Ocean Superfarms: Finding the Best Spots to Grow Seafood

Vedika Shirtekar

11/28/25

Ocean Superfarms: Finding the Best Spots to Grow Seafood

Learning Outcomes

This assignment will reinforce key concepts in geospatial analysis by practicing the following:

- combining vector/raster data
- resampling raster data
- masking raster data
- map algebra

For this assignment, you are tasked with determining which Exclusive Economic Zones (EEZ) on the West Coast of the US are best suited to developing marine aquaculture for several species of oysters ***and*** a species of your choice. Suitable locations will be determined based on range of suitable sea surface temperature (SST) and depth values for the species.

Description

To make your workflow generalizable, you must create a function that has the following characteristics:

- **arguments:**
  - minimum and maximum sea surface temperature
  - minimum and maximum depth
  - species name

- **outputs:**
  - map of EEZ regions colored by amount of suitable area
    - species name should be included in the map's title

Part 1: Map of Suitable Locations for Oyster Aquaculture

Prepare data

To start, we need to load all necessary data and make sure it has the coordinate reference system.

- shapefile for the West Coast EEZ
- bathymetry raster
- SST rasters
  - combine SST rasters into a raster stack

```
# Import packages
library(here) # Load "here" to locate and reference files
library(tidyverse) # Load the tidyverse" for data cleaning
library(sf) # Load "sf" for GIS analysis
library(raster) # Load "raster" for accessing raster data types
library(ggplot2) # Load "ggplot2" for data visualization
library(tmap) # Load "tmap" for functions to create and layer maps
library(kableExtra) # Load "kableExtra" for table formatting
library(stars) # Load "stars" for integration with "sf"
library(raster)
library(terra)
```

```
# West Coast EEZ
eez <- vect(here::here("data", "wc_regions_clean.shp"))
```

```
# Bathymetry raster
depth <- rast(here::here("data", "depth.tif"))
```

```
# Create a list of the tiff files for SST
sst_years <- list.files(path="data/", # File path
                        pattern = "average_annual", # File names matching pattern
```

```
                          full.names = TRUE) # Reference entire file names matching
  # Stack all rasters (stack() used earlier)
  sst <- rast(sst_years)
  names(sst)
```

```
[1] "average_annual_sst_2008" "average_annual_sst_2009"
[3] "average_annual_sst_2010" "average_annual_sst_2011"
[5] "average_annual_sst_2012"
```

```
  #class(sst)
  #class(depth)
  #plot(sst)
```

Double check CRS

```
  # Create list of spatial objects
  spatial_objects <- list(eez, depth, sst)

  # Use eez's CRS as reference
  ref_crs <- st_crs(spatial_objects$eez)
```

```
  # Check and transform each tile with if/else statements
  if (st_crs(spatial_objects$depth) != ref_crs) {
    warning("depth CRS does not match.
            Transforming to match eez CRS.")
    spatial_objects$depth <- st_transform(spatial_objects$depth, ref_crs)
  } else {
    message("depth CRS already matches eez CRS.")
  }
```

```
depth CRS already matches eez CRS.
```

```
  # Check and transform each tile with if/else statements
  if (st_crs(spatial_objects$sst) != ref_crs) {
    warning("sst CRS does not match.
            Transforming to match eez CRS.")
    spatial_objects$sst <- st_transform(spatial_objects$sst, ref_crs)
  } else {
```

```
  message("sst CRS already matches eez CRS.")
}
```

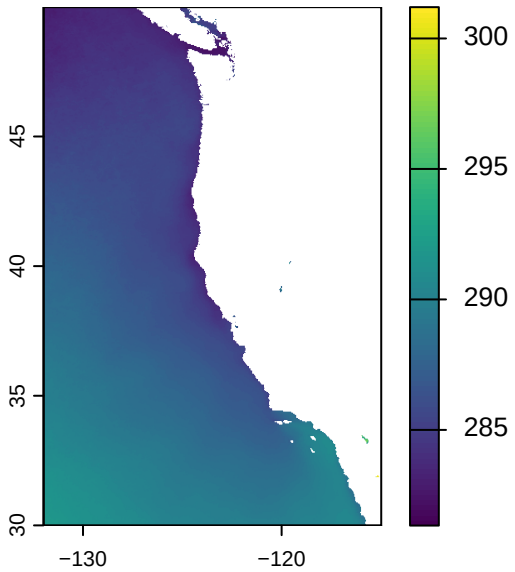
```
sst CRS already matches eez CRS.
```

Process data

Next, we need to process the SST and depth data so that they can be combined. In this case the SST and depth data have slightly different resolutions, extents, and positions.

- find the mean SST from 2008-2012 (e.g. create single raster of average SST)

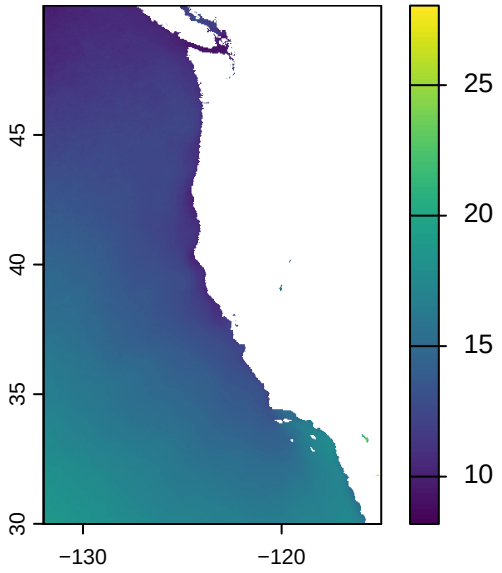
```
# Mean sst
#sst <- project(sst, depth)
depth <- project(depth, sst)

avg_sst <- mean(sst)
plot(avg_sst)
```



- convert average SST from Kelvin to Celsius

  - hint: subtract by 273.15

```
avg_sst <- avg_sst - 273.15
plot(avg_sst)
```



- crop depth raster to match the extent of the SST raster
- note: the resolutions of the SST and depth data do not match
  - resample the depth data to match the resolution of the SST data using the nearest neighbor approach

```
# Depth has slightly different resolutions, need to resample
?resample # Use raster version
```

```
Help on topic 'resample' was found in the following packages:

  Package               Library
  raster                /opt/R/4.2.2/lib/R/library
  terra                 /Users/vrs/R/x86_64-pc-linux-gnu-library/4.2
```

```
Using the first match ...
```

```
#depth_resampled <- resample(depth, sst, method = "near")
#plot(depth_resampled)
```

- check that the depth and SST match in resolution, extent, and coordinate reference system
  - hint: can the rasters be stacked?

```
# match crs
crs(avg_sst) == crs(depth) # match
```

```
[1] TRUE
```

```
#avg_sst <- project(avg_sst, depth)

depth_sst_crop <- crop(depth, avg_sst)

depth_sst_crop <- resample(depth_sst_crop, avg_sst, method = "near")
#depth_sst_crop <- project(depth_sst_crop, avg_sst)



# Check again...
crs(avg_sst) == crs(depth) # do not match
```

```
[1] TRUE
```

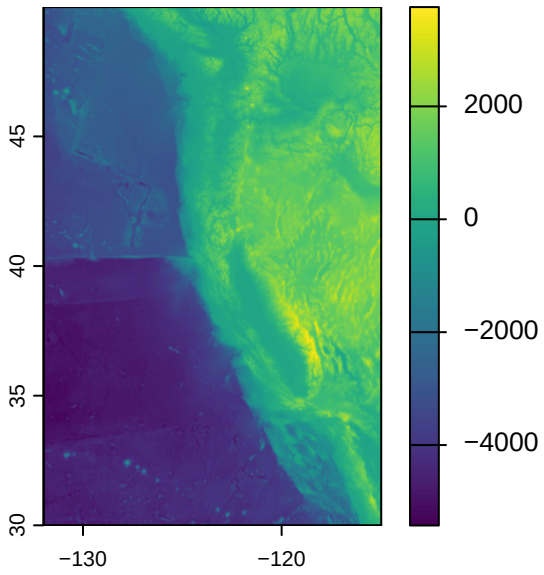
```
#crs(avg_sst) == crs(depth_sst) # do not match

# Don't match, so resample
#res(avg_sst) == res(depth)

# cannot be stacked since extents do not match... after resample, crop
c(depth_sst_crop, avg_sst)
```

```
class       : SpatRaster
size        : 480, 408, 2  (nrow, ncol, nlyr)
resolution  : 0.04166185, 0.04165702  (x, y)
extent      : -131.9848, -114.9867, 29.99305, 49.98842  (xmin, xmax, ymin, ymax)
coord. ref. : lon/lat WGS 84
source(s)   : memory
names       :     depth,     mean
min values  : -5424.830,  8.18400
max values  :  3759.372, 28.04978
```

```
# extent do not match... need to crop BUT after resampling
# crop first
ext(depth) == ext(depth_sst_crop)
```

```
[1] TRUE
```

```
# crop depth to sst extent
plot(depth_sst_crop)
```



```
plot(depth)
ext(depth_sst_crop) == ext(depth)
```

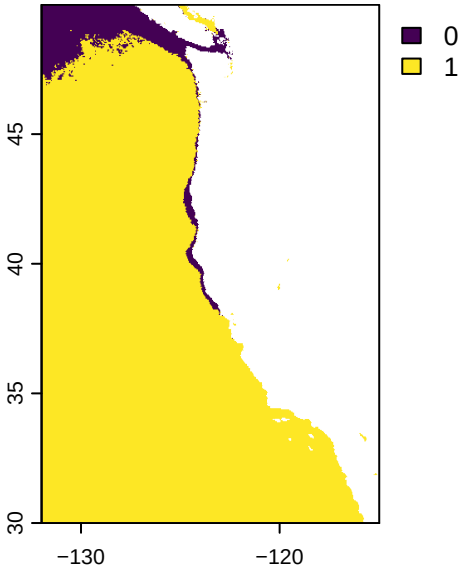
```
[1] TRUE
```

Find suitable locations

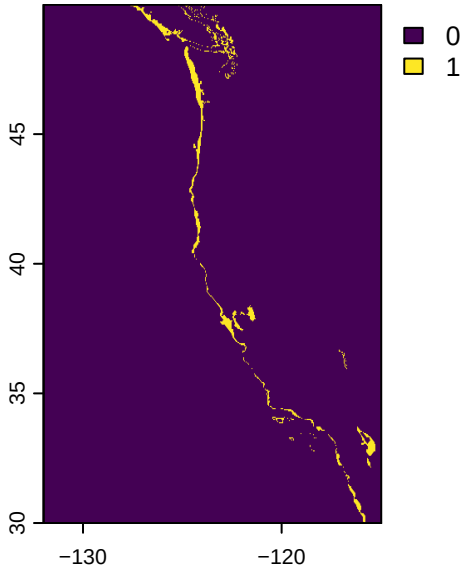
To find suitable locations for marine aquaculture, we'll need to find locations that are suitable in terms of both SST and depth.

- reclassify SST and depth data into locations that are suitable for oysters
  - hint: set suitable values to `1` and unsuitable values to `0`

```
# preferred range:
#sea surface temperature: 11-30°C

## reclassify avg_sst
# reclass matrix
reclass_matrix_sst <- matrix(
  c(-Inf, 11, 0,
    11, 30, 1,
    30, Inf, 0),
  ncol = 3,
  byrow = T
)

avg_sst_reclass <- classify(avg_sst, rcl = reclass_matrix_sst)

plot(avg_sst_reclass)
```




```
## Now, reclassify depth

#depth: 0-70 meters below sea level

# reclass matrix
reclass_matrix_depth <- matrix(
  c(-Inf, -70, 0,
    -70, 0, 1,
    0, Inf, 0),
  ncol = 3,
  byrow = T
)

depth_reclass <- classify(depth_sst_crop, rcl = reclass_matrix_depth)
plot(depth_reclass)
```




```
# make sure extent match
```

- find locations that satisfy both SST and depth conditions

```
# crop extent?
# crs(depth) == crs(avg_sst)
#
#
# ext(depth) == ext(avg_sst)

# can they be stacked?
c(depth_reclass, avg_sst_reclass)
```

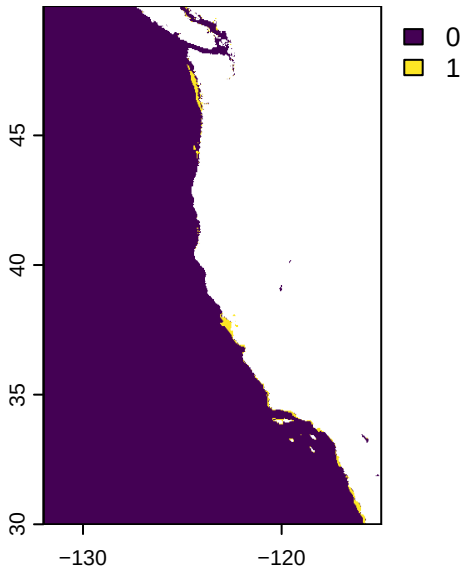
```
class       : SpatRaster
size        : 480, 408, 2  (nrow, ncol, nlyr)
resolution  : 0.04166185, 0.04165702  (x, y)
extent      : -131.9848, -114.9867, 29.99305, 49.98842  (xmin, xmax, ymin, ymax)
coord. ref. : lon/lat WGS 84
source(s)   : memory
names       : depth, mean
min values  :     0,    0
```

```
max values  :     1,    1
```

```
crs(depth_reclass) == crs(avg_sst_reclass)
```

```
[1] TRUE
```

```
# need to find BOTH locations... need to filter for suitable locations then join?
# suitable areas with raster multiplication
?lapp

#lapp(x = c(avg_sst_reclass,depth_reclass), fun = func(x,y) retunx*y)
multiply <- function(x,y){
  multi_raster <- x*y
  return(multi_raster)
  }

avg_sst_depth <- lapp(x = c(avg_sst_reclass,depth_reclass), fun = multiply)
plot(avg_sst_depth)
```



```
#tm_shape(avg_sst_depth) + tm_raster()
res(avg_sst_reclass) == res(depth_reclass)
```

```
[1] TRUE TRUE
```

Determine the most suitable EEZ

We want to determine the total suitable area within each EEZ in order to rank zones by priority. To do so, we need to find the total area of suitable locations within each EEZ.

- select suitable cells within West Coast EEZs
- find area of grid cells
- find the total suitable area within each EEZ
  - hint: it might be helpful to rasterize the EEZ data

```
## suitable cells within west coast eezs
# Note: eez is an sf.. convert to terra?
#eez_vect <- vect(eez)

#plot(eez)
crs(eez) == crs(avg_sst_depth) # not match, reproject
```

```
[1] FALSE
```

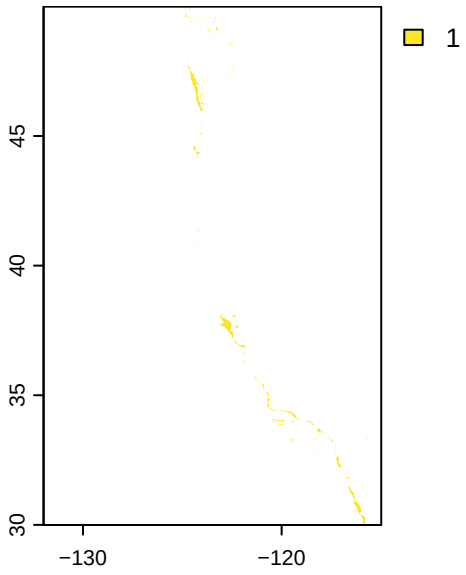
```
eez <- project(eez, avg_sst_depth)
crs(eez) == crs(avg_sst_depth) # now match, reproject
```

```
[1] TRUE
```

```
 # eez_rast <- rasterize(eez, avg_sst_depth_suitable, field = "rgn")
#
#
# suitable_cells_eez <-  mask(avg_sst_depth_suitable, eez)
#
#
```

```
#
# cellarea <- cellSize(suitable_cells_eez, unit = "km")
# suitablearea <- cellarea * suitable_cells_eez
#
#
# area_eez<- zonal(cellarea * suitable_cells_eez, eez_rast, fun = "sum", na.rm = T) %>%  r


#
# cellSize
# extract (sum)
# join to eez  (join by region ID)
```

```
# Find areas of interest
avg_sst_depth_suitable <- ifel(avg_sst_depth == 0, NA, 1)

plot(avg_sst_depth_suitable)
```




```
crs(eez) == crs(avg_sst_depth_suitable)
```

```
[1] TRUE
```

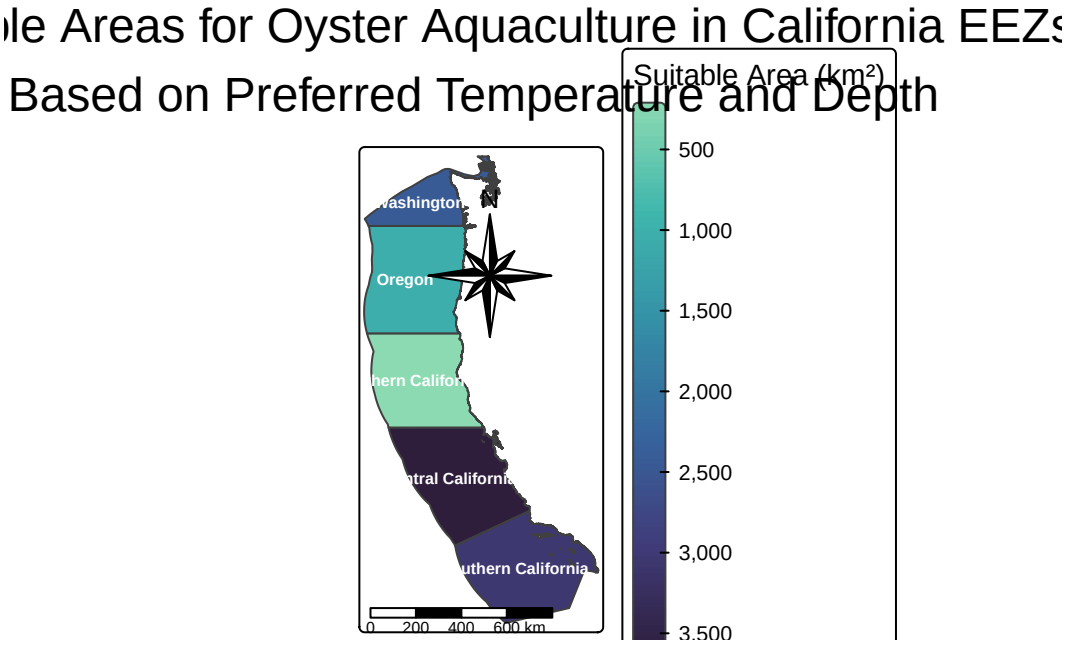
```
#plot(avg_sst_depth_suitable)
```

```
# rasterize eez regions
eez_rast <- rasterize(eez, avg_sst_depth_suitable, field = "rgn")

# identify suitable cells in mask
suitable_cells_eez <-  mask(avg_sst_depth_suitable, eez)

#calculate cell areas (m^2)
# cell_area <- cellSize(avg_sst_depth_suitable, unit = "km")
cell_area <- cellSize(suitable_cells_eez, unit = "km")

# mask areas by suitability (suitable areas within zones)
# keep area only for suitable cells
#suitable_area <- cell_area * avg_sst_depth_suitable
suitable_area <- cell_area * suitable_cells_eez

# sum suitable area in each eez polygon
# this will return data frame
#ea <- extract(suitable_area, eez_rast, fun = sum, na.rm = TRUE)
eez_sf <- st_as_sf(eez) # to have geometry

area_eez <- zonal(cell_area * suitable_cells_eez, eez_rast, fun = "sum", na.rm = T) %>%  r

area_eez <- area_eez %>% st_as_sf()


# # join extracted results back to eez
#eez_sf <- st_as_sf(eez) # to have geometry
#
# eez_joined <- eez_sf %>%
#   left_join(area_eez, by = "rgn")
#
# # convert back to spatvector
# eez_final <- vect(eez_joined)
```

```
# plot
# count for each state as label
```

```r
# basemap
tm_shape(area_eez) +
  tm_polygons(
    "suitable_area_km2",
    palette = "-mako", # reverse blue
    style = "cont", # continuous (styles referenced here: https://r-tmap.github.io/tmap-bo
    #fill.scale = tm_scale_continuous(values = "-suitable_area_km2"), # reverse scale
    title = "Suitable Area (km²)"
  ) +

  tm_text("rgn", size = 0.5, col = "white", fontface = "bold",  xmod = -.5) +

  tm_layout( # Center title outside bounding box
    main.title = "Suitable Areas for Oyster Aquaculture in California EEZs \n          Bas
     main.title.size = 1.5,
    legend.outside = TRUE, # Place legend outside map frame
    legend.outside.position = "right", # Place legend to right
    component.autoscale = FALSE, # Disable autoscaling for title
   outer.margins = c(0.01, 0.25, 0.01, 0.05)

    ) +
   tm_scale_bar( # Add scale bar for scale
     position = c(-.01, 0.08))  +
   # Move 1% from left and 8% from bottom
   tm_compass( # Add compass for orientation
     type = "8star",
     position = c("right", "top"))
```

```
-- tmap v3 code detected -------------------------------------------------------

[v3->v4] `tm_polygons()`: instead of `style = "cont"`, use fill.scale =
`tm_scale_continuous()`.
i Migrate the argument(s) 'palette' (rename to 'values') to
  'tm_scale_continuous(<HERE>)'
[v3->v4] `tm_polygons()`: migrate the argument(s) related to the legend of the
visual variable `fill` namely 'title' to 'fill.legend = tm_legend(<HERE>)'
[v3->v4] `tm_layout()`: use `tm_title()` instead of `tm_layout(main.title = )`
! `tm_scale_bar()` is deprecated. Please use `tm_scalebar()` instead.
```




```
# table with kableextra for zonal stats
# need: region and total area covered as well as how many places within each region
area_eez %>% st_as_sf() %>% st_drop_geometry %>%
  dplyr::select(region = rgn,
         suitable_area_km2,
         total_area_km2 = area_km2
         ) %>%   mutate(suitable_area_km2 = round(suitable_area_km2, 2),
                        percent_suitable = round((suitable_area_km2 / total_area_km2) * 10
kable(caption = "Amount of Suitable Areas by EEZ for Oyster Preferences") %>%
  kable_styling(bootstrap_options = c("striped", "hover"),
                full_width = FALSE,
                position = "center")
```

Table 1: Amount of Suitable Areas by EEZ for Oyster Preferences

region	suitable_area_km2	percent_suitable
Central California	3656.82	1.8
Northern California	194.13	0.1
Oregon	1028.90	0.6
Southern California	3062.20	1.5
Washington	2435.93	3.6

Part 2: Generalized Function of Aquaculture-Important Species Preferences

1. Need to clear environment
2. Define cells to have as context prior to function
3. Run function for regenerated map

```
# clear environment
#rm(list = ls())
```

```
# load in libraries again
library(here) # Load "here" to locate and reference files
library(tidyverse) # Load the tidyverse" for data cleaning
library(sf) # Load "sf" for GIS analysis
library(raster) # Load "raster" for accessing raster data types
library(ggplot2) # Load "ggplot2" for data visualization
library(tmap) # Load "tmap" for functions to create and layer maps
library(kableExtra) # Load "kableExtra" for table formatting
library(stars) # Load "stars" for integration with "sf"
#library(raster)
library(terra)
```

```
# Load in data again
```

```
# perform checks for matching CRS
```

```
# define multiply function for raster
multiply <- function(x,y){
  multi_raster <- x*y
  return(multi_raster)
  }
```

```
# This function takes arguments:
# minimum and maximum sea surface temperature
# minimum and maximum depth
# species name

species_preference <- function(min_temp, max_temp, min_depth, max_depth, species_name){

### # assume files have been loaded in already with matching crs checks
```

```r
###_____________________________________________________________
### data processing
# Mean sst
depth <- project(depth, sst)
avg_sst <- mean(sst)

avg_sst <- avg_sst - 273.15

# match crs
#crs(avg_sst) == crs(depth) # match


#avg_sst <- project(avg_sst, depth)

depth_sst_crop <- crop(depth, avg_sst)

depth_sst_crop <- resample(depth_sst_crop, avg_sst, method = "near")
#depth_sst_crop <- project(depth_sst_crop, avg_sst)

# preferred range:
#sea surface temperature: 11-30°C

## reclassify avg_sst
# reclass matrix
reclass_matrix_sst <- matrix(
  c(-Inf, min_temp, 0,
    min_temp, max_temp, 1,
    max_temp, Inf, 0),
  ncol = 3,
  byrow = T
)

avg_sst_reclass <- classify(avg_sst, rcl = reclass_matrix_sst)

# same for depth
reclass_matrix_depth <- matrix(
  c(-Inf, min_depth, 0,
    min_depth, max_depth, 1,
    max_depth, Inf, 0),
  ncol = 3,
  byrow = T
```

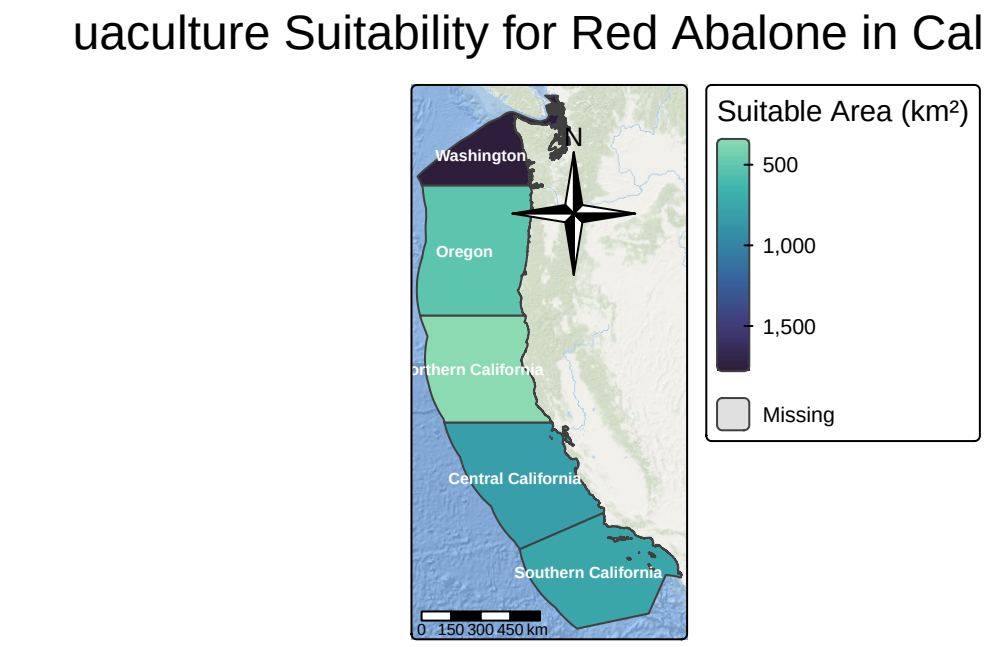
```
)

depth_reclass <- classify(depth_sst_crop, rcl = reclass_matrix_depth)

###_____________________________
# find locations that satisfy both SST and depth conditions
avg_sst_depth <- lapp(x = c(avg_sst_reclass,depth_reclass), fun = multiply)

## Part 4: find suitable locations within EEZ
#crs(eez) == crs(avg_sst_depth) # not match, reproject
eez <- project(eez, avg_sst_depth)

# Find areas of interest
avg_sst_depth_suitable <- ifel(avg_sst_depth == 0, NA, 1)


# rasterize eez regions
eez_rast <- rasterize(eez, avg_sst_depth_suitable, field = "rgn")

# identify suitable cells in mask
suitable_cells_eez <-  mask(avg_sst_depth_suitable, eez)

#calculate cell areas (m^2)
# cell_area <- cellSize(avg_sst_depth_suitable, unit = "km")
cell_area <- cellSize(suitable_cells_eez, unit = "km")

# mask areas by suitability (suitable areas within zones)
# keep area only for suitable cells
suitable_area <- cell_area * suitable_cells_eez

# sum suitable area in each eez polygon
# this will return data frame
#ea <- extract(suitable_area, eez_rast, fun = sum, na.rm = TRUE)
eez_sf <- st_as_sf(eez) # to have geometry

area_eez <- zonal(cell_area * suitable_cells_eez, eez_rast, fun = "sum", na.rm = T) %>%  r

area_eez <- area_eez %>% st_as_sf()


####### now create map
```

```r
# count for each state as label
# basemap
tm_shape(area_eez) +
  tm_polygons(
    "suitable_area_km2",
    palette = "-mako", # reverse blue
    style = "cont", # continuous (styles referenced here: https://r-tmap.github.io/tmap-bo
    #fill.scale = tm_scale_continuous(values = "-suitable_area_km2"), # reverse scale
    title = "Suitable Area (km²)"
  ) +

  tm_text("rgn", size = 0.5, col = "white", fontface = "bold",  xmod = -.5) +

  tm_layout( # Center title outside bounding box
    main.title = paste("Marine Aquaculture Suitability for", species_name, "in California
     main.title.size = 1.5,
    legend.outside = TRUE, # Place legend outside map frame
    legend.outside.position = "right", # Place legend to right
    component.autoscale = FALSE, # Disable autoscaling for title
   outer.margins = c(0.01, 0.25, 0.01, 0.05)

    ) +
   tm_scale_bar( # Add scale bar
     position = c(-.01, 0.08),    # Move 1% from left and 8% from bottom

     breaks = seq(0, 600, 150)) +

   tm_compass( # Add compass for orientation
     type = "4star",
     position = c("right", "top")) +
    tm_basemap("Esri.OceanBasemap")

  }


# outputs:
# map of EEZ regions colored by amount of suitable area
# species name should be included in the map's title
```

```r
# Call function
species_pref_map <- species_preference(min_temp = 8, max_temp = 18, min_depth = -25, max_d
```

```
-- tmap v3 code detected -------------------------------------------------

[v3->v4] `tm_polygons()`: instead of `style = "cont"`, use fill.scale =
`tm_scale_continuous()`.
i Migrate the argument(s) 'palette' (rename to 'values') to
  'tm_scale_continuous(<HERE>)'
[v3->v4] `tm_polygons()`: migrate the argument(s) related to the legend of the
visual variable `fill` namely 'title' to 'fill.legend = tm_legend(<HERE>)'
[v3->v4] `tm_layout()`: use `tm_title()` instead of `tm_layout(main.title = )`
! `tm_scale_bar()` is deprecated. Please use `tm_scalebar()` instead.
```

```
species_pref_map
```




```
tmap_save(species_pref_map, filename = "figs/species_pref_map.png", width= 8, height= 10)
```

```
Map saved to /Users/vrs/MEDS/eds-223/eds-223-hw/eds-223-hw-4/figs/species_pref_map.png
Resolution: 2400 by 3000 pixels
Size: 8 by 10 inches (300 dpi)
```

Marine Aquaculture Suitability for Red Abalone in California EEZs