# Map Configuration and Styling on Geo Maps in Python

How to configure and style base maps for outline-based Geo Maps.



## Tile Maps vs Outline Maps

Plotly supports two different kinds of maps:

- Tile-based maps (https://en.wikipedia.org/wiki/Tiled_web_map)

If your figure is created with a px.scatter_map, px.scatter_mapbox, px.line_map, px.line_mapbox, px.choropleth_map, px.choropleth_mapbox, px.density_map, or px.density_mapbox function or otherwise contains one or more traces of type go.Scattermap, go.Scattermapbox, go.Choroplethmap, go.Choroplethmapbox, go.Densitymap, or go.Densitymapbox, the layout.map object in your figure contains configuration information for the map itself.

- **Outline-based maps**

Geo maps are outline-based maps. If your figure is created with a px.scatter_geo, px.line_geo or px.choropleth function or otherwise contains one or more traces of type go.Scattergeo or go.Choropleth, the layout.geo object in your figure contains configuration information for the map itself.

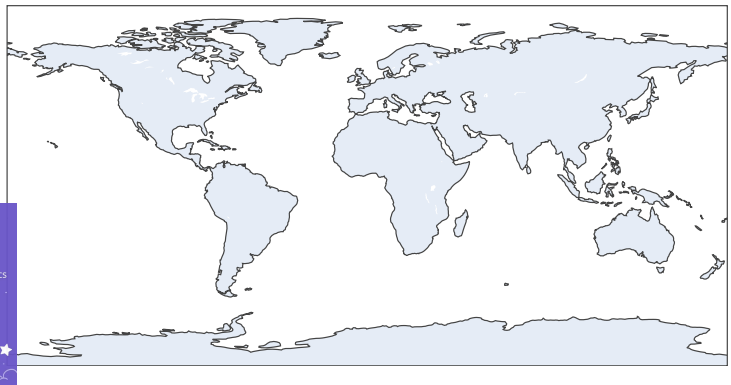
> This page documents **Geo outline-based maps**, and the Tile Map Layers documentation (/python/tile-map-layers/) describes how to configure tile-based maps.

**Note:** Plotly Express cannot create empty figures, so the examples below mostly create an "empty" map using fig = go.Figure(go.Scattergeo()). That said, every configuration option here is equally applicable to non-empty maps created with the Plotly Express px.scatter_geo, px.line_geo or px.choropleth functions.

## Physical Base Maps

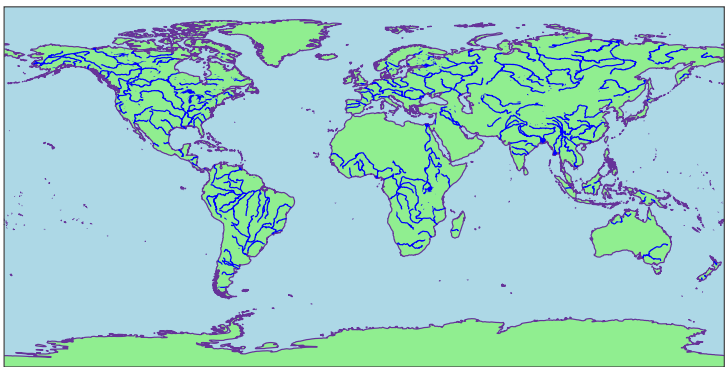
Plotly Geo maps have a built-in base map layer composed of "physical" and "cultural" (i.e. administrative border) data from the Natural Earth Dataset (https://www.naturalearthdata.com/downloads/). Various lines and area fills can be shown or hidden, and their color and line-widths specified. In the default plotly template (/python/templates/), a map frame and physical features such as a coastal outline and filled land areas are shown, at a small-scale 1:110m resolution:

```
import plotly.graph_objects as go

fig = go.Figure(go.Scattergeo())
fig.update_layout(height=300, margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```



Here is a map with all physical features enabled and styled, at a larger-scale 1:50m resolution:

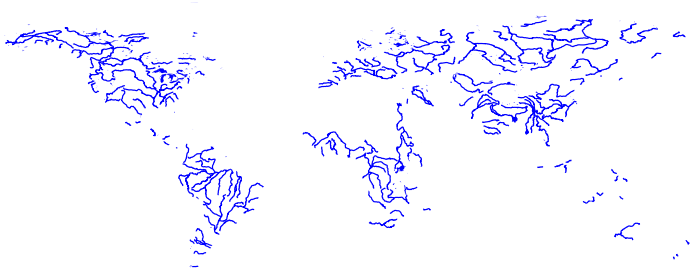
```
import plotly.graph_objects as go

fig = go.Figure(go.Scattergeo())
fig.update_geos(
    resolution=50,
    showcoastlines=True, coastlinecolor="RebeccaPurple",
    showland=True, landcolor="LightGreen",
    showocean=True, oceancolor="LightBlue",
    showlakes=True, lakecolor="Blue",
    showrivers=True, rivercolor="Blue"
)
fig.update_layout(height=300, margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```



## Disabling Base Maps

In certain cases, such as large scale choropleth maps (/python/choropleth-maps/), the default physical map can be distracting. In this case the `layout.geo.visible` attribute can be set to `False` to hide all base map attributes except those which are explicitly set to true. For example in the following map we hide all physical features except rivers and lakes, neither of which are shown by default:

```
import plotly.graph_objects as go

fig = go.Figure(go.Scattergeo())
fig.update_geos(
    visible=False,
    resolution=50,
    showlakes=True, lakecolor="Blue",
    showrivers=True, rivercolor="Blue"
)
fig.update_layout(height=300, margin={"r":0,"t":0,"l":0,"b":0})

fig.show()
```



## Cultural Base Maps

In addition to physical base map features, a "cultural" base map is included which is composed of country borders and selected sub-country borders such as states.

**Note and disclaimer:** cultural features are by definition subject to change, debate and dispute. Plotly includes data from Natural Earth "as-is" and defers to the Natural Earth policy regarding disputed borders (https://www.naturalearthdata.com/downloads/50m-cultural-vectors/50m-admin-0-countries-2/) which read:

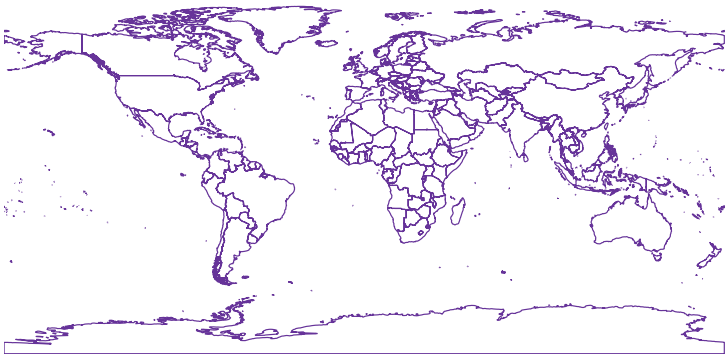
> Natural Earth Vector draws boundaries of countries according to defacto status. We show who actually controls the situation on the ground.

**To create a map with your own cultural features** please refer to our choropleth documentation (/python/choropleth-maps/).

ıg
ıb-Units
Grid

Here is a map with only cultural features enabled and styled, at a 1:50m resolution, which includes only country boundaries. See below for country sub-unit cultural base map features:

```
import plotly.graph_objects as go

fig = go.Figure(go.Scattergeo())
fig.update_geos(
    visible=False, resolution=50,
    showcountries=True, countrycolor="RebeccaPurple"
)
fig.update_layout(height=300, margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```



## Map Projections

Geo maps are drawn according to a given map projection (https://en.wikipedia.org/wiki/Map_projection) that flattens the Earth's roughly-spherical surface into a 2-dimensional space. In the following examples, we show the 'orthographic' and 'natural earth' projections, two of the many projection types available. For a full list of available projection types, see the layout.geo reference documentation (https://plotly.com/python/reference/layout/geo/#layout-geo-projection-type).

```
import plotly.graph_objects as go

fig = go.Figure(go.Scattergeo())
fig.update_geos(projection_type="orthographic")
fig.update_layout(height=300, margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```

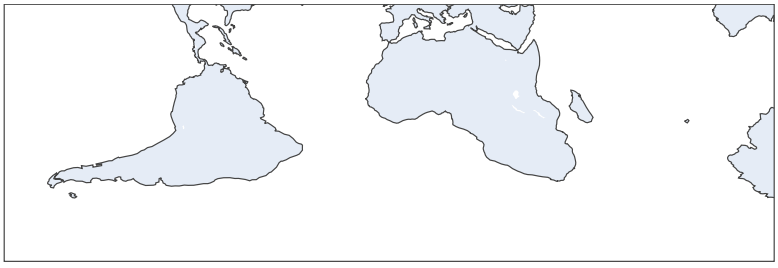
```
import plotly.graph_objects as go

fig = go.Figure(go.Scattergeo())
fig.update_geos(projection_type="natural earth")
fig.update_layout(height=300, margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```

ıg
ıb-Units
Grid

Map projections can be rotated using the layout.geo.projection.rotation attribute, and maps can be translated using the layout.geo.center attributed, as well as truncated to a certain longitude and latitude range using the layout.geo.lataxis.range and layout.geo.lonaxis.range.

The map below uses all of these attributes to demonstrate the types of effect this can yield:

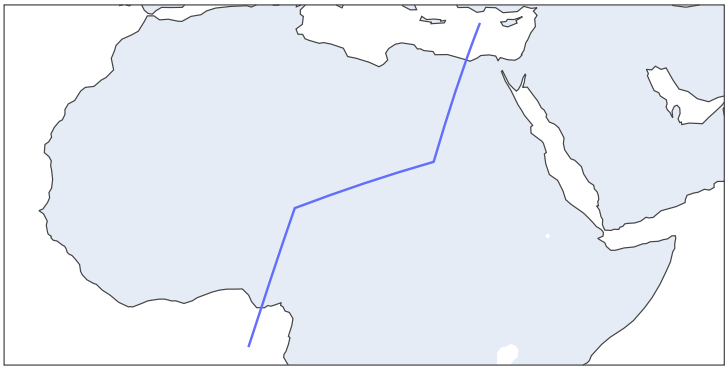
```
import plotly.graph_objects as go

fig = go.Figure(go.Scattergeo())
fig.update_geos(
    center=dict(lon=-30, lat=-30),
    projection_rotation=dict(lon=30, lat=30, roll=30),
    lataxis_range=[-50,20], lonaxis_range=[0, 200]
)
fig.update_layout(height=300, margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```



## Automatic Zooming or Bounds Fitting

The layout.geo.fitbounds attribute can be set to locations to automatically set the center and latitude and longitude range according to the data being plotted. See the choropleth maps (/python/choropleth-maps/) documentation for more information.

```
import plotly.express as px

fig = px.line_geo(lat=[0,15,20,35], lon=[5,10,25,30])
fig.update_geos(fitbounds="locations")
fig.update_layout(height=300, margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```



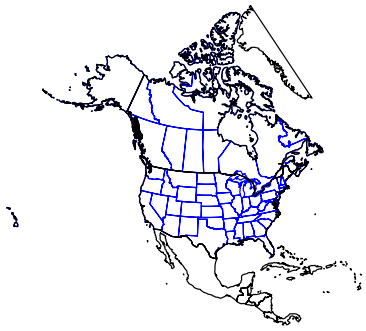
ıg
ıb-Units
Grid

## Named Map Scopes and Country Sub-Units

In addition, the named "scope" of a map defines a sub-set of the earth's surface to draw. Each scope has a *default projection type, center and roll, as well as bounds*, and certain scopes contain country sub-unit cultural layers certain resolutions, such as `scope="north america"` at `resolution=50` which contains US state and Canadian province boundaries.

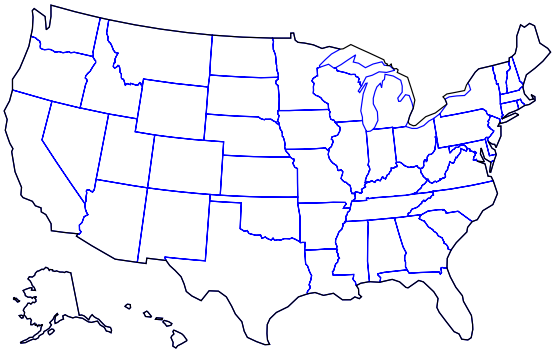
The available scopes are: `'world'`, `'usa'`, `'europe'`, `'asia'`, `'africa'`, `'north america'`, `'south america'`.

```
import plotly.graph_objects as go

fig = go.Figure(go.Scattergeo())
fig.update_geos(
    visible=False, resolution=50, scope="north america",
    showcountries=True, countrycolor="Black",
    showsubunits=True, subunitcolor="Blue"
)
fig.update_layout(height=300, margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```



The `"usa"` scope contains state boundaries at both resolutions, and uses the special `'albers usa'` projection which moves Alaska and Hawaii closer to the "lower 48 states" to reduce projection distortion and produce a more compact map.

```
import plotly.graph_objects as go

fig = go.Figure(go.Scattergeo())
fig.update_geos(
    visible=False, resolution=110, scope="usa",
    showcountries=True, countrycolor="Black",
    showsubunits=True, subunitcolor="Blue"
)
fig.update_layout(height=300, margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```

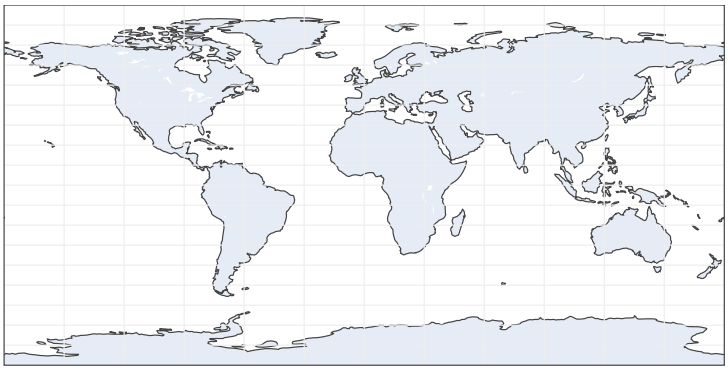


ıg
ıb-Units
Grid

## Graticules (Latitude and Longitude Grid Lines)

A graticule can be drawn using layout.geo.lataxis.showgrid and layout.geo.lonaxis.showgrid with options similar to 2d cartesian ticks (/python/axes/).

```
import plotly.graph_objects as go

fig = go.Figure(go.Scattergeo())
fig.update_geos(lataxis_showgrid=True, lonaxis_showgrid=True)
fig.update_layout(height=300, margin={"r":0,"t":0,"l":0,"b":0})
fig.show()
```



## Reference

See https://plotly.com/python/reference/layout/geo/ (https://plotly.com/python/reference/layout/geo/) for more information and chart attribute options!

## What About Dash?

Dash (https://dash.plot.ly/) is an open-source framework for building analytical applications, with no Javascript required, and it is tightly integrated with the Plotly graphing library.

Learn about how to install Dash at https://dash.plot.ly/installation (https://dash.plot.ly/installation).

Everywhere in this page that you see `fig.show()`, you can display the same figure in a Dash application by passing it to the `figure` argument of the Graph component (https://dash.plot.ly/dash-core-components/graph) from the built-in `dash_core_components` package like this:

```
import plotly.graph_objects as go # or plotly.express as px
fig = go.Figure() # or any Plotly Express function e.g. px.bar(...)
# fig.add_trace( ... )
# fig.update_layout( ... )

from dash import Dash, dcc, html

app = Dash()
app.layout = html.Div([
    dcc.Graph(figure=fig)
])

app.run(debug=True, use_reloader=False)  # Turn off reloader if inside Jupyter
```