HMMoce: An R package for improved geolocation of archival-tagged fishes using a hidden Markov method

Camrin Braun, Ben Galuardi, Paul Gatti & Simon Thorrold

September 16, 2020

Contents

1 Summary

While the number of marine animals being tagged and tracked continues to grow, current satellite tracking techniques largely constrain meaninful inference to largescale movements of surface-dwelling species and are inherently prone to significant error. Hidden Markov models (HMMs) have become increasingly common in the analysis of animal movement data by incorporating underlying behavioral states into movement data. This discretized approach also provides efficient handling of grid-based oceanographic data and likelihood surfaces generated within the package. We present an open-source `R` package, `HMMoce`, that uses a state-space HMM approach to improve position estimates derived from electronic tags using three-dimensional oceanographic data. We demonstrate `HMMoce` with example blue shark (*Prionace glauca*) data that is

included in the package. Our findings illustrate how our software leverages all available tag data, along with oceanographic information, to improve position estimates of tagged marine species. For more details on these methods, including thorough references to the literature `HMMoce` is based on, please refer to (Braun et al., 2018a).

2 Introduction

There are many approaches to estimating animal movements from various types of tag data. The paradigm in fish tracking has been to use light levels to estimate position, but many species spend considerable time away from the photic zone. Diving behavior, like a typical diel vertical migration exhibited by deep diving swordfish, can render light geolocation useless. Yet, deep diving provides depth-temperature profile data recorded by the archival tag as it samples throughout a tagged individual's vertical movements. This sampling provides a unique signature through the oceanographic environment that can be leveraged to help constrain position. When combined with other tag-measured data streams like sea surface temperature (SST), light levels and maximum diving depth, we expect a unique combination of oceanographic characteristics to be diagnostic of an animal's location. Thus, `HMMoce` seeks to provide the framework for improving estimates of animal movements based on these oceanographic characteristics and strives to automate much of the data formatting and calculations in a transparent and flexible way.

`HMMoce` is an R package...

The package is presented in where its use is illustrated on example blue and mako shark PSAT data as well as compared to other commonly employed methods for geolocation of these and similar data. A subset of the data used in that paper is included as example data in the package.

3 Example application

Here we illustrate the use of the package with an example dataset from a blue shark tagged with a Wildlife Computers miniPAT in the NW Atlantic. Depending on which observation likelihoods (Sec. 3.3) you choose to generate for your species of interest, various tag-based data streams (e.g. light, SST) may be more or less useful. Our example dataset is from a blue shark which tends to occupy the epipelagic where the tag is able to consistently record high quality light and SST data that can be used for geolocation. Furthermore, this species tends to also make daily excursions, sometimes to >1000 m, which yields water column temperature profiles that can also be leveraged to improve geolocation estimates. Thus, it probably makes sense to take the time to build the 3D depth-temperature likelihoods in addition to the 2D, surface-only SST, for example. For other species or behaviors, such as benthic species that rarely interact with the sea surface, a bathymetry-based likelihood is probably more appropriate and perhaps SST is less informative.

3.1 Load and prepare the tag data

First, setup the start and end dates and locations. These are then used to build a vector of `POSIXct` timestamps that represent the desired time steps of the likelihoods and, ultimately, most probable track.

```
# SET START/END LOCATIONS iniloc is dataframe containing cols: day, month,
# year, lat, lon and rows: start, end
iniloc <- data.frame(matrix(c(13, 10, 2015, 41.3, -69.27, 10, 4, 2016, 40.251,
    -36.061), nrow = 2, ncol = 5, byrow = T))
names(iniloc) <- list("day", "month", "year", "lat", "lon")
tag <- as.POSIXct(paste(iniloc[1, 1], "/", iniloc[1, 2], "/", iniloc[1, 3],
    sep = ""), format = "%d/%m/%Y", tz = "UTC")
pop <- as.POSIXct(paste(iniloc[2, 1], "/", iniloc[2, 2], "/", iniloc[2, 3],
    sep = ""), format = "%d/%m/%Y", tz = "UTC")

# VECTOR OF DATES FROM DATA. THIS WILL BE THE TIME STEPS, T, IN THE
# LIKELIHOODS
dateVec <- seq.POSIXt(tag, pop, by = "24 hours")
```

3.1.1 Sea surface temperature data

Next, we load some tag data. Raw example data is included with the package and can be read by pointing to the "extdata" directory associated with the install of the `HMMoce` package. Helper functions are provided for directly reading data from Wildlife Computers tags, however `HMMoce` works with any tag data that can be coerced to the minimum required input data for the various observation likelihoods. Over time, these have been generalized from WC-specific to more flexible implementations for use with other manufacturers. The minimum required SST data for use in `HMMoce`, and thus the output from the WC helper function, is "Date", "Depth", and "Temperature". Date must be of class POSIXct. Depth represents the depth at which the corresponding sea surface temperature ("Temperature") was taken. These data requirements are easily fulfilled when working with other tag types (e.g. daily max temperature from Microwave X-tag). Currently the column "Depth" is not used but is retained here for user convenience in case, for example, it makes sense to filter the SST data due to SSTs being recorded deeper than what makes sense given the oceanography the tagged animal is in.

```
sstFile <- system.file("extdata", "141259-SST.csv", package = "HMMoce")
tag.sst <- read.wc(sstFile, type = "sst", tag = tag, pop = pop, verbose = F)
tag.sst <- tag.sst[, c("Date", "Depth", "Temperature")]
head(tag.sst)

                 Date Depth Temperature
1 2015-10-14 10:00:00     0        15.1
2 2015-10-15 10:00:00     0        14.6
3 2015-10-16 05:00:00     0        14.7
4 2015-10-17 07:00:00     0        14.2
5 2015-10-18 03:00:00     0        14.6
6 2015-10-19 09:00:00     0        13.6
```

3.1.2 Depth-temperature data

Next is some representation of water column structure. There are many options here based on tag model, animal behavior, etc. but ultimately the goal is to create a tag-based dataset that represents the thermal structure of the water column and contains columns at least for: "Date" (again POSIXct), "Depth", "MinTemp", and "MaxTemp". A fifth column called "MeanTemp" can be included which will be used as the temperature for likelihood construction. If "MeanTemp" doesn't exist, `HMMoce` will by default calculate the midpoint between min and max temperature for each measurement and use that for subsequent calculations.

By comparing tag-recorded thermal structure to the oceanographic environment, we can often add a very rich set of information for geolocation. For example, a tag sampling the thermal structure of the Sargasso Sea would yield very different results from that of the Gulf Stream or Labrador Sea (*e.g.* basking sharks Braun et al., 2018b). Tags like the example miniPAT used here provide summaries of depth-temperature profiles as a .csv called "-PDTs" for which there's a helper function.

```
# DEPTH-TEMPERATURE PROFILE DATA example is output from Wildlife Computer
# Portal pdt needs to contain at least: - Date (POSIXct) - Depth - MinTemp -
# MaxTemp - MeanTemp (optional): if meantemp doesn't exist for whatever
# reason, HMMoce will calculate the midpoint between min/max temps and use
# that

pdtFile <- system.file("extdata", "141259-PDTs.csv", package = "HMMoce")
pdt <- read.wc(pdtFile, type = "pdt", tag = tag, pop = pop, verbose = F)
pdt <- pdt[, c("Date", "Depth", "MinTemp", "MaxTemp")]
head(pdt)

          Date Depth MinTemp MaxTemp
1   2015-10-14     0    14.2    16.0
96  2015-10-14     8    13.6    16.0
```

```
191 2015-10-14    24    12.2    15.6
286 2015-10-14    48     9.8    14.4
2   2015-10-15     0    14.6    16.6
97  2015-10-15     8    14.6    16.6
```

Depending on tag type, there are many ways to represent this vertical structure. For example, miniPATs can also report summaries of depth-temperature time series which could be used to re-construct custom depth-temperature profiles. Similarly, other models such as the Microwave X-tag or various archival tags from Wildlife Computers or Lotek report high-resolution time series of depth and temperature. These can be used to construct custom depth-temperature profiles. A few things to keep in mind for building custom summaries of this type of data:

- vertical depth levels are ultimately compared to whatever depth levels are available in the environmental dataset you compare to for building likelihoods. Thus, it might make sense to make your custom depth levels match, for example, the HYCOM depth levels you will compare the tag data to
- temporal resolution of your summarized dataset can only be as high as the resolution of your 'dateVec' object that is the "temporal backbone" of this entire modeling process. Thus, it makes sense that the timescales and resolution of these match as much as possible.

The example below shows HMMoce functionality for coercing time series data to depth-temperature summary data using transmitted data from the example miniPAT; however, a similar approach can be used for time series from other tags. Note that the resulting data from this reformatting matches the format of the depth-temperature output above as we're assuming these are equivalent data products. Only one of these is necessary for subsequent likelihood construction. Due to data transmission constraints, it is usually best to use the summarized data for tags that report them (e.g. miniPAT), but constructing custom summaries like this can be useful for certain situations and are necessary for datasets that only have time series (e.g. archival tags).

```
# DEPTH-TEMPERATURE TIME SERIES DATA exampling showing how to coerce
# depth-temp time series to a PDT-like summarized product

tsFile <- system.file("extdata", "141259-Series.csv", package = "HMMoce")
## bandaid to access the series data without installing new HMMoce
tsFile <- "~/work/RCode/HMMoce/inst/extdata/141259-Series.csv"
ts <- read.table(tsFile, sep = ",", header = T)
ts$Date <- as.POSIXct(paste(ts$Day, ts$Time), format = "%d-%b-%Y %H:%M:%S",
    tz = "UTC")
ts <- ts[, c("Date", "Depth", "Temperature")]

pdt_dates <- seq.POSIXt(tag, pop, by = "day")  ## generate example daily summary of depth-temp profiles

## generate depth-temp summary from time series
pdt <- bin_TempTS(ts, out_dates = pdt_dates, bin_res = 25)
pdt <- pdt[, c("Date", "Depth", "MinTemp", "MaxTemp")]
head(pdt)

        Date Depth MinTemp MaxTemp
1 2015-10-15     0   15.35   15.50
2 2015-10-15    25   14.30   15.35
3 2015-10-15    50   12.75   13.75
4 2015-10-15    75   10.17   10.17
5 2015-10-19     0   13.66   13.71
6 2015-10-19    25   13.69   14.11
```

3.1.3 Light data

Light data has been at the core of geolocation estimates in marine and terrestrial environments for decades (hill ref). As such, there are a number of methods for using tag-based light measurements to generate

likelihoods of where on Earth those light measurements were recorded. Taking light measurements from the back of a marine animal can be particularly error-prone as that animal is moving through a range of turbid to clear water, diving from the photic zone to great depths, and at least indirectly experiencing cloud cover and many other factors that can affect light measurements. Here, we focus on two main types of light data one might get from an archival tag in the marine environment: raw light levels and light-based position estimates. This can be an important distinction when building likelihoods based on light (Sec. 3.3).

"Raw" light data

Raw light curves and/or sunrise and sunset times are recorded by most (if not all) archival tags. Our example miniPAT contains raw light curves but onboard processing has already determined sunrise and sunset times for us. If you go this route, required columns are "Date" (POSIXct) and "Type" where only "Dawn" and "Dusk" are recognized as valid types corresponding to sunrise and sunset, respectively.

```
lightFile <- system.file("extdata", "141259-LightLoc.csv", package = "HMMoce")
light <- read.wc(lightFile, type = "light", tag = tag, pop = pop, verbose = F)
## combine character vectors 'Day' and 'Time' to generate POSIXct object
light$Date <- lubridate::dmy_hms(paste(light$Day, light$Time, sep = " "))
light <- light[, c("Date", "Type")]
```

Our example miniPAT looks like this:

```
head(light)

                  Date Type
5  2015-10-15 09:56:15 Dawn
6  2015-10-15 21:52:30 Dusk
7  2015-10-16 09:45:00 Dawn
8  2015-10-16 21:50:00 Dusk
9  2015-10-17 09:57:30 Dawn
10 2015-10-17 21:48:45 Dusk
```

Similar measurements are taken by nearly every other archival or pop-up tag we're aware of and can thus be coerced to the same format as above.

Light-based position estimates

Many models of archival tag require manufacturer-specific post-processing which often generates light-based position estimates. These estimates often result in more realistic light-based likelihoods and are thus recommended over the "raw" light approach above. To incorporate light-based position estimates into the likelihood process, required columns are "Date" (POSIXct), estimated "Longitude", and "Error.Semi.minor.axis" which represents the error/uncertainty in the longitude estimate in METERS(!). With only these columns, you can generate longitude-only light-based likelihoods (i.e. if latitude estimates are unreliable and thus uninformative). To include latitude information and thus generate elliptical light-based likelihoods, the following additional columns are required: "Latitude", "Error.Semi.major.axis" (again, in meters), "Offset" (shift the ellipse this distance in METERS), "Offset.orientation" (currently 0 for North and 180 for South, no other orientation angles are currently supported). If your manufacturer didn't provide offset values, use 0 for both.

```
# LIGHT BASED POSITIONS FROM GPE2 (INSTEAD OF RAW LIGHTLOCS FROM PREVIOUS)
llFile <- system.file("extdata", "141259-Locations-GPE2.csv", package = "HMMoce")
lightloc <- read.table(llFile, sep = ",", header = T, blank.lines.skip = F)
lightloc <- lightloc[which(lightloc$Type != "Argos"), ]
lightloc <- lightloc[, c("Date", "Longitude", "Error.Semi.minor.axis", "Latitude",
    "Error.Semi.major.axis", "Offset", "Offset.orientation")]
lightloc$Date <- as.POSIXct(lightloc$Date, format = findDateFormat(lightloc$Date))
head(lightloc)

                 Date Longitude Error.Semi.minor.axis Latitude
1 2015-10-14 10:28:45 -68.20703                 71078     42.5
2 2015-10-14 22:36:15 -68.83478                 58991     45.0
```

```
3 2015-10-15 22:26:15 -68.13254                16835     49.0
4 2015-10-16 10:18:45 -68.86938                46343     45.0
5 2015-10-16 22:23:45 -68.65265                27293     41.5
6 2015-10-17 10:31:15 -68.34464                38261     45.5
  Error.Semi.major.axis Offset Offset.orientation
1               1583460 611160                  0
2               1527900 611160                  0
3               1416780 500040                  0
4               1277880 333360                  0
5                972300 277800                  0
6               1111200 222240                  0
```

** show what the ellipse looks like **

The column names above are derived from Wildlife Computers outputs but data from other manufacturers can be readily coerced to match this format. For example, Microwave Telemetry provides estimated latitude/longitude values based on their proprietary post-processing methods (in 'Lat&Long' sheet in an x-tag report). Since these aren't provided with error estimates, we can use empirical error estimates fixed at 0.7° longitude following Musyl et al. (2011) (although note that such error seems rather generous and is likely higher).

```
MWTdata  ## from Lat&Long sheet from x-tag
names(MWTdata) = c("Date", "Latitude", "Longitude")
## set fixed longitude error estimate in METERS
MWTdata$Error.Semi.minor.axis = 0.7 * 1000 * 111
```

3.1.4 Depth-only data

Most archival tags collect data on depth and many report statistics for depth, such as min and max, over a given summary period. It can often be useful to use at least the maximum depth over, for example, each day of a deployment to inform geolocation estimates. For benthic or bottom-oriented species, we can leverage both the max and min depths to constrain an animal's possible movements. To do this, we need some kind of summary of the depth data.

Our example miniPAT reports a summary sheet that already contains these values:

```
mmdFile <- system.file("extdata", "141259-MinMaxDepth.csv", package = "HMMoce")
mmd <- read.table(mmdFile, sep = ",", header = T, blank.lines.skip = F)[, c("Date",
    "MinDepth", "MaxDepth")]
mmd$Date <- as.POSIXct(mmd$Date, format = findDateFormat(mmd$Date))
head(mmd)

        Date MinDepth MaxDepth
1 2015-10-13    0.375       51
2 2015-10-14    0.000       48
3 2015-10-15    0.000       88
4 2015-10-16    0.000       56
5 2015-10-17   16.000       16
6 2015-10-18    0.000       16
```

However, given a depth time series data stream from your tag, which is far more common, it is trivial to generate your own custom depth summary statistics. The benefit of the latter approach is full control over the temporal resolution of the summary if, for example, you wanted to generate likelihoods at finer timesteps than the onboard processing of depth summary statistics would allow.

```
seriesFile <- system.file("extdata", "141259-Series.csv", package = "HMMoce")
series <- read.table(seriesFile, sep = ",", header = T, blank.lines.skip = F)[,
    c("Day", "Time", "Depth", "Temperature")]
```

```
series$Date <- as.Date(as.POSIXct(paste(series$Day, series$Time), format = "%d-%b-%Y %H:%M:%S",
    tz = "UTC"))
mmd <- series %>% group_by(Date) %>% dplyr::summarise(n = n(), MinDepth = min(Depth,
    na.rm = T), MaxDepth = max(Depth, na.rm = T), .groups = "keep")
head(mmd)

# A tibble: 6 x 4
# Groups:   Date [6]
  Date           n MinDepth MaxDepth
  <date>     <int>    <dbl>    <dbl>
1 2015-10-13   219      0.5     51
2 2015-10-14   240      0.5     34.5
3 2015-10-15   480      0.5     80
4 2015-10-16    48      2.5     17
5 2015-10-17    96      0.5     26.5
6 2015-10-19   432      0.5     51
```

3.2 Environmental data

Now that the tag data is prepared, the next major step is to acquire all the necessary environmental data to compare the tag data to. Depending on which likelihoods you choose to build, the required environmental data will vary. Light-based likelihoods require no environmental data as the spatial variation in sunrise and sunset times are known.

3.2.1 Spatial limits

First, establish the spatial bounds of this model run. How you do this can vary widely based on species, their movement ecology, and the quality of the tag data used. This step is **critical** in the model setup because it **must** incorporate the complete geographic limits of your animal(s) movements! These limits are used for downloading environmental data and, ultimately, serve as the bounds for the model. Thus, err on the side of making the bounds too large to be sure the extent of movements are adequately captured. If they aren't, the estimated movements will likely run into the edges of your spatial limits (specified here), and you will have to start over. The tradeoff as model bounds expand is, of course, computational demands increase as larger grids are computed. In the end, the choice of spatial bounds comes down to expert opinion so choose carefully.

This step is typically accomplished best by a combination of expert opinion (thats you!) and by looking at the spatial bounds of tagging and pop-up locations and longitude estimates (if you have them). If you don't have position estimates to start with, do some preliminary plotting of tag data (such as SST) to try to constrain where you think the animal went. There's no harm in having to come back to this when your model runs into the boundaries except that you'll have to download all the environmental data again which, you will soon find out, takes time. To that end, be smart about setting spatial limits when you plan to analyze data for a group of tag datasets. Find the largest common grid and use that for all analyses. That means you only have to download the oceanographic data once!

Here, we set our bounds over a large portion of the NW Atlantic as blue sharks are known to be highly migratory, and this individual could easily cover much of this area over the duration of the tag deployment. This decision was ultimately made by comparing the tag and pop-up locations to the depth-temperature information recorded onboard the tag to explore the likely water masses the animal encountered and thus the extent of movement.

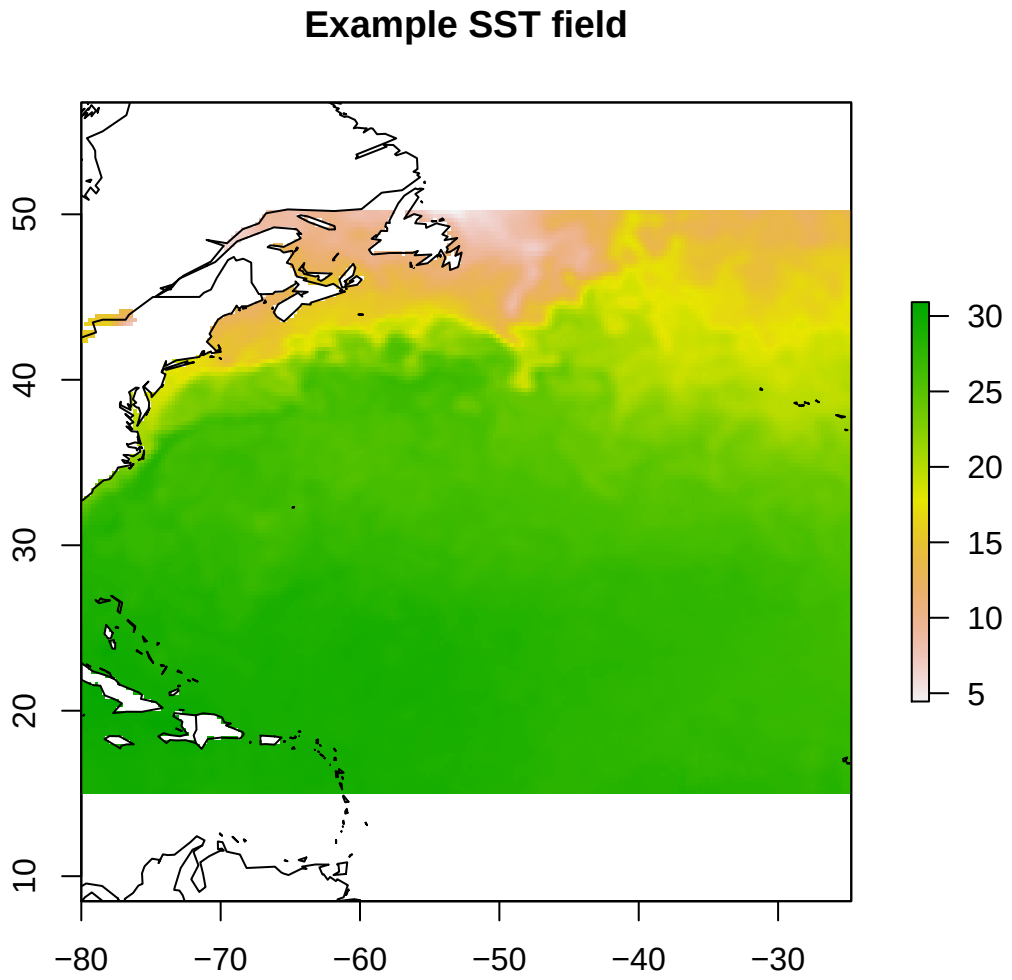
```
# SET SPATIAL LIMITS these are the lat/lon bounds of your study area (e.g.
# where you think the animal went)
sp.lim <- list(lonmin = -80, lonmax = -25, latmin = 15, latmax = 50)

## setup the spatial grid to base likelihoods on
locs.grid <- setup.locs.grid(sp.lim, res = "quarter")
```

If you plan to analyze multiple tag datasets over a similar time period and spatial domain, consider that before downloading the environmental datasets to save yourself from having to change spatial/temporal bounds later and download everything again. For a group of tags, combine the unique dates and expected spatial bounds across all tags to find a common spatial extent before downloading the environmental data (e.g. SST).

3.2.2 SST data

There are a number of SST products out there that meet our needs. While modeled SST products work fine, we typically use remote-sensing products. There are 3 products currently built into the `get.env` functionality within `HMMoce` (see `?get.env`). Here we'll use the Optimum Interpolation SST (OISST) from NOAA. Note the tradeoffs associated with different SST products such as gaps, gap-filling techniques/interpolations, product resolution and temporal coverage/availability of a given product. Here we get one file as an example:

```
## here we get use a ridculously small spatial extent for this species, just
## to show some example environmental data
udates <- seq.Date(as.Date(tag), as.Date(pop), by = "day")
sst.dir <- paste0(dir, "/EnvData/sst/")
if (!dir.exists(sst.dir)) dir.create(sst.dir, recursive = TRUE)
if (!file.exists(paste0(sst.dir, "oisst_", udates[1], ".nc"))) get.env(udates[1],
    filename = "oisst", type = "sst", sst.type = "oi", spatLim = sp.lim, save.dir = sst.dir)

sst <- raster::raster(paste0(sst.dir, "oisst_", udates[1], ".nc"))

Loading required namespace:  ncdf4

Error in as.Date(time, origin = startDate) : object 'startDate' not found

plot(sst, main = "Example SST field")
world(add = T)
```

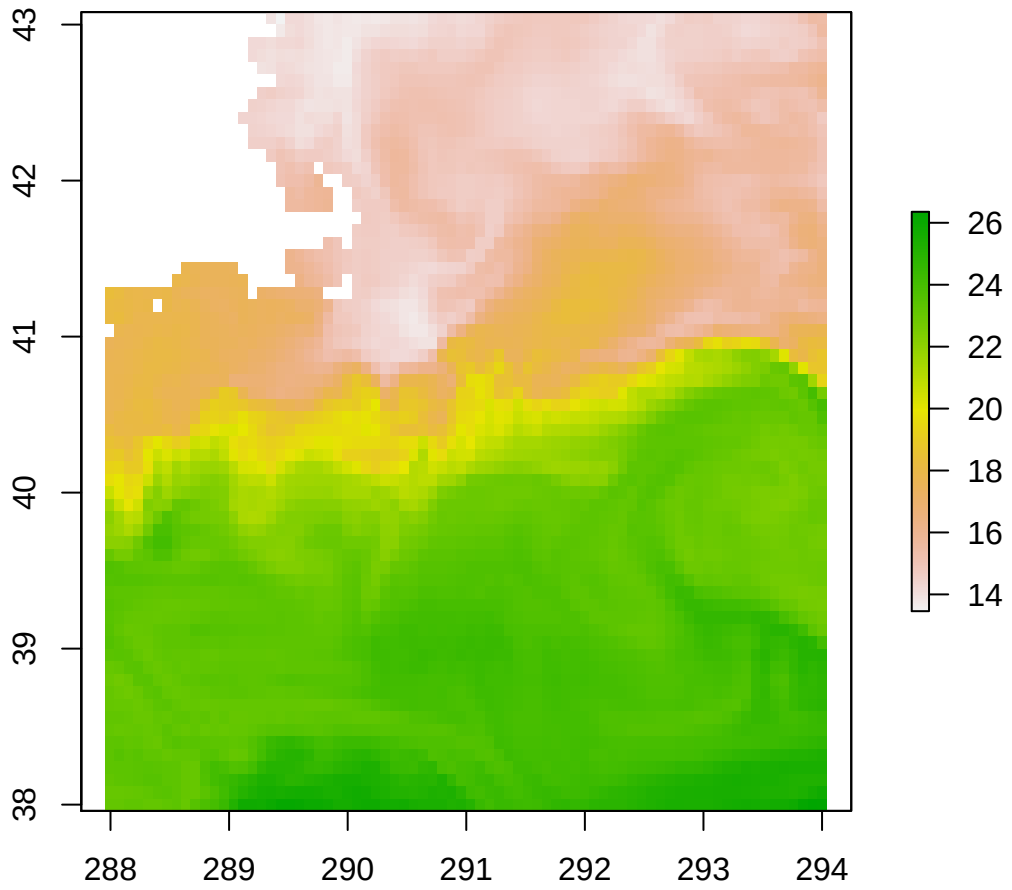


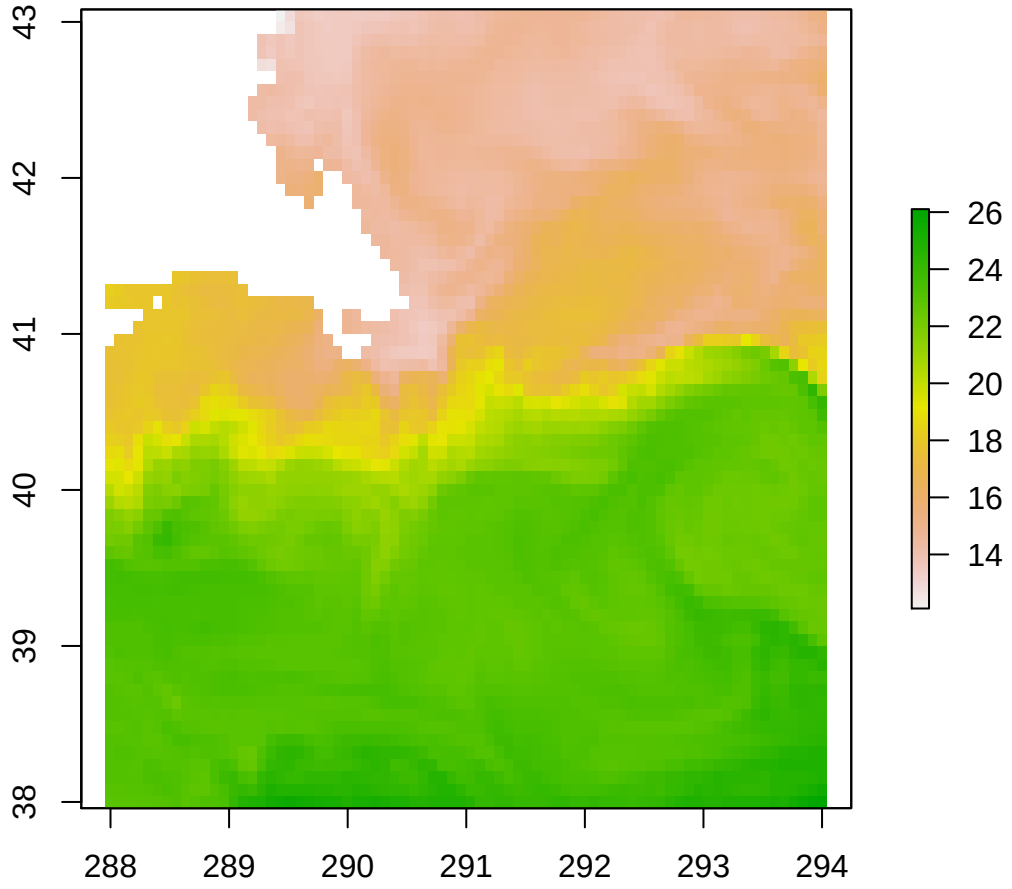
3.2.3 Depth-temperature data

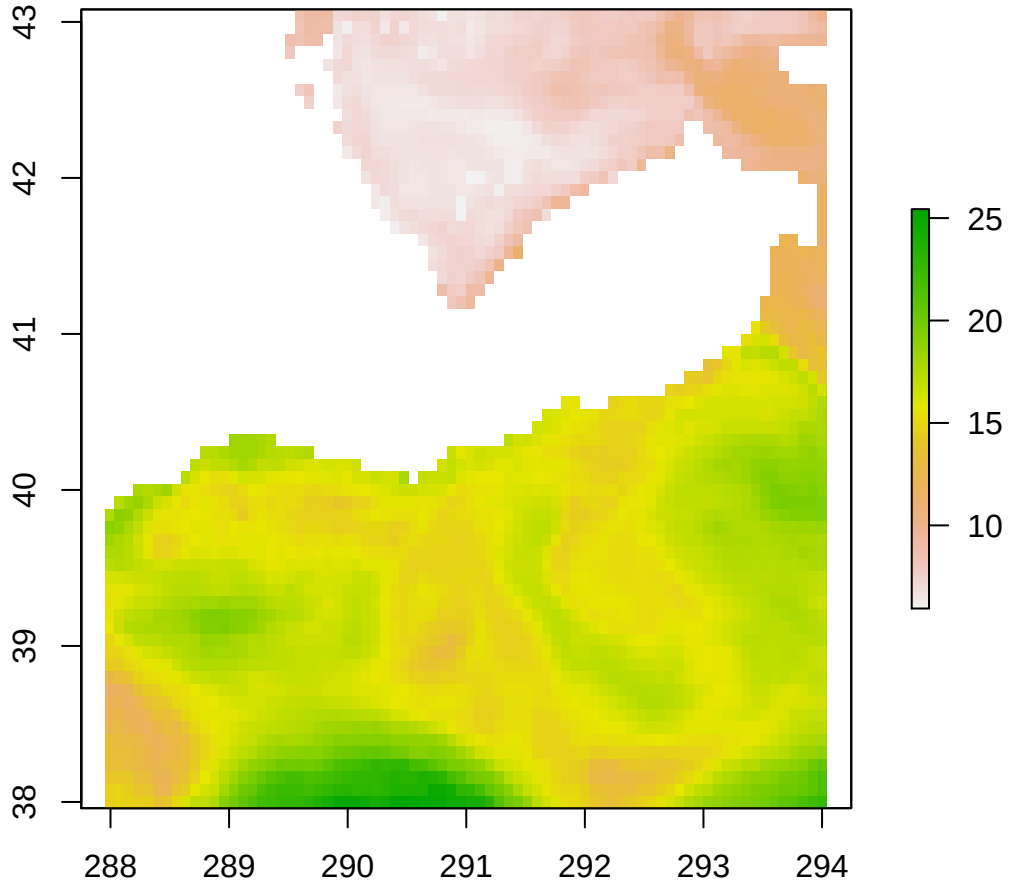
As with SST above, there are a number of ways to represent temperature of the 3D ocean. While observations are nice, they are typically too sparse to generate a reliable grid on which we can calculate likelihoods. Thus, our only good option (usually) is to use oceanographic models. Data-assimilating models constantly "nudge" the model outputs to match reality and `HMMoce` includes functionality for accessing outputs of the HYbrid Coordinate Ocean Model (HYCOM). For most applications, models such as HYCOM are more likely to generate realistic likelihood results when comparing *in situ* tag data to a dynamic ocean. In principle, any representation of the 3D ocean can be used for likelihood construction as is done here with HYCOM.

```
## you need some representation of environmental depth-temperature in this
## case we're using hycom

sp.lim_small <- list(lonmin = -72, lonmax = -66, latmin = 38, latmax = 43)
dir <- getwd()
hycom.dir <- paste0(dir, "/EnvData/hycom/")
if (!dir.exists(hycom.dir)) dir.create(hycom.dir, recursive = TRUE)
if (!file.exists(paste0(hycom.dir, "hycom_", udates[1], ".nc"))) get.env(udates[1],
```

```
    filename = "hycom", type = "hycom", spatLim = sp.lim_small, save.dir = hycom.dir)

b <- raster::brick(paste0(hycom.dir, "hycom_", udates[1], ".nc"))

Loading required namespace:  ncdf4

## plot top 5 depth levels from HYCOM par(mfrow=c(3,1))
for (i in c(1, 10, 20)) {
    plot(b[[i]])
    world(add = T)
}
```




43
42
41
40
39
38
288
289
290
291
292
293
294
26
24
22
20
18
16
14

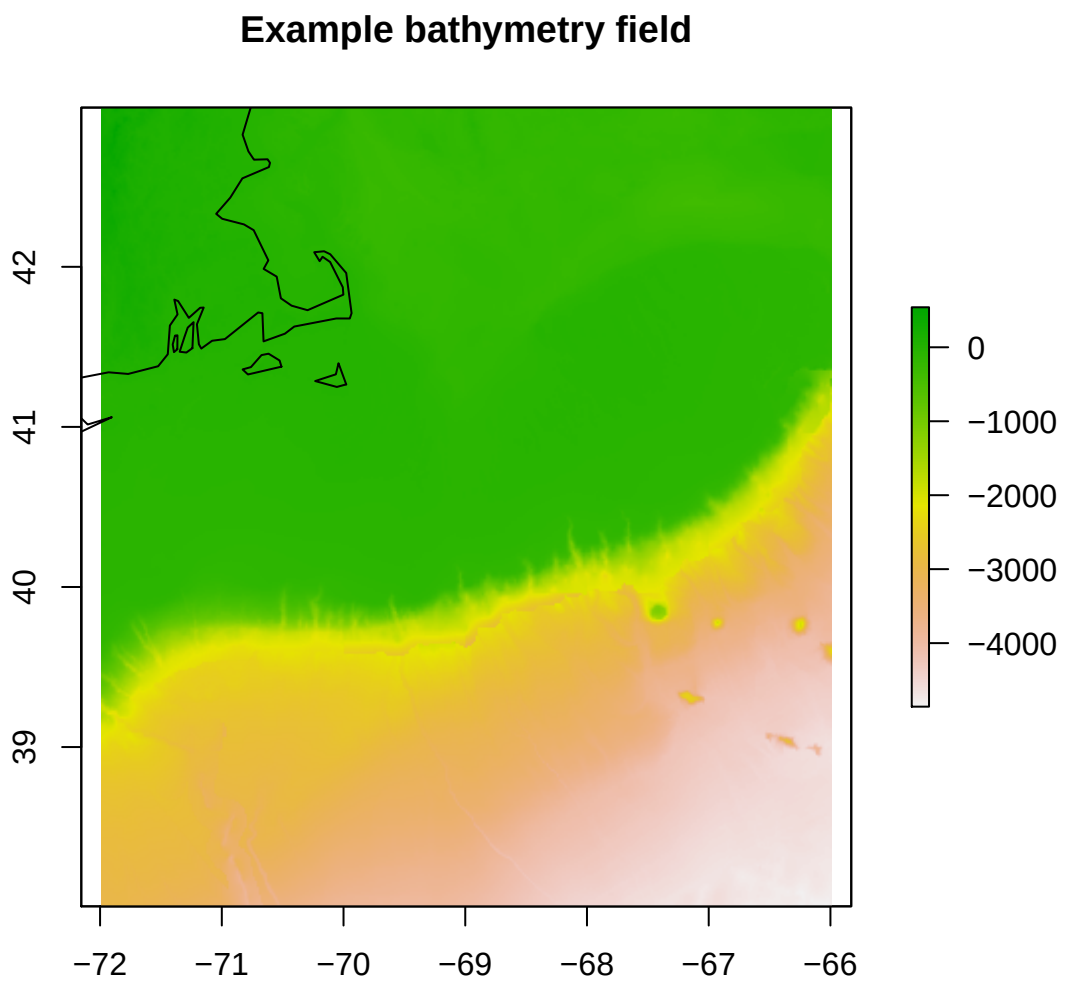


3.2.4 Bathymetry

The last "environmental" dataset we need is bathymetry. Bathymetry has multiple uses in `HMMoce` but is primarily used as a mask that eliminates portions of a likelihood grid that are shallower than the maximum depth the tag recorded during a given timestep (*i.e.* the likelihood has to incorporate bottom depth at least as deep as the animal's tag recorded). For primarily benthic species, bathymetry can also be used to generate a likelihood based on the tag-recorded depth compared to bottom depth.

```
bathy.dir <- paste0(dir, "/EnvData/bathy/")
if (!dir.exists(bathy.dir)) dir.create(bathy.dir, recursive = TRUE)

if (!file.exists(paste0(bathy.dir, "bathy.nc"))) {
    bathy <- HMMoce::get.bath.data(sp.lim_small$lonmin, sp.lim_small$lonmax,
        sp.lim_small$latmin, sp.lim_small$latmax, folder = bathy.dir, res = 1)
} else {
    ## OR (once downloaded and reading the .nc later)
    bathy <- irregular_ncToRaster(paste0(bathy.dir, "bathy.nc"), varid = "topo")
}
```

```r
## example bathymetry data
plot(bathy, main = "Example bathymetry field")
world(add = T)
```



3.3 Observation likelihoods

The basic premise of the observation likelihoods is that we compare a tag-based data stream to some representation of the environment to generate a likelihood surface. This comparison generates information about the likely location of the animal in the global ocean. Light levels and SST are the current paradigm for fish tracking and are the most straightforward to use for positioning. Three-dimensional depth-temperature data are more complex and computationally intensive but can provide rich information about oceanographic characteristics and water masses that animals experience as they move (*e.g.* Braun et al., 2018b). Each data type has its own likelihood function(s) that does the grunt work for you. It often makes sense to build all the appropriate likelihoods given your tag data and study species and use model selection to determine the "best" combination of likelihoods (for example, see Braun et al., 2018a, and section **??**).

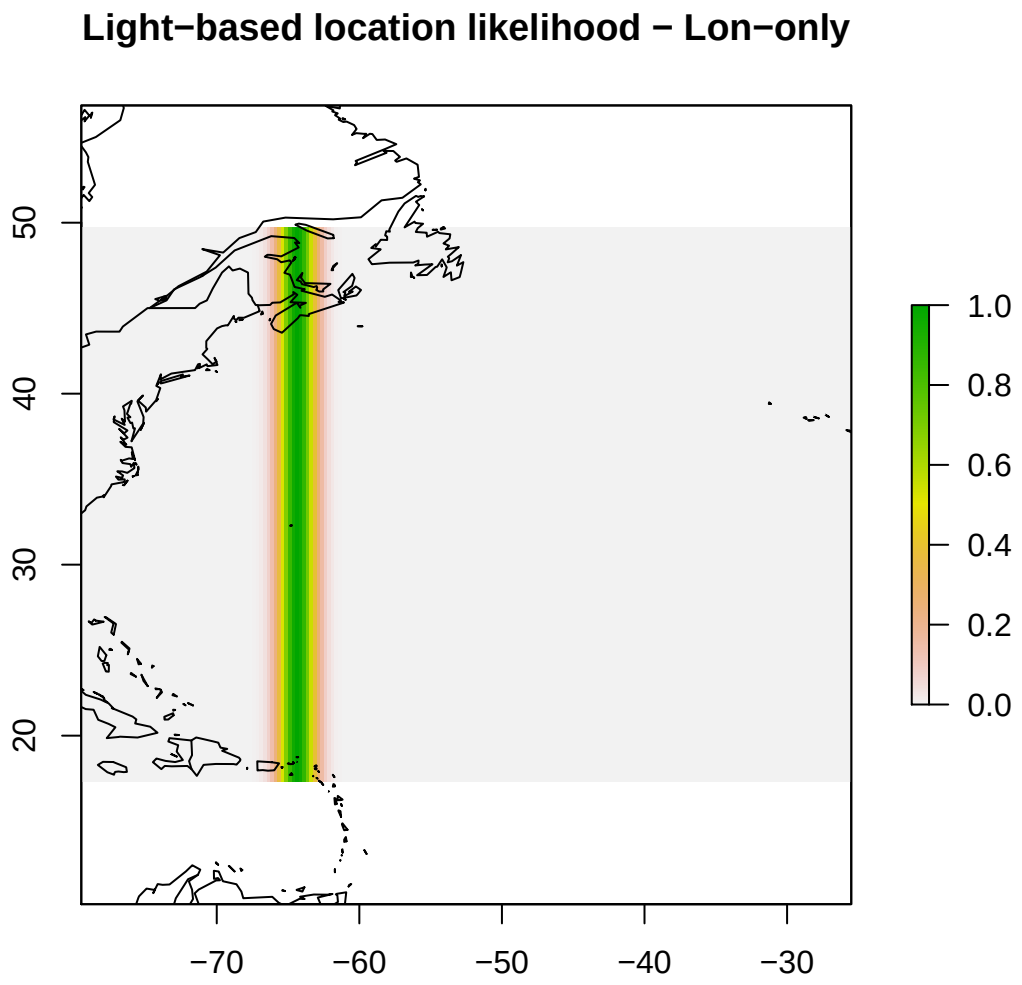
3.3.1 Light likelihood

Light-based geolocation is a complicated topic with a myriad of potential implementations for generating light-based likelihoods. For now, `HMMoce` has two main treatments of light to inform geolocation estimates (Sec. 3.1.3). The simplest is using tag-based light levels to estimate sunrise and sunset times and thus position (`calc.srss`). This is usually an overly simplistic treatment of this data so we often 1) throw out latitude estimates and only keep longitude and 2) get a lot of bad location information, particularly from species that don't frequent the photic zone. For example, a surface-oriented species that dives below the photic zone 30 minutes before sunset would generate an artificial sunset time 30 minutes early, causing $\sim 8^{\circ}$ longitudinal difference in position estimate! The other approach currently implemented in `HMMoce` generates likelihoods based on existing light-based position estimates(`calc.lightloc`). Most manufacturers either provide the analysis or the analysis software to generate light-based position estimates. For example, Wildlife Computers uses a thresholding algorithm (hill ref) in their GPE2 software that allows user-controlled QC of light curves in a GUI. While this software has been replaced by their cloud-based GPE3 algorithm, GPE2 remains the tool of choice for this specific analysis as the user retains some control over the conversion of light curves to location estimates and the ability to visually check the quality of these curves. Other manufacturers generate similar light-based position outputs such as Lotek's daily summary for archival tags. Additional light-based options will be implemented in the future as they become available and proven for marine animal telemetry. To that end, we are happy to receive feedback via Github on methods to include or simply submit a pull request.

```
# LIGHT-BASED LIKELIHOODS
L.light.srss <- calc.srss(light, locs.grid = locs.grid, dateVec = dateVec, res = 1,
    focalDim = 15)
L.light <- calc.lightloc(lightloc, locs.grid = locs.grid, dateVec = dateVec,
    errEll = TRUE)
```

Here's an example of what the lon-only likelihoods look like:

```
plot(L.rasters$L.light[[12]], main = "Light-based location likelihood - Lon-only")
world(add = T)
```



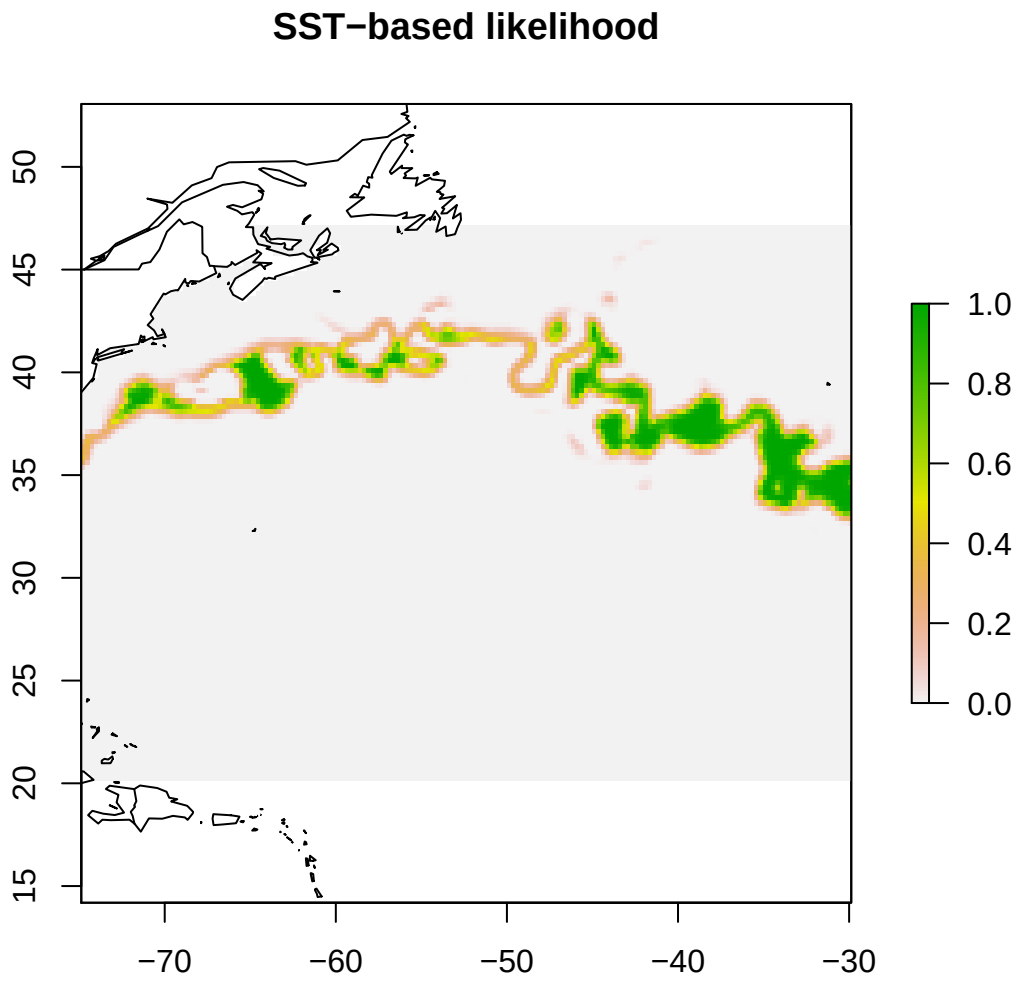
3.3.2 SST likelihood

SST-based likelihoods are perhaps among the most simple, and often informative, for geolocation as SST can be highly dynamic and exhibit strong gradients over relatively small spatial scales. If your study animal regularly visits the surface and thus collects SST data, it's likely that including that information will improve your geolocation results. In `HMMoce`, tag-based SST data is represented as a daily range in SST ± error (currently defaults to 1%) and that SST envelope is then compared to a remotely-sensed SST product to generate a likelihood surface.

```
# SST LIKELIHOODS
L.sst <- calc.sst(tag.sst, filename = "oisst", sst.dir = sst.dir, dateVec = dateVec[1:5],
    sens.err = 1)
L.sst <- calc.sst.par(tag.sst, filename = "oisst", sst.dir = sst.dir, dateVec = dateVec,
    sens.err = 1, focalDim = 3, ncores = 2)
```

Here's an example of what these look like:

```
## here's what the likelihoods look like
plot(L.rasters$L.sst[[12]], main = "SST-based likelihood")
world(add = T)
```



3.3.3 3D depth-temperature likelihood

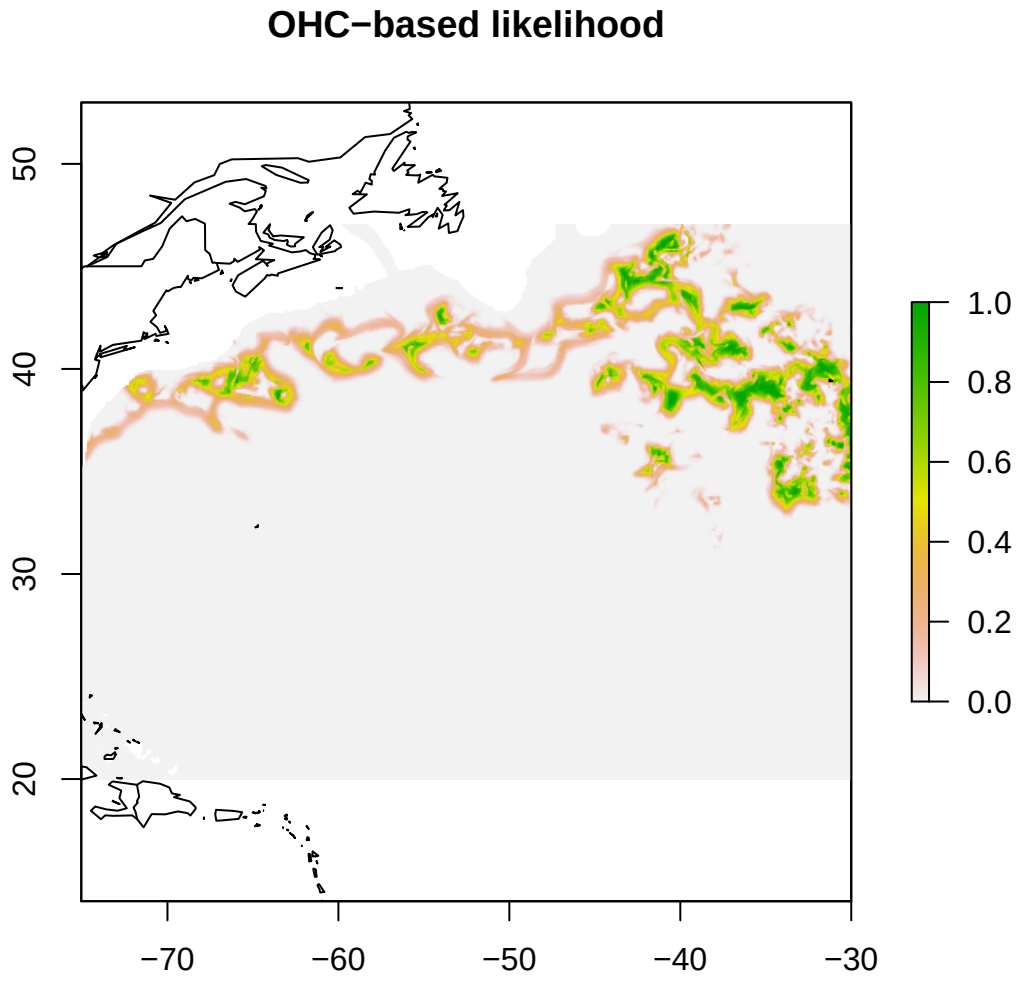
The depth-temperature based likelihoods allow users to generate 3D likelihoods for tagged animals to improve position estimates, which is particularly useful for study species that rarely visit the photic zone during the day ((e.g. swordfish, Neilson et al., 2009; Braun et al., 2019) or spend considerable periods of time in the mesopelagic (e.g. basking sharks, Skomal et al., 2009; Braun et al., 2018b). In `HMMoce`, there are currently two approaches to using the 3D data for geolocation. The first follows (Luo et al., 2015) by integrating profile data to calculate Ocean Heat Content (OHC). We integrate tag-based depth-temperature data from a given isotherm (default or user-selected) to the surface to calculate the "heat content" of that layer measured by the tagged animal. Similarly, we perform the same integration on the model ocean as represented in the HYbrid Coordinate Ocean Model ([HYCOM](http://hycom.org/)) and compare the two integrated metrics to generate a likelihood surface representing the animal's estimated daily position. The second approach is to use the profile in 3D space and compare it to oceanography at measured depth levels. This uses the

same tag-based depth-temperature data (not integrated) and compares it to modeled oceanography (such as HYCOM). In either case, we use a linear regression to predict the tag-based temperature at the standard depth levels measured in the oceanographic datasets (this is why we generate custom depth-temperature tag data to match oceanographic depth levels whenever possible, 3.1). Then a likelihood is calculated in the same fashion by comparing temperature from the tag to ocean temperature at each depth level and resulting likelihood layers are multiplied across depth levels to result in a single daily likelihood layer based on the tagged animal's dive data. Because these can be very computationally demanding, parallelized calculation functions are available for all three depth-temperature based likelihood functions.

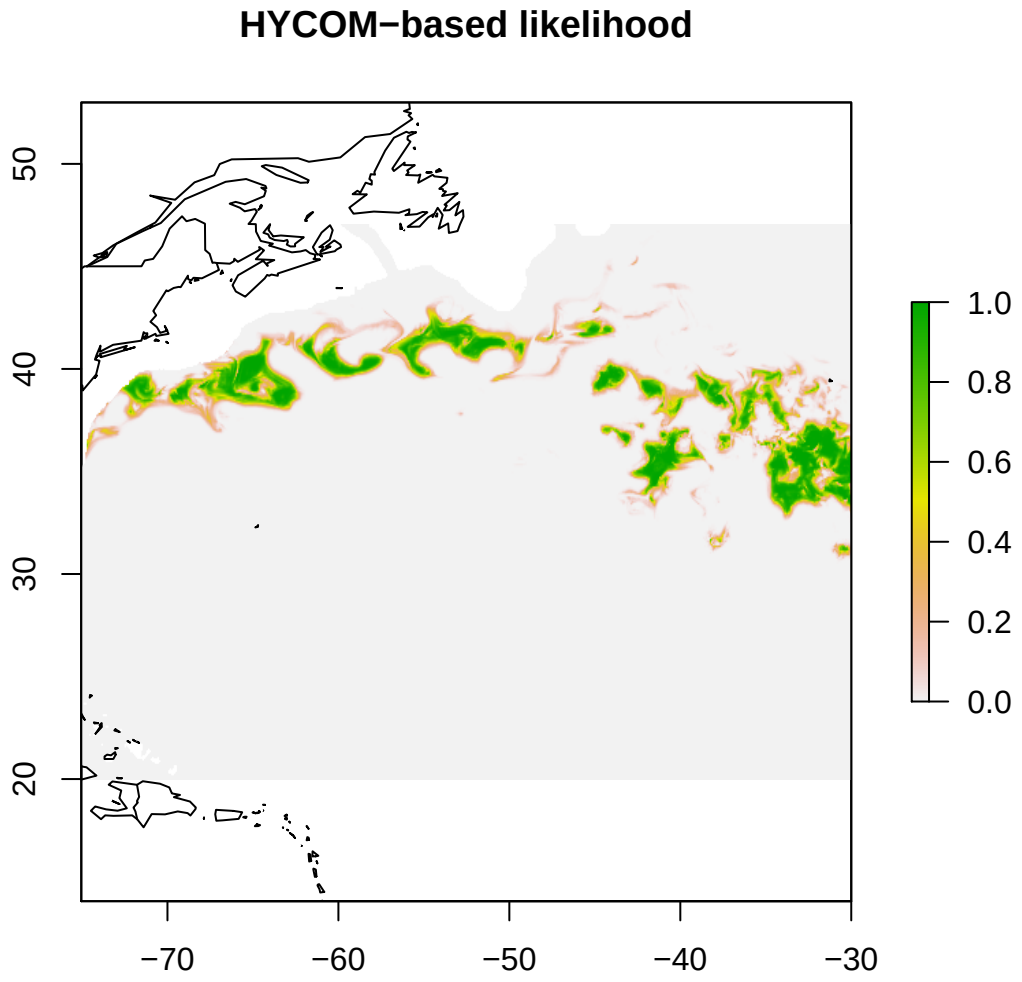
```
# OCEAN HEAT CONTENT (INTEGRATED PDTS)
L.ohc <- calc.ohc(pdt, filename = "hycom", ohc.dir = hycom.dir, dateVec = dateVec,
    isotherm = "", use.se = F)
## the parallel version: L.ohc <- calc.ohc.par(pdt, filename='hycom', ohc.dir
## = hycom.dir, dateVec = dateVec, isotherm = '', use.se = F)

# HYCOM PROFILE BASED LIKELIHOODS
L.hycom <- calc.hycom(pdt, filename = "hycom", hycom.dir, focalDim = 9, dateVec = as.POSIXct(dateVec),
    use.se = T)
## the parallel version: L.hycom <- calc.hycom.par(pdt, filename='hycom',
## hycom.dir, focalDim = 9, dateVec = as.POSIXct(dateVec), use.se = T)
```

Here's an example of what these look like:

```
## here's what the likelihoods look like
plot(L.rasters$L.ohc[[13]], main = "OHC-based likelihood")
world(add = T)
```




```
plot(L.rasters$L.hycom.se[[13]], main = "HYCOM-based likelihood")
world(add = T)
```



3.3.4 Bathymetry likelihood

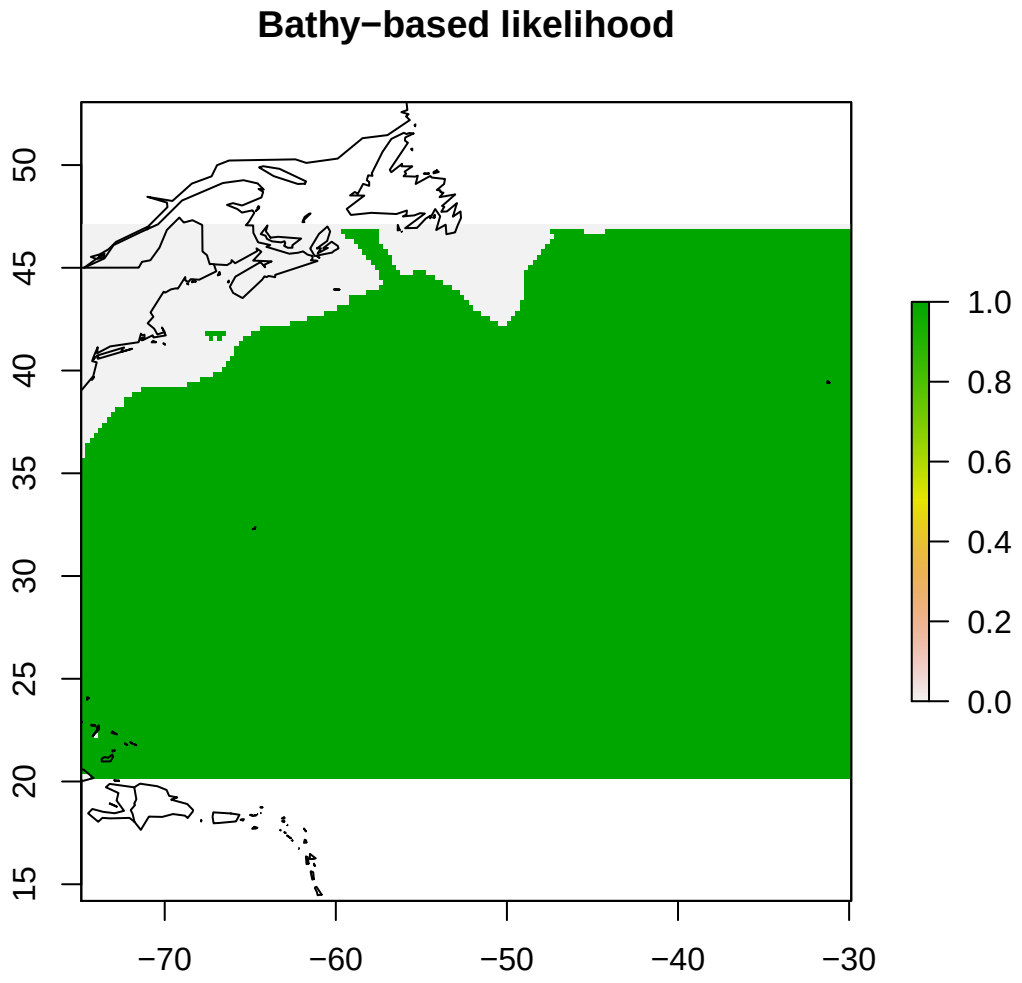
Bathymetry can be used to further constrain the likely location of a tagged animal. The most common implementation of this is to use the maximum measured depth in each time step, compared to available bathymetry, to ensure the most probable track occupies water deep enough for the animal to make the observed vertical movements (in the implementation of `calc.bathy` this is referred to as `lik.type = 'max'`). The same approach could be used for a benthic species, but it may often make more sense for those species with reasonable certainty of interacting with the bottom to use the same approach as previous likelihoods and calculate a formal likelihood (`lik.type = 'dnorm'`). Since most bathymetry data is rather high resolution relative to the other environmental variables used here to calculate liklihoods, it usually will speed things up to down-sample the bathymetry before calculating the likelihoods. The resulting likelihood will have to be re-sampled later anyway to match other likelihood grids.

```
## bathymetry based likelihood resample bathy to a more reasonable (coarse)
## grid for likelihood calculations hi-res bathy grid will work but will take
## longer
bathy_resamp <- raster::resample(bathy, L.sst)  # or whatever grid makes sense to resample to
```

```
L.bathy <- calc.bathy(mmd, bathy_resamp, dateVec, focalDim = 3, sens.err = 5,
    lik.type = "dnorm")
L.bathy <- calc.bathy.par(mmd, bathy_resamp, dateVec, focalDim = 3, sens.err = 5,
    lik.type = "max", ncores = 4)
```

Here's an example of what these look like:

```
## here's what the likelihoods look like
plot(L.rasters$L.bathy[[12]], main = "Bathy-based likelihood")
world(add = T)
```



3.3.5 Bottom temperature likelihood

Gradients in bottom temperature have been used to inform geolocation of benthic species, particularly in relatively shallow and/or enclosed basins (*e.g.* cod Le Bris et al., 2013). The implementation of this likelihood in `HMMoce` is similar to SST-based likelihoods but the surface temperature grid has simply been exchanged for a bottom temperature grid. Typically the environmental data for generating these likelihoods is derived from oceanographic models or some aggregate (often interpolated) set of observations.

```
## bottom temperature based likelihood
L.bt <- calc.bottomTemp(tag.bt, dateVec, focalDim = 3, sens.err = 1, bt.dir = bt.dir,
    filename = "bottomT", varName = "Temperature")
# L.bt <- calc.bottomTemp.par(tag.bt, dateVec, focalDim = 3, sens.err = 1,
# bt.dir = bt.dir, filename = 'bottomT', varName = 'Temperature', ncores =
# 4)
```

3.3.6 Overall observation likelihood

The final step in generating likelihoods is to combine the likelihoods of interest to generate an overall observation-based likelihood for each time step. This is another key decision point in the geolocation process as there are a number of likelihood combinations that might make sense to base the geolocation estimates on. Here, we show just an example of one combination of likelihoods but in practice we usually iterate through multiple combinations and use a model selection approach to determine the "best" model. The final step in this section is to generate the overall observation likelihood as a combination of the selected likelihoods. At this point, you can also supply any other known locations (*e.g.* from sightings, acoustic detections, etc).

```
# COMBINE LIKELIHOOD MATRICES make list of rasters
L.rasters <- list(L.light, L.sst, L.ohc)

## typically these will need to be resampled to have matching resolution and
## extent
resamp.idx <- which.max(lapply(L.rasters, FUN = function(x) raster::res(x)[1]))
L.res <- resample.grid(L.rasters, L.rasters[[resamp.idx]])

## good idea to save these likelihood rasters at some point to keep from
## having to re-calculate them if (when) R crashes

## finally, combine the likelihoods into a master observation likelihood that
## will be used for the modeling
L <- make.L(ras.list, iniloc, dateVec, known.locs = NULL)
```

Here's an example of what these look like:

```
## here's what the likelihoods look like
image.plot(g$lon[1, ], g$lat[, 1], L, main = "Example overall likelihood at single time step")
world(add = T)

## not working yet
```

3.4 Model fitting

3.4.1 Parameter estimation

`HMMoce` currently incorporates a handful of parameters in the HMM modeling steps. The framework currently allows 1-2 behavior states and the necessary associated parameters: diffusion or movement speed (sigmas) and state-switching probability. Obviously if only 1 behavior state is desired, state-switching probability becomes not applicable.

We currently include two general approaches to parameter estimation: gradient-based (*e.g.* `optim`, `nlminb`) or an evolutionary / genetic algorithm. The former is faster while the latter appears to result in better estimates.

In this example, we show each of the available parameter estimation methods using a two-state model followed by one example of a single state model. In most cases, we provide upper and lower bounds on the parameters as well as initial values (the latter is not required in the genetic algorithm). Finally, parameters can be estimated much more rapidly using a more coarse grid than full resolution, however results are rarely as robust as the full resolution grids.

```
## if you want to try coarse grids for parameter estimation
## use coarse.L() and supply the outputs in place of L and g below
L.mle <- coarse.L(L, L.res$L.rasters)$L.mle
g.mle <- coarse.L(L, L.res$L.rasters)$g.mle

## opt.params is a wrapper for the various optimization routines in HMMoce
pars.optim <- opt.params(pars.init = c(2,.2,.6,.8),
                         lower.bounds = c(0.1, 0.001, .1, .1),
                         upper.bounds = c(6, .6, .9, .9),
                         g = L.res$g,
                         #g = g.mle,
                         L = L,
                         #L = L.mle,
                         alg.opt = 'optim',
                         write.results = FALSE)
## about 22 mins on example blue shark 141259 with full grid

pars.optim.mle <- opt.params(pars.init = c(2,.2,.6,.8),
                         lower.bounds = c(0.1, 0.001, .1, .1),
                         upper.bounds = c(6, .6, .9, .9),
                         #g = L.res$g,
                         g = g.mle,
                         #L = L,
                         L = L.mle,
                         alg.opt = 'optim',
                         write.results = FALSE)

## about 1.5 mins on example blue shark 141259 with coarse grid

## nlminb is also supported in HMMoce but testing suggests this is rarely the best choice due to lack of convergenc
pars.nlminb <- opt.params(pars.init = c(2,.2,.6,.8),
                          lower.bounds = c(0.1, 0.001, .1, .1),
                          upper.bounds = c(5, .5, .9, .9),
                          g = L.res$g,
                          #g = g.mle,
                          L = L,
                          #L = L.mle,
                          alg.opt = 'nlminb',
                          write.results = FALSE)
## about 30 mins on blue shark 141259

pars.ga <- opt.params(pars.init = c(2,.2,.6,.8),
                        lower.bounds = c(0.1, 0.001, .1, .1),
                          upper.bounds = c(6, .6, .9, .9),
                          g = L.res$g,
                          #g = g.mle,
                          L = L,
                          #L = L.mle,
                          alg.opt = 'ga',
                          write.results = FALSE,
                          ncores = ceiling(parallel::detectCores() * .9))
  ## about 1.6 hrs on blue shark 141259 w 15 cores

  pars.ga.mle <- opt.params(pars.init = c(2,.2,.6,.8),
                        lower.bounds = c(0.1, 0.001, .1, .1),
                        upper.bounds = c(6, .6, .9, .9),
                        #g = L.res$g,
                        g = g.mle,
```

```
                        #L = L,
                        L = L.mle,
                        alg.opt = 'ga',
                        write.results = FALSE,
                        ncores = ceiling(parallel::detectCores() * .9))
   ## about 2 mins with MLE grid but results way different

  ## example using the genetic algorithm and only one behavior state
  pars.ga.one <- opt.params(pars.init = c(2),
                        lower.bounds = c(1),
                        upper.bounds = c(8),
                        g = L.res$g,
                        #g = g.mle,
                        L = L,
                        #L = L.mle,
                        alg.opt = 'ga',
                        write.results = FALSE,
                        ncores = 4)
  ## about 7 hrs on blue shark 141259 w 4 cores
```

3.4.2 HMM Filter / Smoother

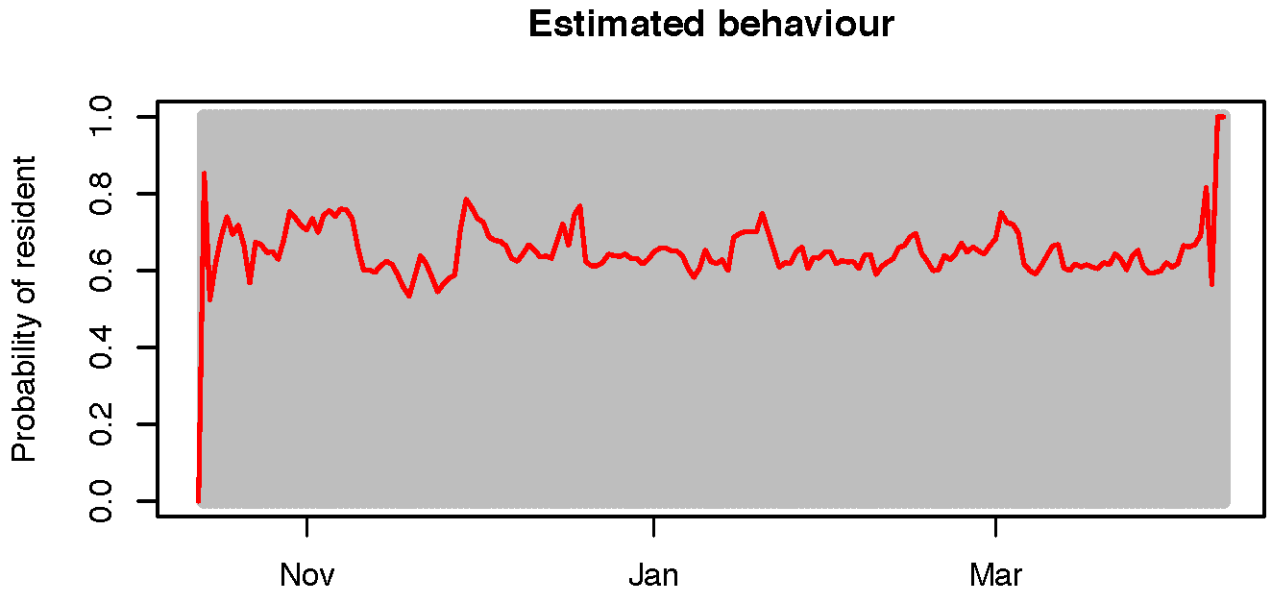
Once the parameter estimation routine(s) are finished, finalize those for input to the HMM filter/smoother. This primarily involves 1) converting movement parameters into kernels for convolution (K1/K2) and 2) converting state switching probabilities into a transition matrix (P). Once those steps are complete, everything is in place for the filter and smoother steps that generate the posterior state distributions. From that, we calculate most probable tracks by, in this case, finding the max of each posterior distribution. Other methods could be used here and may be implemented in the future (*e.g.* Viterbi).

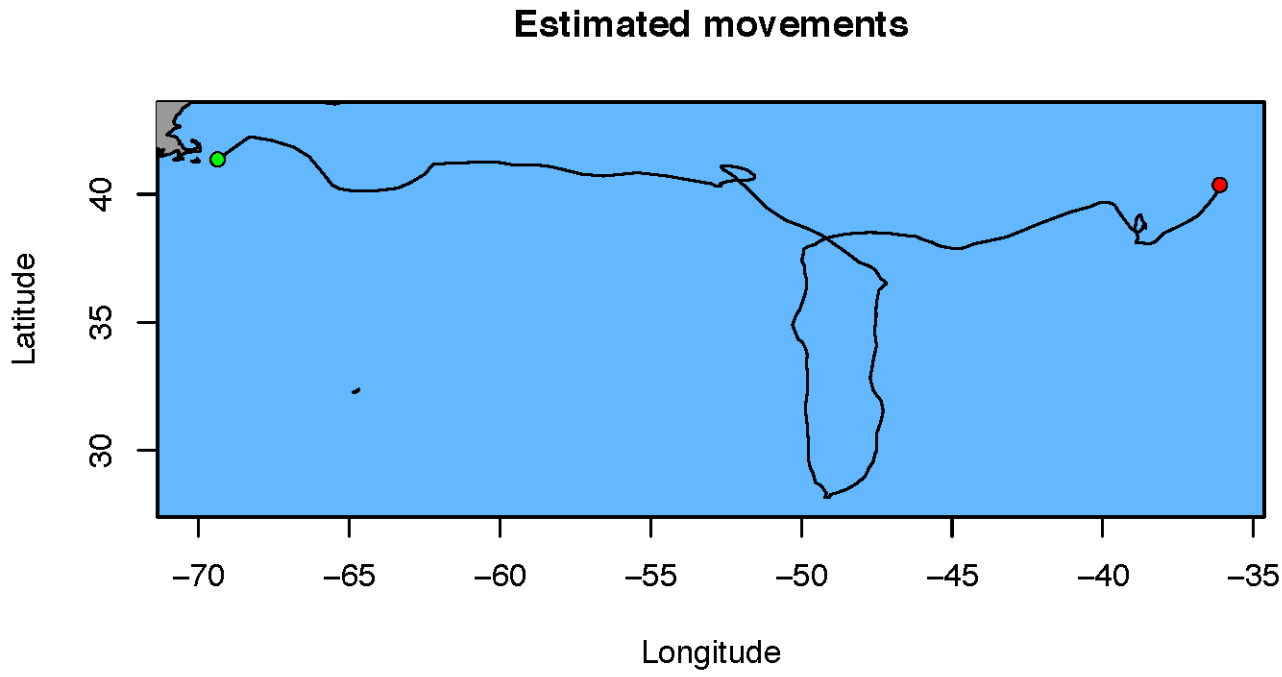
```
## as an example, grab the pars from the GA using both states pars <-
## pars.ga$par

## or use values from a previous run
pars <- c(4.964, 0.217, 0.367, 0.484)

## if only one state is used in the estimation routines (above), this will
## catch that here and set the parameters accordingly
if (length(pars) == 4) {
    sigmas = pars[1:2]
    sizes = rep(ceiling(sigmas[1] * 4), 2)
    pb = pars[3:4]
    muadvs = c(0, 0)
} else if (length(pars) == 1) {
    sigmas = pars[1]
    sizes = rep(ceiling(sigmas[1] * 4), 2)
    pb = NULL
    muadvs = c(0)
}

K1 <- gausskern.pg(sizes[1], sigmas[1], muadv = muadvs[1])
if (!is.null(pb)) K2 <- gausskern.pg(sizes[2], sigmas[2], muadv = muadvs[2])


## set transition matrix, if applicable
if (!is.null(pb)) {
    P <- matrix(c(pb[1], 1 - pb[1], 1 - pb[2], pb[2]), 2, 2, byrow = TRUE)
} else {
```

```
    P <- NULL
}

# RUN THE FILTER STEP
if (!is.null(pb)) {
    K <- list(K1, K2)
    f <- hmm.filter(g = L.res$g, L = L, K = K, P = P, m = 2)
} else {
    K <- list(K1)
    f <- hmm.filter(g = L.res$g, L = L, K = K, P = P, m = 1)
}
nllf <- -sum(log(f$psi[f$psi > 0]))  # negative log-likelihood
AIC <- 2 * nllf + 2 * length(which(!is.na(pars)))

# RUN THE SMOOTHING STEP
s <- hmm.smoother(f, K = K, L = L, P = P)

# GET THE MOST PROBABLE TRACK
tr <- calc.track(s, g = L.res$g, dateVec, iniloc)
```

3.4.3 Plotting results

```
plotHMM(s, tr, dateVec, ptt = "141259_example", save.plot = F)

plotRD(s, tr, ptt = "141259_example", g = g, xlims = c(sp.lim$xmin, sp.lim$xmax),
    ylims = c(sp.lim$ymin, sp.lim$ymax), makePlot = TRUE, save.plot = FALSE)
```





3.4.4 Comparing models and selection

Info here on NLL/AIC and model scoring

3.5 Resources

While the above is a nice simple introduction to `HMMoce` using 1 example tag dataset, rarely are we running only 1 model and for just a single individual. Often we want to iterate through a bunch of different potential likelihood combinations and parameters (*i.e.* 1 state vs 2). Comparing and selecting models is covered above, but some "real world" example run scripts may also be helpful to show how we might tackle applying this model framework to multiple datasets for a given study, for example.

Here's an example run script for a set of bigeye tuna tagged with Microwave X-tags: https://github.com/camrinbraun/HMMoce_run/blob/master/run_GoletBET.r

Here's an example run script for a handful of porbeagle tagged with PSATs, some of which were physically recovered: https://github.com/camrinbraun/HMMoce_run/blob/master/run_porbeagle_archival.r

Note that the Github repo (https://github.com/camrinbraun/HMMoce_run) where a lot of these run scripts live is messy and contains many outdated (likely disfunctional) scripts. The best resources will be the newest or most recently modified such as those linked above.

References

Braun, C. D., Galuardi, B., and Thorrold, S. R. (2018a). HMMoce: An R package for improved geolocation of archival-tagged fishes using a hidden Markov method. *Methods in Ecology and Evolution*, 9:1212–1220.

Braun, C. D., Gaube, P., Afonso, P., Fontes, J., Skomal, G. B., and Thorrold, S. R. (2019). Assimilating electronic tagging, oceanographic modelling, and fisheries data to estimate movements and connectivity of swordfish in the North Atlantic. *ICES Journal of Marine Science*.

Braun, C. D., Skomal, G. B., and Thorrold, S. R. (2018b). Integrating archival tag data and a high-resolution oceanographic model to estimate basking shark (Cetorhinus maximus) movements in the western Atlantic. *Frontiers in Marine Science*, 5(25).

Le Bris, A., Fréchet, A., Galbraith, P. S., and Wroblewski, J. S. (2013). Evidence for alternative migratory behaviours in the northern Gulf of St Lawrence population of Atlantic cod (Gadus morhua L.). *ICES Journal of Marine Science: Journal du Conseil*, 70(4):793–804.

Luo, J., Ault, J. S., Shay, L. K., Hoolihan, J. P., Prince, E. D., Brown, C. a., and Rooker, J. R. (2015). Ocean Heat Content Reveals Secrets of Fish Migrations. *Plos One*, 10(10):e0141101.

Musyl, M. K., Domeier, M. L., Nasby-Lucas, N., Brill, R. W., McNaughton, L. M., Swimmer, J. Y., Lutcavage, M. S., Wilson, S. G., Galuardi, B., and Liddle, J. B. (2011). Performance of pop-up satellite archival tags. *Marine Ecology Progress Series*.

Neilson, J. D., Smith, S., Royer, F., Paul, S. D., Porter, J. M., and Lutcavage, M. (2009). Investigations of horizontal movements of Atlantic swordfish using pop-up satellite archival tags. In *Tagging and tracking of marine animals with electronic devices*, pages 145–159. Springer.

Skomal, G. B., Zeeman, S. I., Chisholm, J. H., Summers, E. L., Walsh, H. J., McMahon, K. W., and Thorrold, S. R. (2009). Transequatorial migrations by basking sharks in the western Atlantic Ocean. *Current Biology*, 19(12):1019–1022.