixels.
- **threshold** (*PositiveSmallIntegerField*) – A threshold in pixels above which to apply this offset.





### 4.1 bootstrap\_geography

This command will download shapefiles from the U.S. Census Bureau, process them and create a complete set of Geography and Geometry fixtures for states, congressional districts, counties and townships in your database.

```
usage: manage.py bootstrap_geography [-h] [--year YEAR] [--congress CONGRESS]
                                     [--nationThreshold NATIONTHRESHOLD]
                                     [--stateThreshold STATETHRESHOLD]
                                     [--districtThreshold DISTRICTTHRESHOLD]
                                     [--countyThreshold COUNTYTHRESHOLD]
                                     [--version] [-v {0,1,2,3}]
                                     [--settings SETTINGS]
                                     [--pythonpath PYTHONPATH] [--traceback]
                                     [--no-color]
```

Downloads and bootstraps geographic data for states, congressional districts, counties and townships from the U.S. Census Bureau simplified cartographic boundary files.

optional arguments:

-h, --help	show this help message and exit
--year YEAR	Specify year of shapefile series (default, 2017)
--congress CONGRESS	Specify congress of district shapefile series (default, 115)
--nationThreshold NATIONTHRESHOLD	Simplification threshold value for nation topojson (default, 0.005)
--stateThreshold STATETHRESHOLD	Simplification threshold value for state topojson (default, 0.05)
--districtThreshold DISTRICTTHRESHOLD	Simplification threshold value for district topojson (default, 0.08)

(continues on next page)

(continued from previous page)

```

--countyThreshold COUNTYTHRESHOLD
    Simplification threshold value for county topojson
    (default, 0.075)
--version
    show program's version number and exit
-v {0,1,2,3}, --verbosity {0,1,2,3}
    Verbosity level; 0=minimal output, 1=normal output,
    2=verbose output, 3=very verbose output
--settings SETTINGS
    The Python path to a settings module, e.g.
    "myproject.settings.main". If this isn't provided, the
    DJANGO_SETTINGS_MODULE environment variable will be
    used.
--pythonpath PYTHONPATH
    A directory to add to the Python path, e.g.
    "/home/djangoprojects/myproject".
--traceback
    Raise on CommandError exceptions
--no-color
    Don't colorize the command output.

```

## 4.2 Overriding bootstrapped geometry

In some cases, we need to overwrite geometry. In those cases, we generally package source shapefiles with this module and write an additional management command to be run *after* bootstrapping geography from the Census.

Those commands are called by the format: `bootstrap_geom_<id>`

## 4.3 bake\_geography

This command will export state and congressional district boundary files in TopoJSON to an AWS S3 bucket.

```

usage: manage.py bake_geography [-h] [--year YEAR] [--version] [-v {0,1,2,3}]
                                [--settings SETTINGS]
                                [--pythonpath PYTHONPATH] [--traceback]
                                [--no-color]
                                states [states ...]

```

Uploads topojson files by state and district to an Amazon S3 bucket.

positional arguments:

```

    states
        States to export by FIPS code. Use 00 to export all
        geographies.

```

optional arguments:

```

-h, --help
    show this help message and exit
--year YEAR
    Specify year of shapefile series (default, 2017)
--version
    show program's version number and exit
-v {0,1,2,3}, --verbosity {0,1,2,3}
    Verbosity level; 0=minimal output, 1=normal output,
    2=verbose output, 3=very verbose output
--settings SETTINGS
    The Python path to a settings module, e.g.
    "myproject.settings.main". If this isn't provided, the
    DJANGO_SETTINGS_MODULE environment variable will be
    used.
--pythonpath PYTHONPATH

```

(continues on next page)

(continued from previous page)

	A directory to add to the Python path, e.g. "/home/djangoprojects/myproject".
--traceback	Raise on CommandError exceptions
--no-color	Don't colorize the command output.



## CHAPTER 5

---

### Links

---

- Code: <https://github.com/The-Politico/politico-civic-geography>
- Issues: <https://github.com/The-Politico/politico-civic-geography/issues>



### D

Division (class in geography.models), [9](#)

DivisionLevel (class in geography.models), [10](#)

### G

Geometry (class in geography.models), [10](#)

### I

IntersectRelationship (class in geography.models), [10](#)

### P

Point (class in geography.models), [11](#)

PointLabelOffset (class in geography.models), [11](#)