# BIOM4031 Tutorial 4: Multi-dimensional rasters

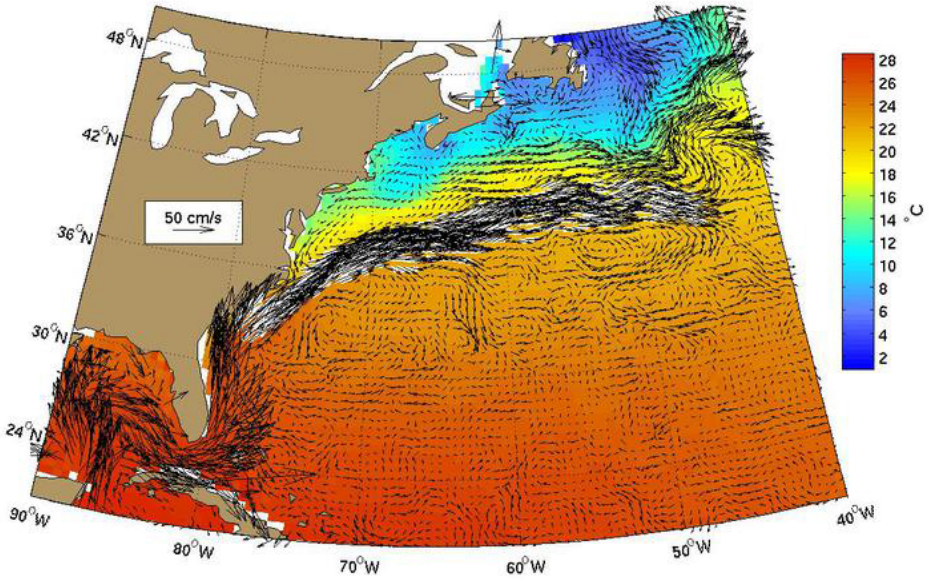



Figure 1: The Gulf Stream in the North Atlantic Ocean

## Introduction

In Tutorial 3 you learned how to perform a range of common operations using single-layer spatial raster datasets in R. In this tutorial we are going to build on those skills and introduce some more complex operations using multi-dimensional raster datasets (often known as 'raster cubes') consisting of multiple layers or 'bands'.

To illustrate some common applications of multi-dimensional rasters, we are going to use open source satellite data to map the seasonal position of the Gulf Stream: one of the major current systems in the Atlantic Ocean. The Gulf Stream plays a pivotal role in regulating the regional climate system in the North Atlantic, transporting warm water from the Gulf of Mexico up the eastern seaboard of the United States and towards Western Europe. It is also an important wildlife habitat, acting as a migration corridor and generating productive foraging areas for many large, pelagic species. Evidence suggests that the Gulf Stream is weakening and approaching a tipping point as a result of anthropogenic climate change Caesar et al. 2018, with potentially far-reaching societal and ecological consequences.

Our goal in today's tutorial is to recreate a surface temperature and current map like that shown in Figure 1 which we could use as a starting point for applied ecological research (e.g. mapping dispersal/migrations, modelling species distributions, and studying fisheries interactions). It is important to note that the techniques we will be applying here are relevant to any multidimensional raster problems (e.g. calculating average rainfall, mapping winds etc.)

### Learning outcomes

By the end of this tutorial you should be able to:

1. Understand the structure of multi-dimensional rasters and the way they are represented in R
2. Use local statistics to summarise and calculate new variables from multi-dimensional rasters
3. Export multiband rasters

### Data

In the Tutorial 4 data folder you will find two files containing surface oceanographic data for the northwest Atlantic downloaded from the Copernicus Marine Environment Monitoring Service.

**sst.nc** Daily sea surface temperature data for the northwest Atlantic in 2020

**currents.nc** Daily current velocity data for the northwest Atlantic in 2020

Both of these datasets were obtained from the Global Physics Reanalysis and Forecast product but have been split to reduce file size and allow us to introduce multi-dimensional rasters in stages. The data are in NetCDF (.nc) format which is a common format used for storing high dimensional raster datasets but is easily imported with R.

### Packages

The packages used in this tutorial are the same as in Tutorial 3, with the addition of the `lubridate` package - a tidyverse package that we will use to simplify working with dates and times.

```
library(tidyverse)
library(terra)
library(tidyterra)
library(lubridate)
```

## 1. Introducing multilayer rasters

To help us familiarize ourselves with the structure of multidimensional rasters let's start off by importing our sea surface temperature data from the northwest Atlantic using `rast`.

```
sst = rast('data/sst.nc')
sst
```

If we print a summary you can see that `sst` is still a `SpatRaster` object with much of the same information as we saw for a single layer rasters in Tutorial 3. The only difference is that in this case we have a stack of 366 layers, one for each day of the year. Raster stacks function quite like a `list` object in R and we can extract a single layer using the double square bracket list notation:

```
# This extracts the second layer from the stack
sst[[2]]

# Or extract layers 2-5
sst[[2:5]]
```

You can think of this stack as a three dimensional raster with latitude and longitude dimensions and a third time dimension describing how the values in each cell change over time. Indeed, you will notice that our `sst` stack has a 'time' attribute associated with it which we can extract using the function `time`:

```
time(sst)
```

A lot of operations work in the same way on multidimensional rasters as they do on single layers. For example, we could use the `global` function we introduced in Tutorial 2 to find the average sea surface temperature in each layer (i.e. day of the year):

```
global(sst, fun='mean', na.rm=T)
```

However, in our case, we are more interested in how the average SST varies spatially (i.e. for each cell) across our study area, which involves the use of 'local statistics'.

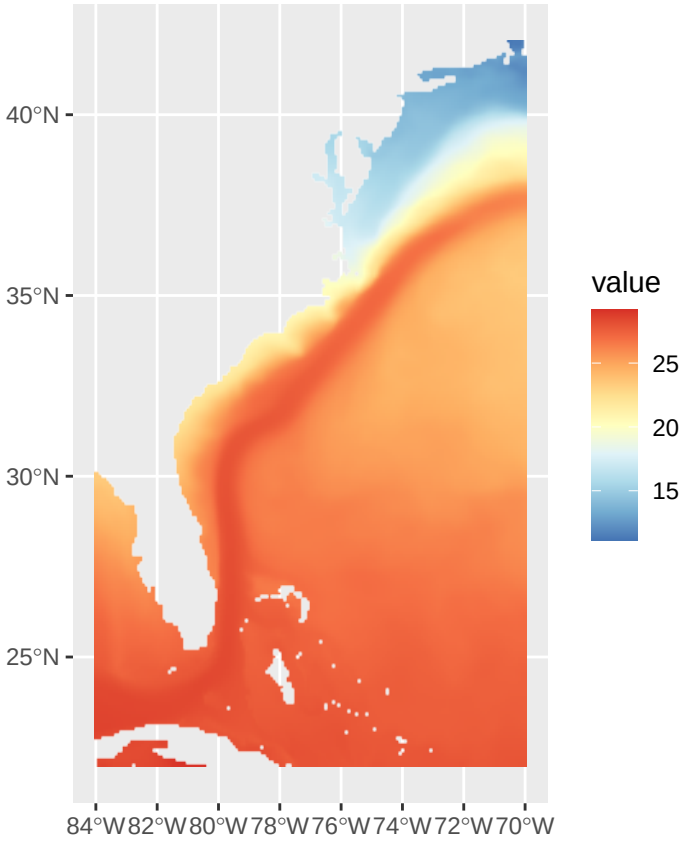
## 2. Local statistics on multilayer rasters

Local statistics are cell-by-cell operations that calculate a new value for each cell and return a new raster layer. For example, we might want to produce a single raster layer containing the average annual temperature for each cell across our study area. In `terra` we do that using a function called `app` which is short for 'apply' and means **"apply a function to each cell in the stack"**. In this case the function, `fun`, is the mean.

```
mean.sst = app(sst, fun='mean', na.rm=T)
```

The mean is calculated for cell 1 in all layers, then cell 2 in all layers and so forth to produce a new mean temperature raster. Then we plot our mean SST raster using ggplot. This time we'll us another one of our tidyterra palletes added using `scale_fill_whitebox` ('whitebox' is the name of another open source GIS that these palettes are taken from):

```
sst.plot =
ggplot() +
  geom_spatraster(data=mean.sst) +
  scale_fill_whitebox_c(palette='bl_yl_rd')

sst.plot
```




**Can everyone see the Gulf Stream!?**

But what if we don't want to average the entire stack? For example, we might want to calculate the average temperature by month to map the seasonal position of the Gulf Stream. In `terra` we do that using a variant of `app` called `tapp` (short for `tapply`, which is a base R function used to apply a function within groups). In `tapp` the groups are set using an `index`. Any layers that share the same index will be grouped and a new raster layer returned for each group.

To calculate the mean monthly sea surface temperatures for this area. First we set up our indices. As above, we will use the `time` function from `terra` to extract the time dimension from our stack, and the `month` function from package `lubridate` to extract the month from each of these times:

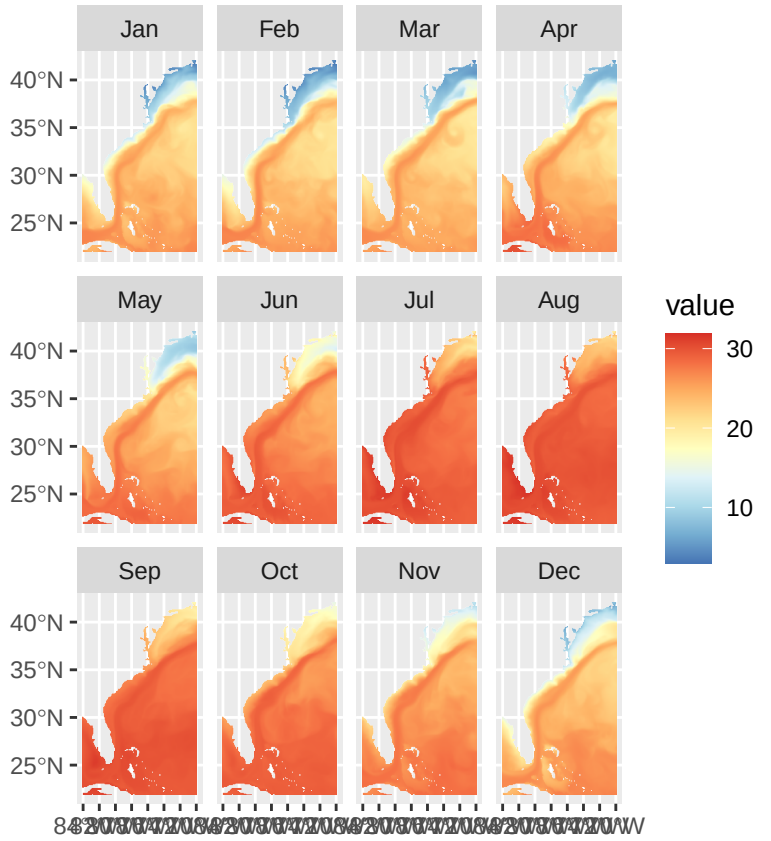
```
months = month(time(sst), label = T)
```

We set `label=T` to get the month name rather than its number as these indices will become the layer names in our new, summarized raster. Now we use the months as our indices in `tapp`. You'll see we end up with a new stack with 12 named layers, one for each month:

```
monthly.sst = tapp(sst,index = months, fun = 'mean', na.rm=T)
monthly.sst
plot(monthly.sst)
```

To plot multilayer rasters using `ggplot` we add a `facet_wrap` just as we did in Tutorial 1 and tell the function that we want to facet by `lyr` (short for layer):

```
ggplot() +
  geom_spatraster(data = monthly.sst) +
  scale_fill_whitebox_c(palette='bl_yl_rd') +
  facet_wrap(~lyr)
```




It seems like the Gulf Stream is more defined outside of summer months, which explains its importance for bringing milder winters to Eastern Seaboard of USA and parts of Western Europe, including Cornwall!

## Raster algebra (it's easier than it sounds!)

So far we've worked with a single raster dataset containing multiple layers (temperature). However, in other cases we want to combine data from different raster datasets to calculate a new variable. To demonstrate, we are going to calculate the average current direction and velocity in our northwest Atlantic study area to help us map the flow of the Gulf Stream. First, let's load in the `currents` NetCDF file from the Tutorial 4 data folder.

```
rast('data/currents.nc')
```

If we inspect the summary of this `SpatRaster` dataset you can see that it is very similar to the `sst` data we imported earlier, but it contains two variable names, or `varnames` (Northward Velocity and Eastward Velocity) from two `sources`, each with 365 layers. Essentially, this dataset consists of two different raster stacks, each with the same dimensions, containing different variables. The easiest way to access these different datasets is to load our data in as a `SpatRasterDataset` using the function `sds`

```
currents = sds('data/currents.nc')
currents
```

In the summary of this `SpatRasterDataset` you can see we now have two subdatasets called 'vo' (Northward Velocity) and 'uo' (Eastward Velocity) each with 365 layers containing data for each day of the year. You can subset each dataset using normal square bracket notation to obtain single variable raster stacks like our `sst` data:

```
v = currents['vo']
u = currents['uo']
v
u
```

Now for the raster algebra. For practical reasons, data on current flows (e.g. ocean currents and winds) are generally stored as northward velocities (often abbreviated to 'V') and eastward velocities (often abbreviated to 'U'). We need to calculate the absolute current velocity and flow direction from these using the following equations:

$$velocity = \sqrt{(u^2 + v^2)}$$

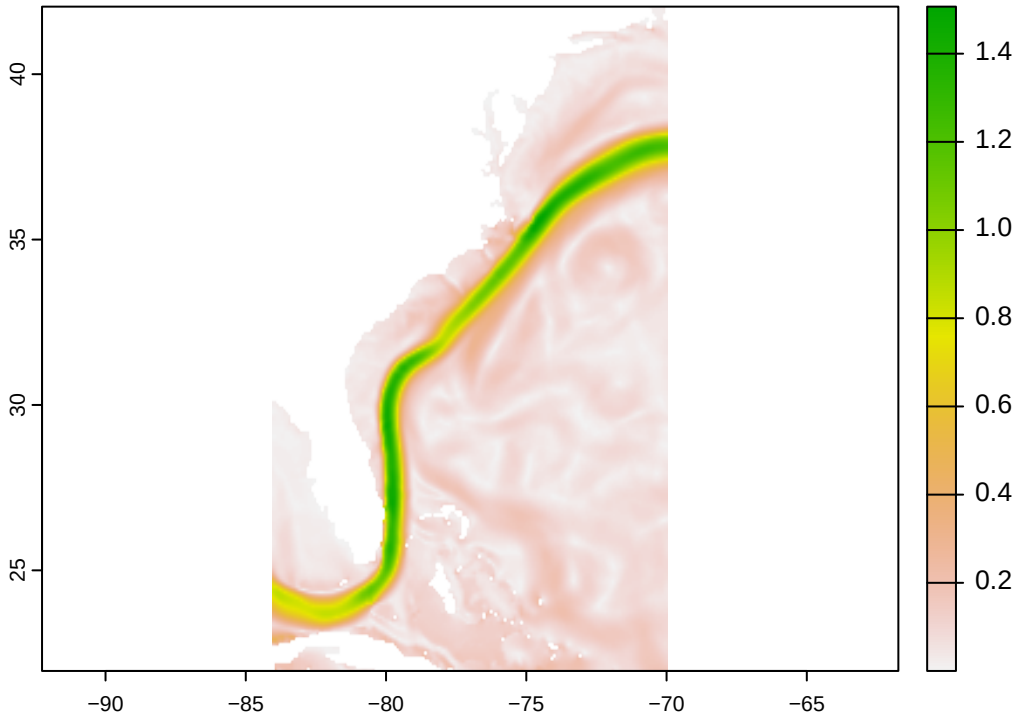
$$direction = atan2(v, u)$$

Let's start off by calculating the average northward and eastward velocities across the whole year using local statistics we introduced above:

```
mean.u = app(u,fun='mean',na.rm=T)
mean.v = app(v,fun='mean',na.rm=T)

plot(mean.u)
plot(mean.v)
```

We end up with two single layer rasters. Calculating the average current strength is now as simple as plugging our `mean.u` and `mean.v` rasters into our first equation:

```
# Velocity in m/s
mean.velocity = sqrt(mean.u^2 + mean.v^2)
plot(mean.velocity)
```




Anyone spot the Gulf Stream this time?! This is basically another kind of local statistics. The calculation is applied to cell 1 in the U and V rasters, then cell 2, and so on. The result is a new raster layer with the calculated value for each cell (**Note: Rasters used in the calculation must have the same dimensions and projections for this to work!**). We do the same to calculate current direction (in radians):

```
mean.dir = atan2(mean.v,mean.u)
plot(mean.dir)
```

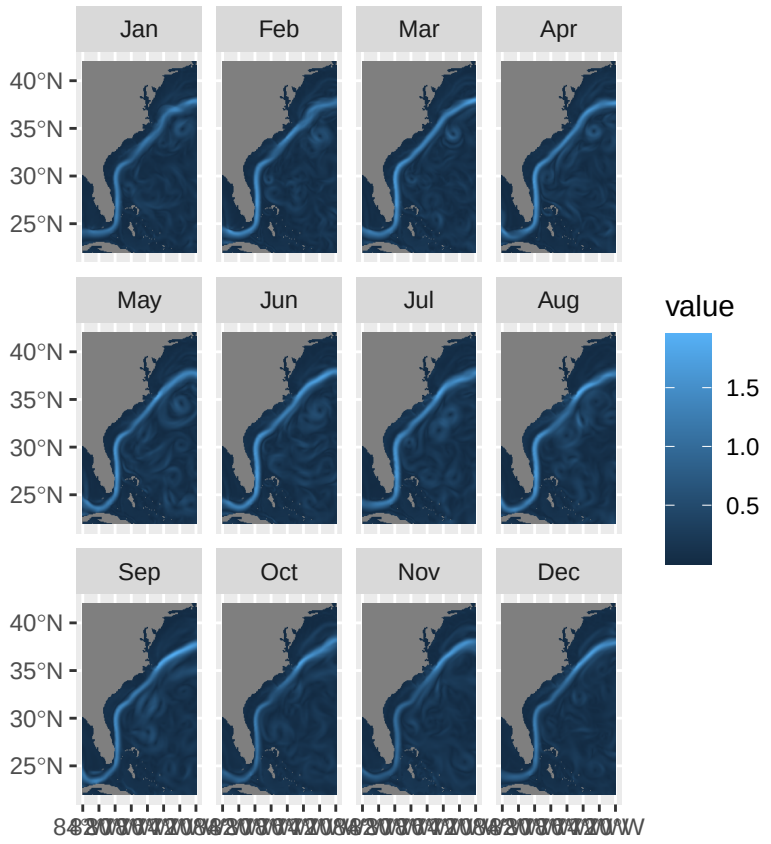
Note that the current direction plot isn't very easy to interpret, for the same reason as aspect was hard to plot on a continuous scale in Tutorial 3: it is a circular variable. More on this below!

We can do raster algebra on stacks too. For example, let's calculate the monthly mean current strength and direction. We start by calculating the monthly mean northward and eastward velocities, just as we did for temperature above:

```
months = month(time(u), label = TRUE)

monthly.u = tapp(u,index=months,fun='mean',na.rm=T)
monthly.v = tapp(v,index=months,fun='mean',na.rm=T)

monthly.u
monthly.v
```

We now have two stacks of 12 layers each, one for each month. If we do raster algebra now, the function is applied to each pair of cells in the first layer of each stack, and then to each pair of cells in the second layer and so forth to give us 12 new layers:

```
monthly.velocity = sqrt(monthly.u^2 + monthly.v^2)
monthly.dir = atan2(monthly.u,monthly.v)

ggplot() +
  geom_spatraster(data = monthly.velocity) +
  facet_wrap(~lyr)
```




## And finally (more advanced, for interest!)....

In the Introduction we set ourselves the goal of recreating the map in Figure 1 which combines ocean temperature, current direction and strength to visualise the Gulf Stream. So how do we emulate it?

The easiest way to make this map using `ggplot` is with the help of another 'geom' called `geom_arrow` from the meteorological package `metR`.

```
# install.packages('metR')
library(metR)
```

`geom_arrow` does not work with spatial rasters so we need to make our current speed and direction rasters into a dataframe. We do this by stacking them, setting the layer names in the stack (which become column names in the dataframe), and then converting them to a dataframe (ensuring we set `xy = T` to retain the coordinates of each cell)

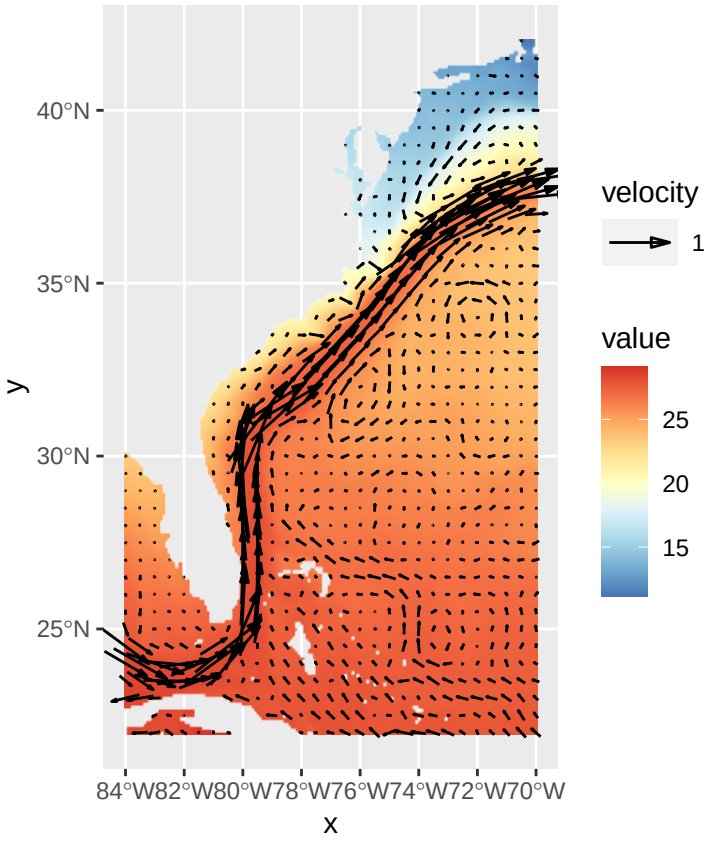
```
arrow.data = c(mean.velocity,mean.dir) %>%
             setNames(c('velocity','direction')) %>%
             as.data.frame(xy = T)
```

The current direction we calculated earlier is expressed in radians. We can convert this to degrees by multiplying by $180/\pi$ and then put it onto a 0-360 degree scale (rather than -180 to 180) using the modulo %% operator:

```
arrow.data$direction = (arrow.data$direction * 180/pi) %% 360
```

Now we add the flow arrows to the sea surface temperature plot we made earlier using `geom_arrow`. We set `skip = 5` to plot flow arrows for every fifth cell to thin them out a bit and use `scale_mag` from `metR` to make the length of the reference arrow in the figure legend equivalent to 1 m/s.

```
sst.plot +
 geom_arrow(data = arrow.data, aes(x=x,y=y,mag=velocity,angle=direction), skip=5) +
 scale_mag(max = 1)
```



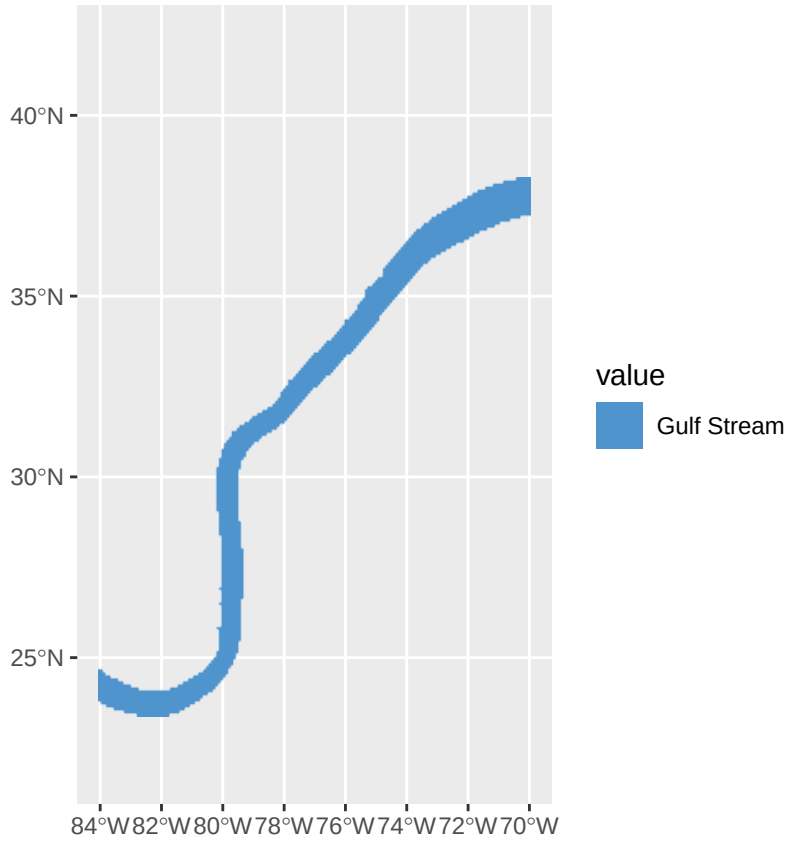

And voila! Using this method you can map any global ocean current or wind system you are interested in: visit https://oceancurrents.rsmas.miami.edu/ for inspiration.

---

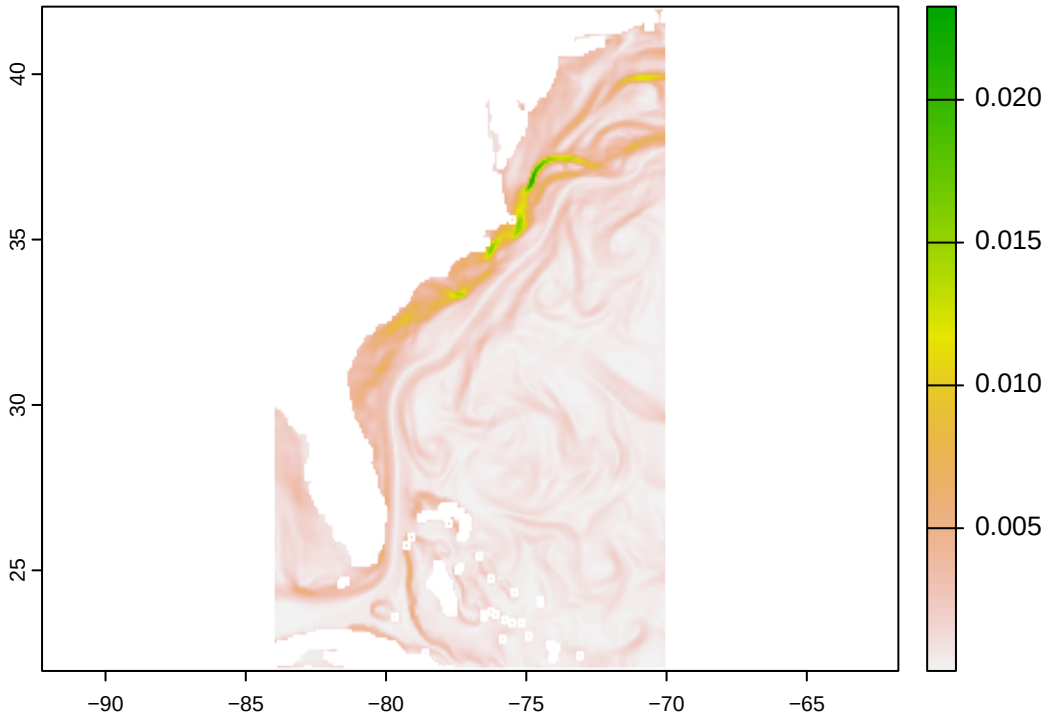
## Taking it further (optional activities for those who like a challenge!)

1. We noted that the Gulf Stream appears to be more pronounced outside of the summer months. Using operations introduced in Tutorials 3 and 4, see if you can map the monthly temperature anomalies across our study area (the anomaly is how much each cell differs from the overall mean for each month).

2. Using techniques you learned in Tutorial 3, see if you can select out and reclassify all cells with annual mean current velocity $> 0.5$ m/s to map the core of Gulf stream. You should end up with something like this:




3. Where the Gulf Stream collides with the cooler waters of the North Atlantic it generates major frontal systems characterized by steep gradients in sea surface temperature. These 'frontal zones' between water masses are important feeding habitat for many large marine vertebrates, including leatherback turtles. See if you can map the position of these frontal zones in each month (hint: the clue is in 'steep temperature gradients').


40
35
30
25
−90
−85
−80
−75
−70
−65
0.020
0.015
0.010
0.005