Tutorial of Cartopy (https://scitools.org.uk/cartopy/docs/latest/)

Created by Josué Martinez Moreno!

Goals:

- Create your first map with cartopy.
- Explore different projections and their attributes.
- Experiment creating a new map.
- Create your first scientific map.
- Understand the basics of cartopy.
- Create your own projection.

In this tutorial, you can just follow along and evaluate cells to see how to make nice, pretty plots! :)

Note: you must first download cartopy in order to run this tutorial.

Uncomment the cell below (i.e. delete the "#") to install cartopy. The "!" at the start of the line indicates that the line should be evaluated as a unix command (essentially turning that cell into a terminal or command prompt). You only need to do this once!

In [1]:
```
#!conda install -c conda-forge cartopy
```

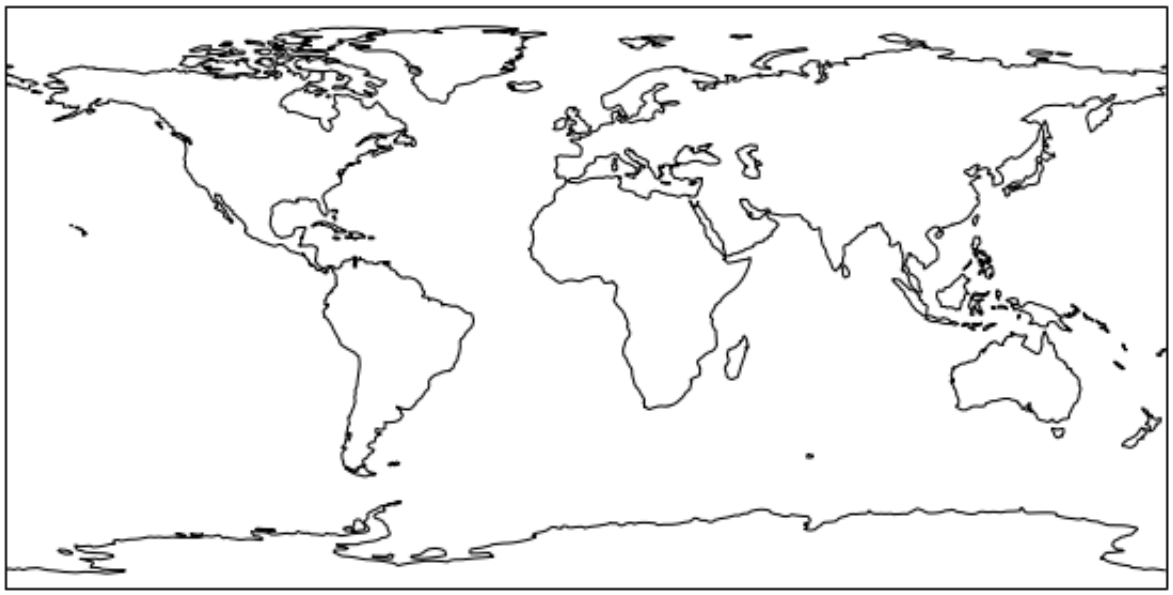
If you ran the line above successfully but are still having trouble running the remaining cells, try updating your matplotlib library (type "!conda update matplotlib" in a new code cell). It may make you update all of your packages. The problem is that cartopy requires an updated version of matplotlib. If you run into this problem and still don't know how to proceed, please ask for help in the Slack #python-help channel and we can help you out!

Let's create our first map!

In [2]:
```
# Import libraries:
# Import coordinate reference system
import cartopy.crs as ccrs
import matplotlib.pyplot as plt
```

In [3]:
```
# Creating a map can be as simple as:
fig = plt.figure(figsize=(10, 5))
# Create axis with a map projection
ax = plt.axes(projection=ccrs.PlateCarree())
# Draw coastlines
ax.coastlines()
```

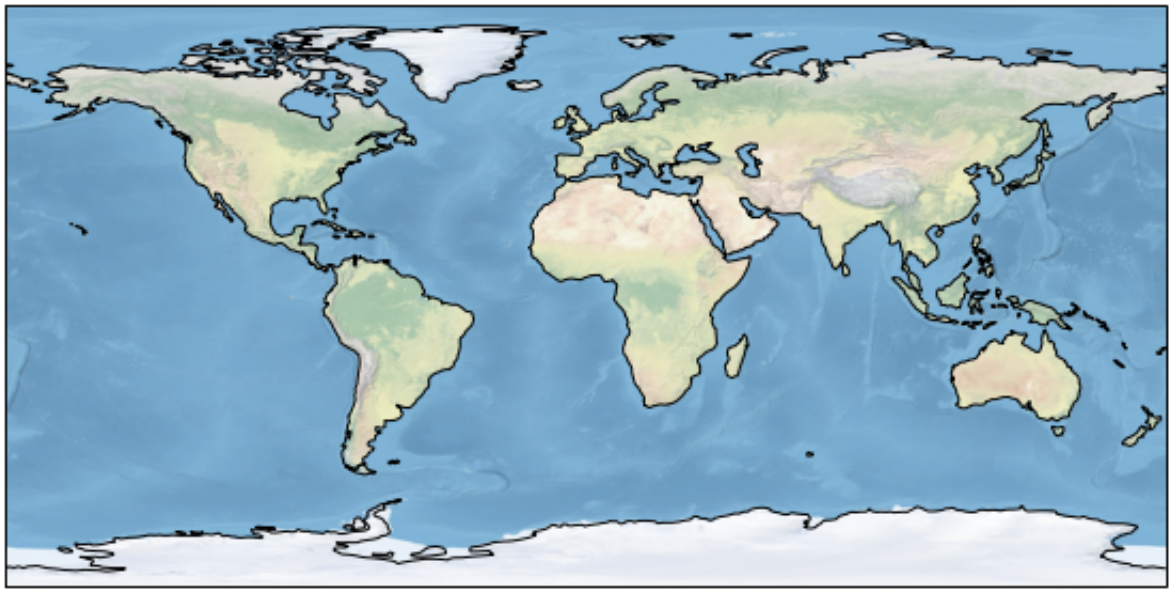
Out[3]: <cartopy.mpl.feature_artist.FeatureArtist at 0xa24887a90>



This map is a bit boring, so let's add some colors:

In [4]:
```
fig = plt.figure(figsize=(10, 5))
# Another way to create an axis with a map projection
ax = fig.add_subplot(1, 1, 1, projection=ccrs.PlateCarree())
# Default topography image
ax.stock_img()
# Draw coastlines
ax.coastlines()
```

Out[4]:
```
<cartopy.mpl.feature_artist.FeatureArtist at 0xa24a75990>
```



Cartopy has multiple projections (https://scitools.org.uk/cartopy/docs/latest/crs/projections.html):

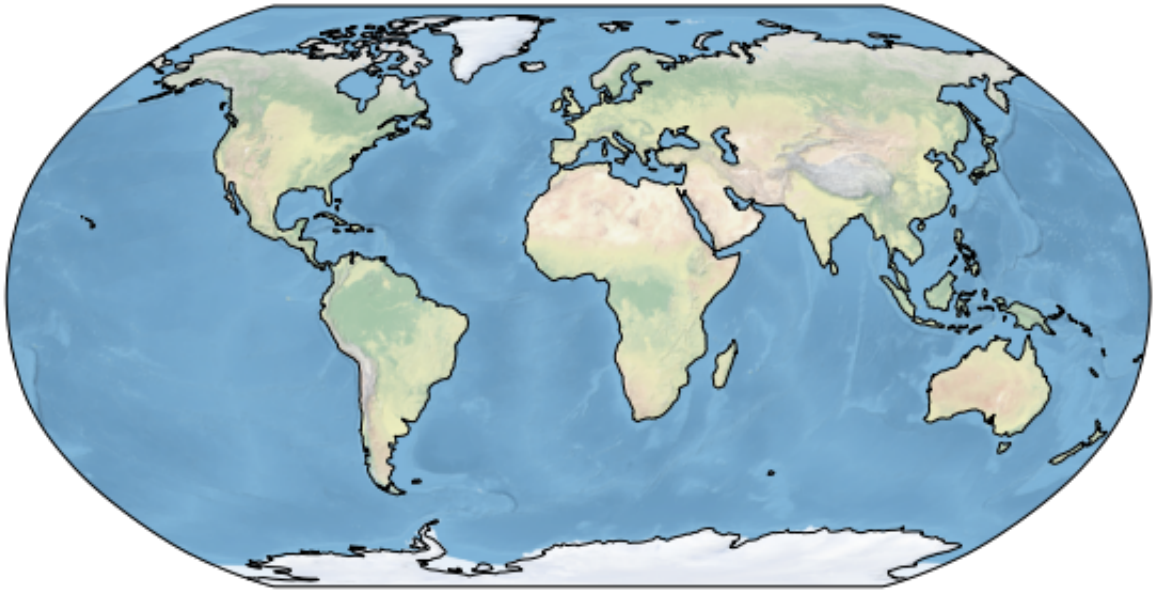
The following examples will include:

- PlateCarree
- Robinson
- Orthographic
- Sinusoidal

Let's start with a Robinson projection:

In [5]:
```
fig = plt.figure(figsize=(10, 5))
# Another way to create an axis with a map projection
ax = fig.add_subplot(1, 1, 1, projection=ccrs.Robinson())
# Default topography image
ax.stock_img()
# Draw coastlines
ax.coastlines()
```

Out[5]: <cartopy.mpl.feature_artist.FeatureArtist at 0xa25b73d90>



Time to play arround with other projections!

Clue: Replace the argument projection with any of the projections availabe at:

https://scitools.org.uk/cartopy/docs/latest/crs/projections.html
(https://scitools.org.uk/cartopy/docs/latest/crs/projections.html)

In [6]:
```
# Experiment 1
```

In [7]:
```
# Experiment 2
```

Now we are ready to plot some data on top of the maps:

In [8]:
```
#Import library to read netCDFs
import xarray as xr
```

Download any of the following data:

- Mean SST: ftp://ftp.cdc.noaa.gov/Datasets/noaa.oisst.v2/sst.mnmean.nc
- SST 1990-present: ftp://ftp.cdc.noaa.gov/Datasets/noaa.oisst.v2/sst.wkmean.1990-present.nc
- Ice concentration 1990-present: ftp://ftp.cdc.noaa.gov/Datasets/noaa.oisst.v2/icec.wkmean.1990-present.nc

One way of doing it is by using `!wget ftp:path` in the next cell. Note that this will download a 57MB dataset, but you only have to run it once! So uncomment the first time through, but then rememebr to add the "#" back to comment it out for future runs!

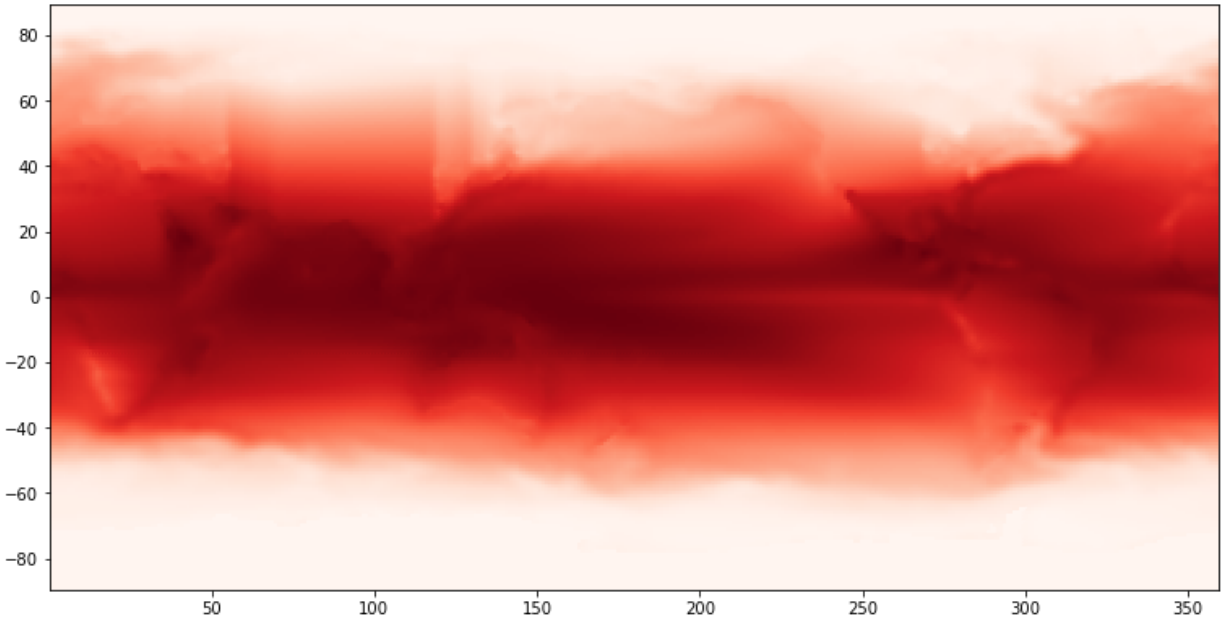
In [9]:
```
#!wget ftp://ftp.cdc.noaa.gov/Datasets/noaa.oisst.v2/sst.mnmean.nc
```

In [10]:
```
# Lazy opening of dataset.
# Lazy means means that only the metadata of the netcdf is read.
monthly_mean_sst = xr.open_dataset('sst.mnmean.nc')
```

In [11]:
```
# Compute mean over record.
mean_sst = monthly_mean_sst.mean('time').sst
```

In [12]:
```
# Create figure
fig = plt.figure(figsize=(10, 5))
# Add figure axes
ax = fig.add_axes([0,0,1,1])
# Plot data
plt.pcolormesh(mean_sst.lon,mean_sst.lat,mean_sst,cmap='Reds')
```

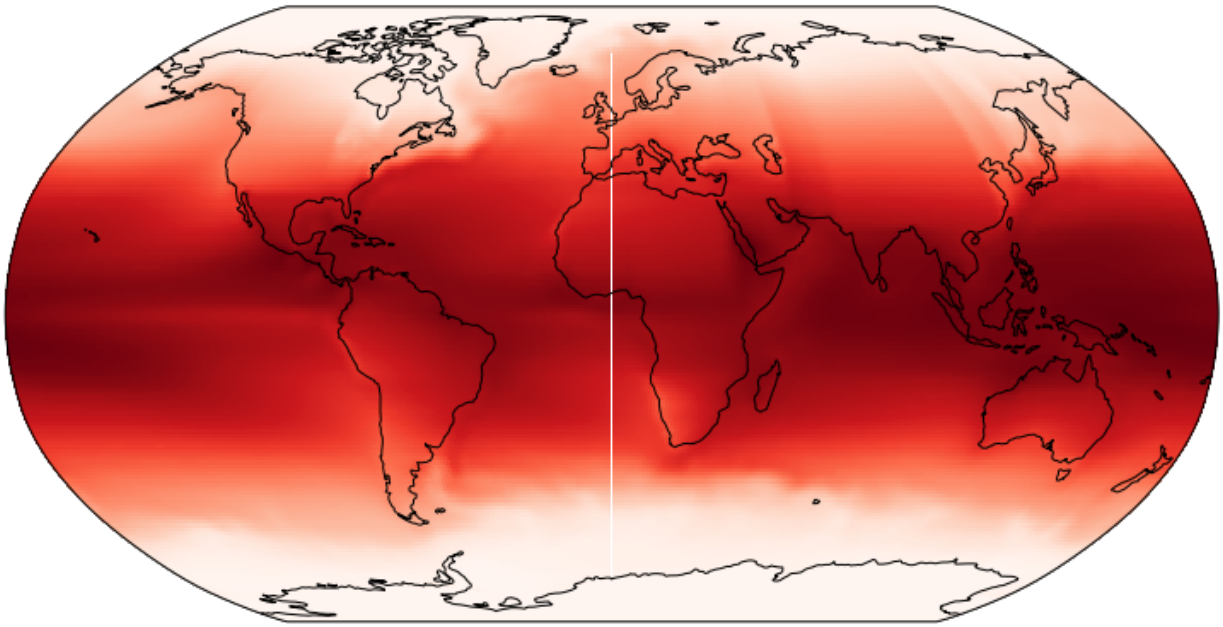
Out[12]: <matplotlib.collections.QuadMesh at 0xa2c60b250>



This is horrible, let's use cartopy!

In [13]:
```
# Create figure
fig = plt.figure(figsize=(10, 5))
# Add figure axes
ax = fig.add_axes([0,0,1,1], projection=ccrs.Robinson())
# Plot data
p = plt.pcolormesh(mean_sst.lon,mean_sst.lat,mean_sst,transform=ccrs.P
lateCarree(),cmap='Reds')
# Draw coastlines
ax.coastlines()
```

Out[13]: <cartopy.mpl.feature_artist.FeatureArtist at 0xa2d2095d0>

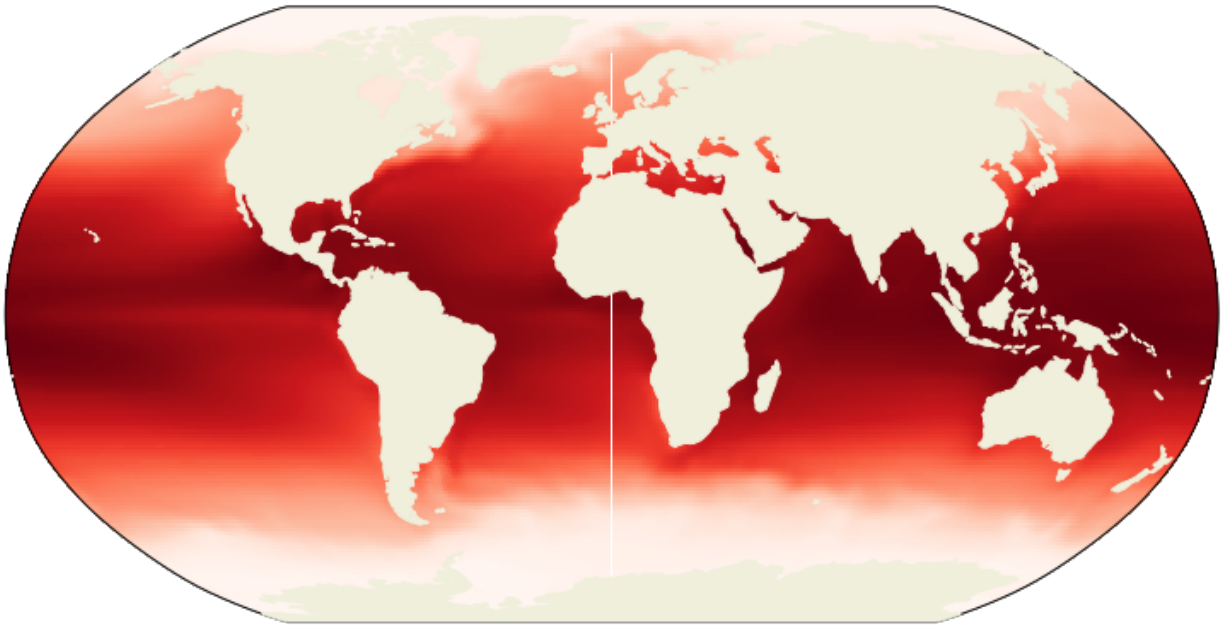


We don't expect to have Sea surface temperature over the continents, so let's mask the land.

In [14]:
```
# Import map features such as land, continents, contry boundaries and
other.
import cartopy.feature as cfeature
```

In [15]:
```
# Create figure
fig = plt.figure(figsize=(10, 5))
# Add figure axes
ax = fig.add_axes([0,0,1,1], projection=ccrs.Robinson())
# Plot data
p = plt.pcolormesh(mean_sst.lon,mean_sst.lat,mean_sst,transform=ccrs.P
lateCarree(),cmap='Reds')
# Draw continents
ax.add_feature(cfeature.LAND,zorder=10)
```

Out[15]: <cartopy.mpl.feature_artist.FeatureArtist at 0xa2cc32510>



Why can we see a line in the middle of the plot?

In [16]:
```
## In order to solve it, uncomment the following line,
mean_sst_seam = xr.concat([mean_sst, mean_sst.isel({mean_sst.dims[-1]:
0})], dim=mean_sst.dims[-1])
## This line join the seam of the data by duplicating the value in $35
9^\circ$ to $360^\circ$, so the field is defined all around the globe.
```

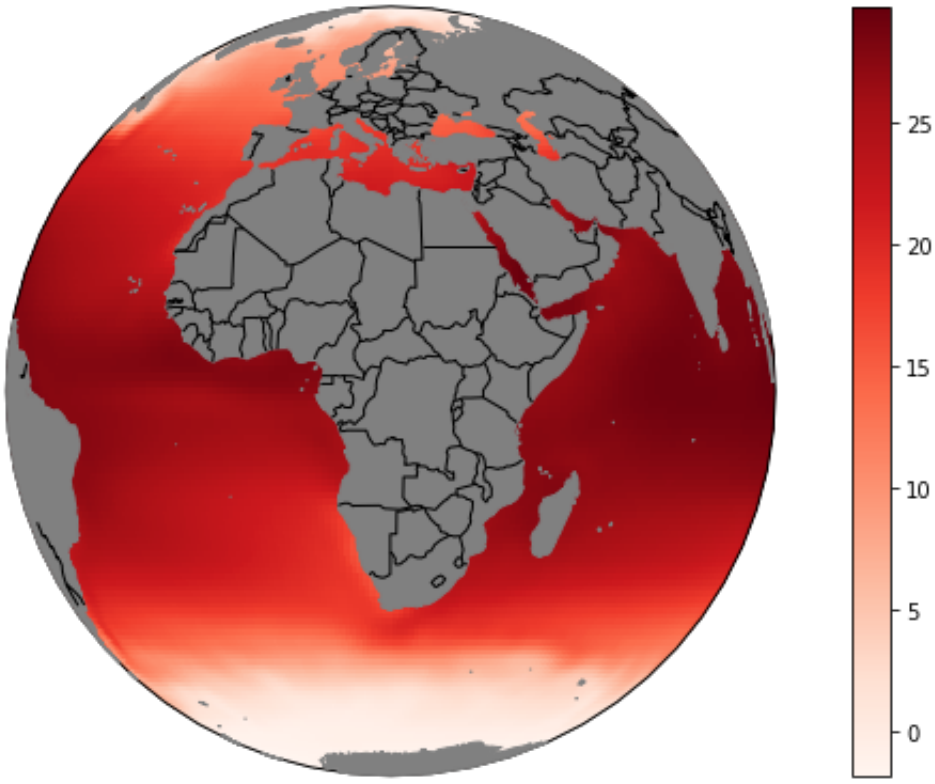
With that continental feature, we cant differentiate between low temperatures and the continentes, so let's customize a bit more our map.

First, we will create some custom features:

In [17]:
```
# Create country boundaries using Natural Earth Features: https://www.
naturalearthdata.com
countries = cfeature.NaturalEarthFeature('cultural', 'admin_0_boundary
_lines_land',
                                         scale='110m',
                                         edgecolor='black',
                                         facecolor='none')

# Create continents mask with 50 meter resolution data
land_50m = cfeature.NaturalEarthFeature('physical', 'land',
                                         scale='50m',
                                         edgecolor='face',
                                         facecolor='gray')

# Create continents mask with 10 meter resolution data
land_10m = cfeature.NaturalEarthFeature('physical', 'land',
                                         scale='10m',
                                         edgecolor='face',
                                         facecolor='gray')

# Create ocean mask
oceans_50m = cfeature.NaturalEarthFeature('physical', 'ocean',
                                         scale='50m',
                                         edgecolor='face',
                                         facecolor='stealblue')
```

In [18]:
```
# Create figure
fig = plt.figure(figsize=(10, 5))
# Add figure axes
ax = fig.add_axes([0,0,1,1], projection=ccrs.Orthographic(20, -0))
# Plot data
p = plt.pcolormesh(mean_sst_seam.lon,mean_sst_seam.lat,mean_sst_seam,t
ransform=ccrs.PlateCarree(),cmap='Reds')
# Add land feature
ax.add_feature(land_50m, zorder=10)
# Add countries feature
ax.add_feature(countries,zorder=11)
# Add colorbar
plt.colorbar()
```

Out[18]: <matplotlib.colorbar.Colorbar at 0xa2ccb3710>

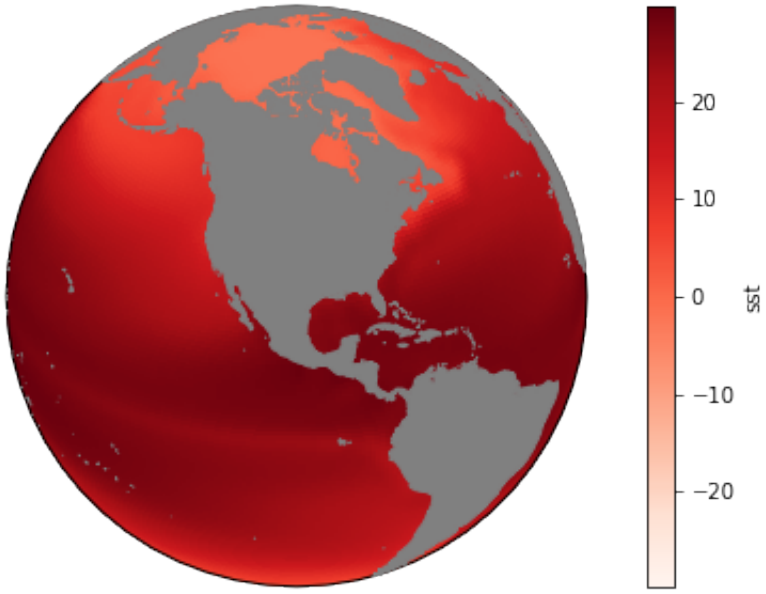


`xarray` is a package that natively allows users to plot using cartopy

Note that by default xarray deals with the same issue.

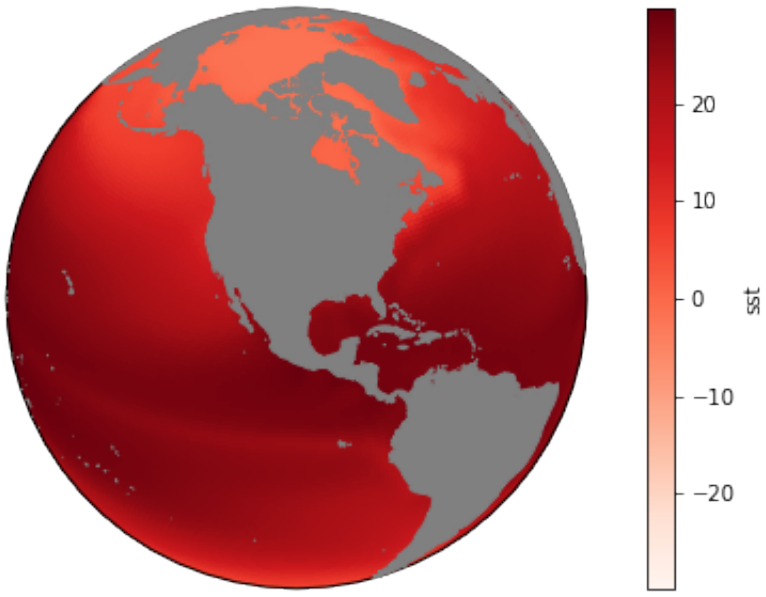
In [19]:
```
# Create figure
fig = plt.figure(figsize=(10, 5))
# Plot data
p = mean_sst.plot(subplot_kws=dict(projection=ccrs.Orthographic(-100,
30), facecolor="gray"), transform=ccrs.PlateCarree(),cmap='Reds')
# Add land feature
p.axes.add_feature(land_50m, zorder=10)
```

Out[19]: <cartopy.mpl.feature_artist.FeatureArtist at 0xa2baee390>



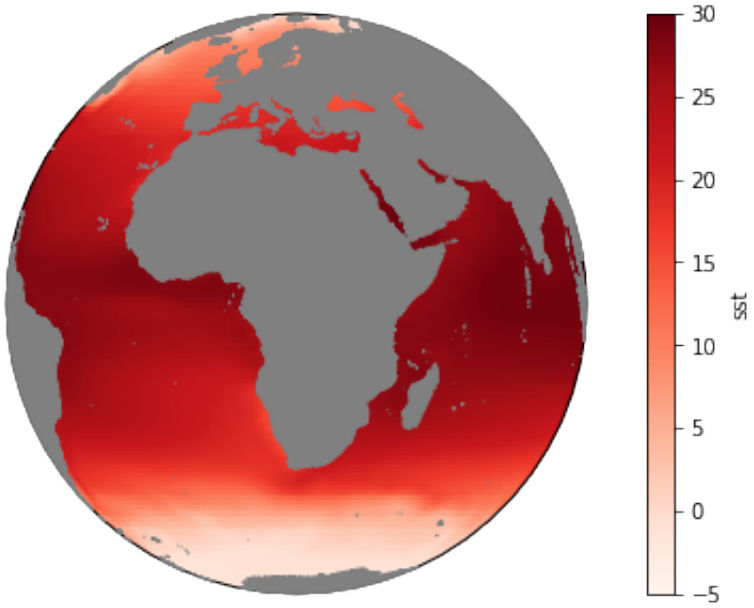
In [20]:
```
# Create figure
fig = plt.figure(figsize=(10, 5))
# Plot data
proj = ccrs.PlateCarree() #ccrs.Orthographic(-100, 30)
p = mean_sst.plot(subplot_kws=dict(projection=ccrs.Orthographic(-100,
30), facecolor="gray"), transform=ccrs.PlateCarree(),cmap='Reds')
#p = mean_sst.plot(subplot_kws={'projection':proj, 'facecolor':"gray"}
)
# Add land feature
p.axes.add_feature(land_50m, zorder=10)
```

Out[20]: <cartopy.mpl.feature_artist.FeatureArtist at 0xa2c592e10>



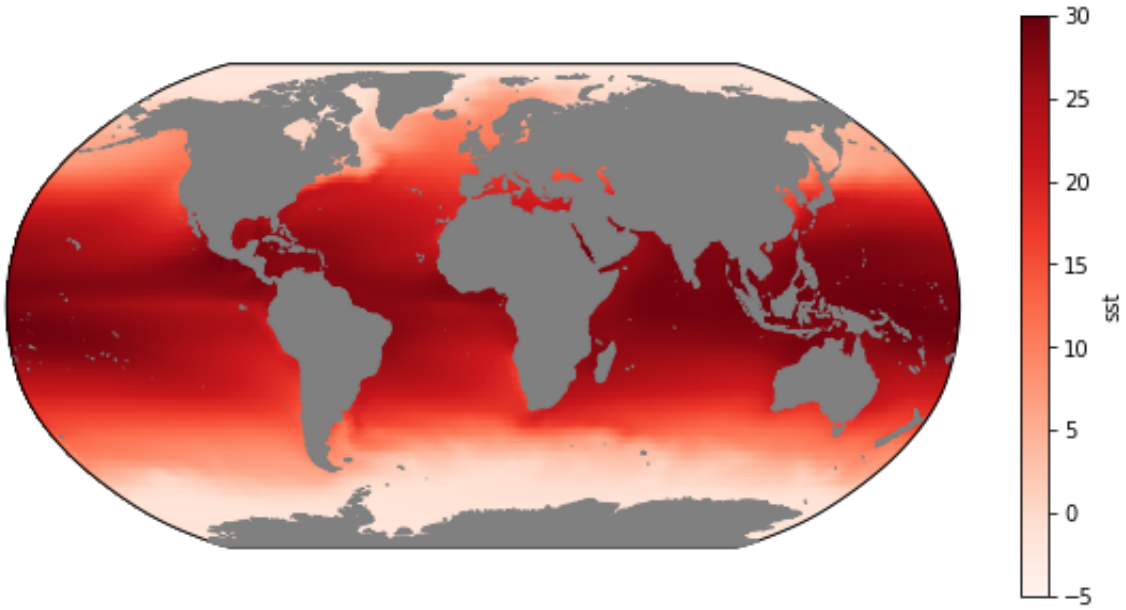
In [21]:
```
# Create figure
fig = plt.figure(figsize=(10, 5))
# Plot data
p = mean_sst.plot(subplot_kws=dict(projection=ccrs.Orthographic(20, -0
), facecolor="gray"), transform=ccrs.PlateCarree(),cmap='Reds',vmin=-5
,vmax=30)
# Add land feature
p.axes.add_feature(land_10m, zorder=10)
```

Out[21]: <cartopy.mpl.feature_artist.FeatureArtist at 0xa2c503310>



In [22]:
```
# Create figure
fig = plt.figure(figsize=(10, 5))
# Plot data
p = mean_sst.plot(subplot_kws=dict(projection=ccrs.Robinson(central_lo
ngitude = 0), facecolor="gray"), transform=ccrs.PlateCarree(),cmap='Re
ds',vmin=-5,vmax=30)
# Add land feature
p.axes.add_feature(land_50m, zorder=10)
```

Out[22]: <cartopy.mpl.feature_artist.FeatureArtist at 0xa2aea3350>

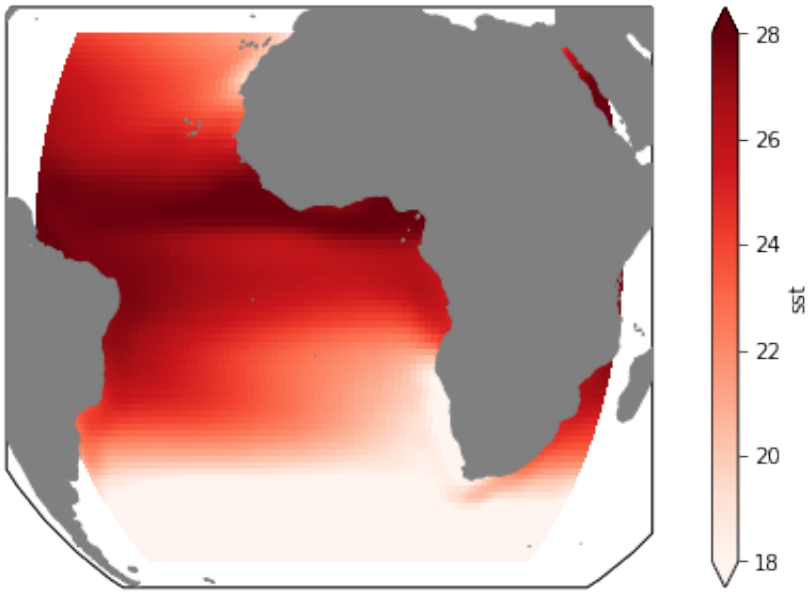


Now let's plot only a section of the data:

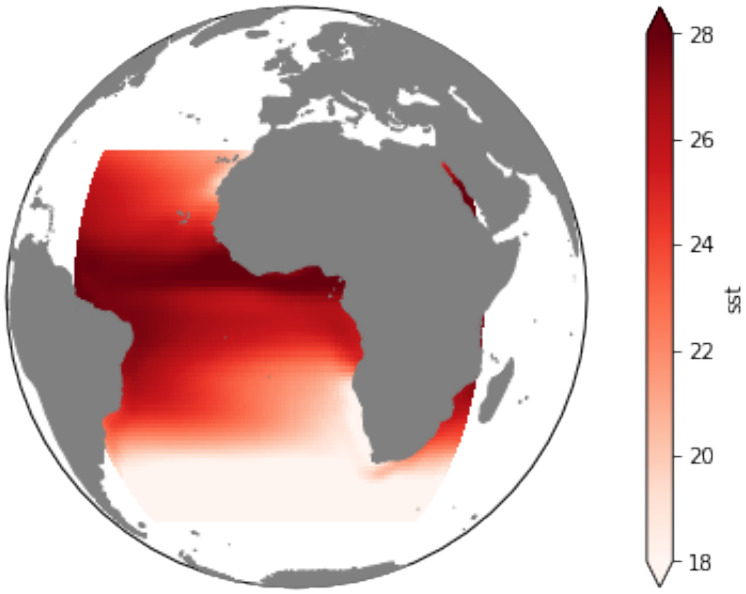
In [23]:
```
# Rotate coordinates from 0-360 degrees to -180 to 180.
mean_sst_rotate_coords = mean_sst.assign_coords(lon=(((mean_sst.lon +
180) % 360) - 180)).sortby('lon')
# Slice data
SST_atlantic_ocean = mean_sst_rotate_coords.sel({'lat':slice(30,-50),'
lon':slice(-50,40)})
```

In [24]:
```
# Create figure
fig = plt.figure(figsize=(10, 5))
# Plot data
p = SST_atlantic_ocean.plot(subplot_kws=dict(projection=ccrs.Orthograp
hic(central_longitude = 0), facecolor="gray"), transform=ccrs.PlateCar
ree(),cmap='Reds',vmin=18,vmax=28)
# Add land feature
p.axes.add_feature(land_50m, zorder=10)
```

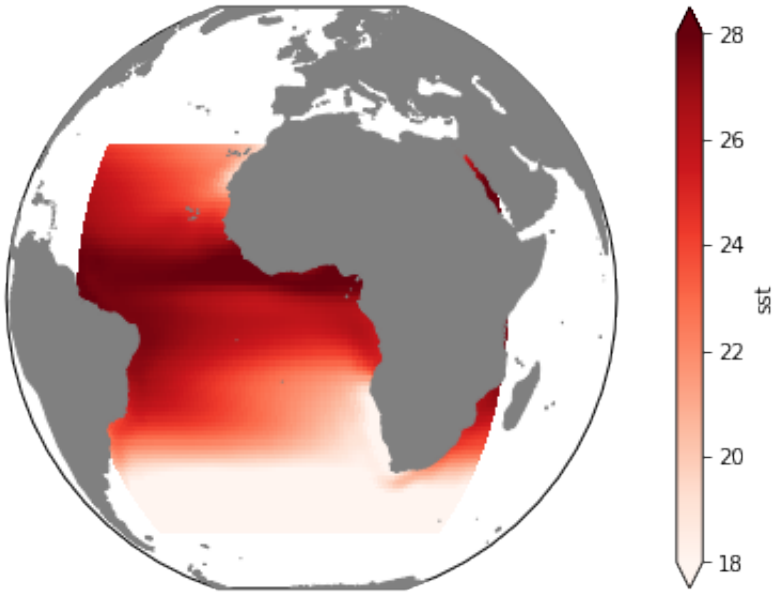
Out[24]: <cartopy.mpl.feature_artist.FeatureArtist at 0xa3008e3d0>



In [25]:
```
# Create figure
fig = plt.figure(figsize=(10, 5))
# Plot data
p = SST_atlantic_ocean.plot(subplot_kws=dict(projection=ccrs.Orthograp
hic(central_longitude = 0), facecolor="gray"), transform=ccrs.PlateCar
ree(),cmap='Reds',vmin=18,vmax=28)
# Add land feature
p.axes.add_feature(land_50m, zorder=10)
# Set extent of map to global
p.axes.set_global()
```



In [26]:
```
# Create figure
fig = plt.figure(figsize=(10, 5))
# Plot data
p = SST_atlantic_ocean.plot(subplot_kws=dict(projection=ccrs.Orthograp
hic(central_longitude = 0), facecolor="gray"), transform=ccrs.PlateCar
ree(),cmap='Reds',vmin=18,vmax=28)
# Add land feature
p.axes.add_feature(land_50m, zorder=10)
# Set extent of map allows to select the coordinates of the map (Simil
ar result as before).
p.axes.set_extent((-89,89,-89,89))
```



Now let's create our own projection:

We will explore Myriahedral projections.

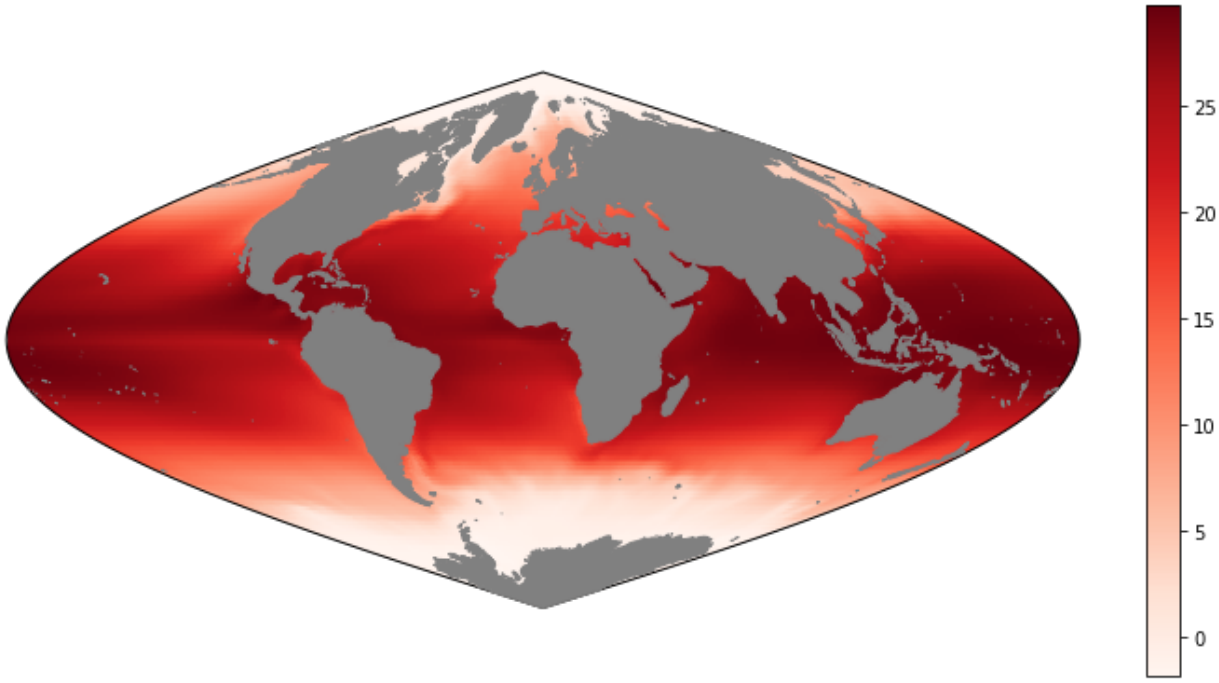
Have a look here to learn more about them: https://philogb.github.io/page/myriahedral/ (https://philogb.github.io/page/myriahedral/).

Here we will reproduce the Cylindrical myriahedral projection, however some of the other projections will be significantly more challenging.

The Cylindrical Myriahedral projection is a natural development from a Sinusoidal projection, therefore, we will start ploting the data in a sinusoidal projection.

In [27]:
```
# Create figure
fig = plt.figure(figsize=(10, 5))
# Add axes
ax = fig.add_axes([0,0,1,1], projection=ccrs.Sinusoidal(0))
# Plot data
p = plt.pcolormesh(mean_sst_rotate_coords.lon,mean_sst_rotate_coords.l
at,mean_sst_rotate_coords,transform=ccrs.PlateCarree(),cmap='Reds')
# Set extent of map.
ax.set_global()
# Add land feature
ax.add_feature(land_50m, zorder=10)
# Add colorbar
plt.colorbar()
```

Out[27]: <matplotlib.colorbar.Colorbar at 0xa3077ca10>



Now we set the extend of the plot to only obtain the width of a gore, in this case -15° to 15°

In [28]:
```
# Create figure
fig = plt.figure(figsize=(10, 5))
# Add axes
ax = fig.add_axes([0,0,1,1], projection=ccrs.Sinusoidal(0))
# Plot data
p = plt.pcolormesh(mean_sst_rotate_coords.lon,mean_sst_rotate_coords.l
at,mean_sst_rotate_coords,transform=ccrs.PlateCarree(),cmap='Reds')
# Set extent of map to slice data.
ax.set_extent((-15,15,-90,90))
# Add land feature
ax.add_feature(land_50m, zorder=10)
```

Out[28]: <cartopy.mpl.feature_artist.FeatureArtist at 0xa2d2c0510>

In order to mask the data we have to provide a boundary to the map:

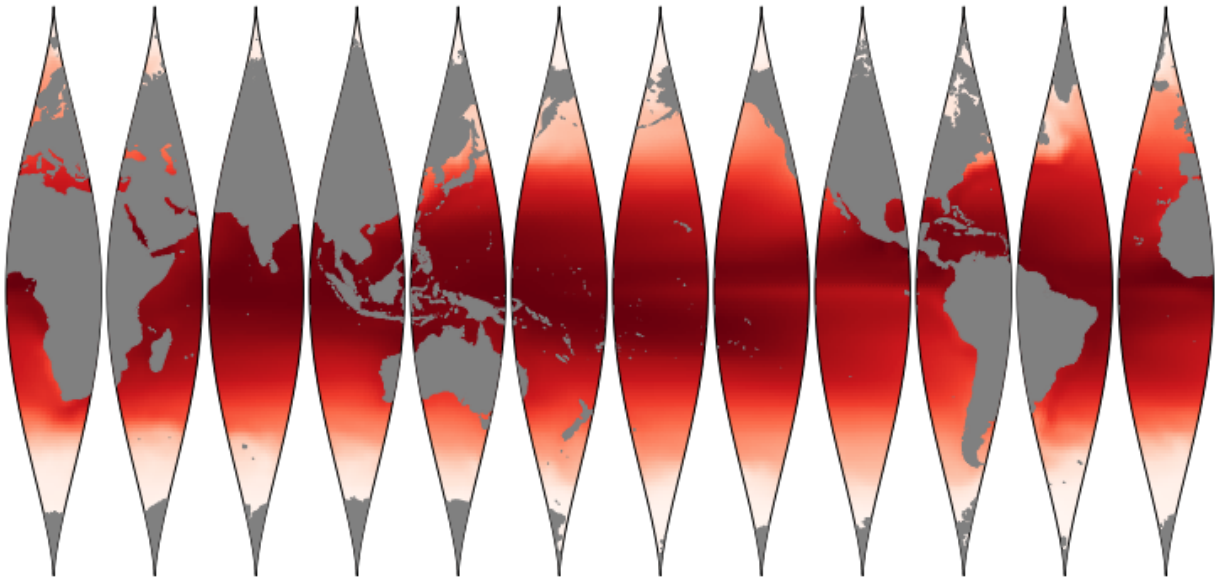
In [29]:
```
# Import additional libraries to construct Myriahedral projection
import numpy as np
import matplotlib.path as mpath
import matplotlib.patches as mpatches

# Create figure and axes
fig = plt.figure(figsize=(10, 5))
ax = fig.add_axes([0,0,1,1], projection=ccrs.Sinusoidal(0))

# Geometry of ellipse to bound map.
theta = np.linspace(-np.pi, np.pi, 100)
psi = np.linspace(-np.pi, np.pi, 100)
r, R = 15, 90
verts = np.vstack([ r * np.cos(theta), -R *np.sin(psi) ]).T

bound = mpath.Path(verts)
# Set bounds to map.
ax.set_boundary(bound,ccrs.PlateCarree(0))

# Slice data to region of interest
data = mean_sst_rotate_coords.sel(lon=slice(-15,15))
# Plot data
p = plt.pcolormesh(data.lon,data.lat,data,transform=ccrs.PlateCarree()
,cmap='Reds')

# Add land
ax.add_feature(land_50m, zorder=10)
```

Out[29]: <cartopy.mpl.feature_artist.FeatureArtist at 0xa3147fe50>

Now let's create all the gores.

In [30]:
```
# Define a function to generate the gore boundary.
def gore_boundary(gore_width):
    # Sphere coordinates
    theta = np.linspace(-np.pi, np.pi, 100)
    psi = np.linspace(-np.pi, np.pi, 100)
    r, R = gore_width/2, 90
    # Ellipse verts
    verts = np.vstack([ r * np.cos(theta), - R *np.sin(psi) ]).T
    # Bounding path
    bound = mpath.Path(verts)
    return bound
```

In [31]:
```
import numpy as np
import matplotlib.path as mpath
import matplotlib.patches as mpatches

fig = plt.figure(figsize=(10, 5))

# Width of gores
gore_width = 30
gores = 360//gore_width

for ii in range(gores):
    # Project data into Sinusoidal projection.
    ax = fig.add_axes([ii*(1/gores),0,1/gores,1], projection=ccrs.Sinu
soidal(ii*(gore_width)+gore_width/2))
    # Set boundary according to an ellipse of radius r and R
    ax.set_boundary(gore_boundary(gore_width),ccrs.PlateCarree(ii*(gor
e_width)+gore_width/2))
    # Slice data
    data = mean_sst.sel(lon=slice(ii*(gore_width),ii*(gore_width)+gore
_width))
    # Pcolormesh data over map
    p = plt.pcolormesh(data.lon,data.lat,data,transform=ccrs.PlateCarr
ee(),cmap='Reds')
    # Add land to each subplot
    ax.add_feature(land_50m, zorder=10)
```



Now you can play with different projections and the cyldrical Myriahedral projection.