Guidance Document

Seaside Chatters

2024-01-24

Table of contents

Preface 4

1 Introduction 5

I Tutorials 6

II Relevant Tutorials from Seaside Chats 7

2 Creating a Quarto Book 9
2.1 Creating a new Quarto Book & Repo . . . 9

3 Adapt NE SoE Indicator 11
3.1 Steps when first becoming acquainted . . . 11
3.1.1 1. View NE SoE report to determine indicator to replicate . . . 11
3.1.2 2. Go to GitHub site of the group that creates the NE SoE report to search for code . . . 11
3.1.3 3. Search for the "seasonal-sst-anom-gridded" code in the ecodata repository (still in the EDAB GitHub) . . . 15
3.1.4 4. Change Indicator to Gulf of Mexico . . . 21
3.1.5 5. Process New SST Anomaly Data for the GoM . . . 26
3.1.6 6. Final Indicator Plotting . . . 30
3.2 Steps now that we have more information . . . 33
3.2.1 1. Search for indicators in the technical document . . . 33
3.2.2 2. Replicate Indicator . . . 34
3.2.3 3. Check Data . . . 34
3.2.4 4. Plotting . . . 34

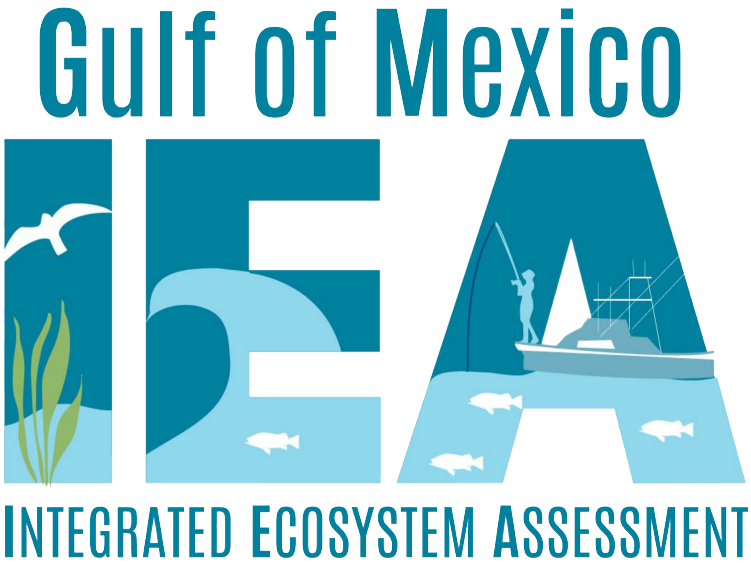
4 Intro to Spatial 35
4.1 Introduction . . . 35
4.2 Let's rasterize . . . 36
4.3 Customized bathymetric shapefile . . . 37
4.4 Random and uniform points . . . 40

III Second Part **42**

5 Products **43**

References **44**

Preface



This is the guidance document compiled by members of the Gulf of Mexico Integrated Ecosystem Assessment members to facilitate program missions and promote open science.

1 Introduction

This is a book created from markdown and executable code.

See Knuth (1984) for additional discussion of literate programming.

```
1 + 1
```

```
[1] 2
```

Part I

Tutorials

Part II

Relevant Tutorials from Seaside Chats

We can put relevant leading text here for a part if necessary.

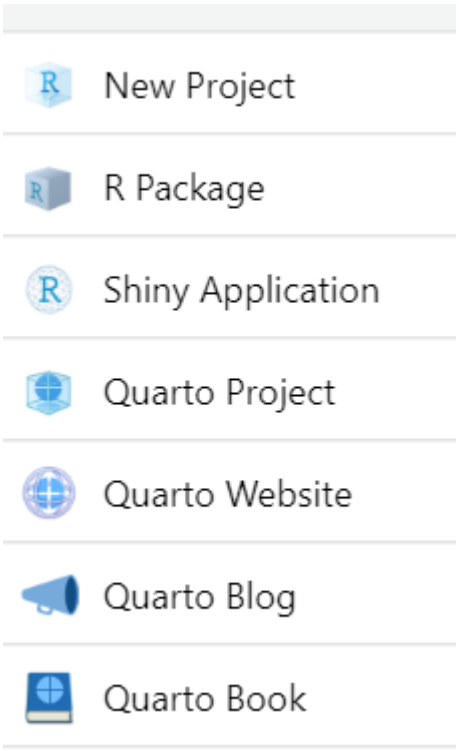
2 Creating a Quarto Book

2.1 Creating a new Quarto Book & Repo

There are several ways to do this which are well outlined in chapters 15, 16, and 17 of Happy Git and GitHub for the useR.

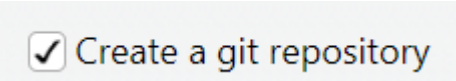
To be able to use the Quarto Book template when creating a new project, so that all the necessary files are included I followed these steps.

1. Create a New Project in RStudio
2. Choose "New Directory" to see a list of possible formats



3. Select "Quarto Book"

4. Enter the name of the project (will eventually become your repo name) and make sure it is in the correct directory.
5. Click "Create a git repository" before making the project.



6. Stage and commit the files the files that are part of the Quarto Book template (can alter later)
7. Create a repo on GitHub by running usethis::use_github() in the console
    - To add the quarto book to the Gulf IEA organisation I added the organisation="Gulf-IEA" argument
    - If you have already used RStudio with GitHub it will know your account and should know where to go
8. This create a new repo on GitHub as the origin and opens the repo in your browser

3 Adapt NE SoE Indicator

This is a tutorial providing guidance on how to adapt an indicator from the Northeast State of the Ecosystem report for use in GoM IEA products. This tutorial uses the "Seasonal SST Anomaly Indicator" and reproduces it for the Gulf of Mexico. Data are provided from the NOAA Optimal Interpolation SST High Resolution Dataset (here).

This tutorial is not an example of a "perfect process" but instead is an outline of my thought process of finding an indicator I would like to replicate, hunting down the code in the SoE repo, and the steps I had to complete for replication. I hope detailing this process will help others to adapt indicators/code from the SoE and other resources.

3.1 Steps when first becoming acquainted

3.1.1 1. View NE SoE report to determine indicator to replicate

NE State of the Ecosystem Report 2022

3.1.2 2. Go to GitHub site of the group that creates the NE SoE report to search for code

- Searched "GitHub NOAA NE SoE" and found the "Ecosystem Dynamics and Assessment Branch" GitHub (EDAB GitHub)
- Their GitHub hosts the repo "SOE-NEFMC" where the NE SoE report is housed
- Inside the "SOE-NEFMC" repo there are many files. I know that ".Rmd" and ".Qmd" are Rmardown and Quarto file extensions, respectively. These file types are used to create reports, so I knew that this was the code that was used to create the SoE report.
- I searched through the .Rmd document code using Ctrl+F from text in the report to find where the code for the indicator is.
- This gave me the name of the indicator throughout their code is "seasonal-sst-anom-gridded" and gave me more information for my search on the actual code to analyze and plot the data.

Ocean temperature and salinity Ocean temperatures continue to warm at both the surface (Fig. 23) and bottom (Fig. 24) throughout the Northeast Shelf including New England. Seasonal sea surface temperatures in 2021 were above average throughout the year, with some seasons rivaling or exceeding the record warm temperatures observed in 2012.

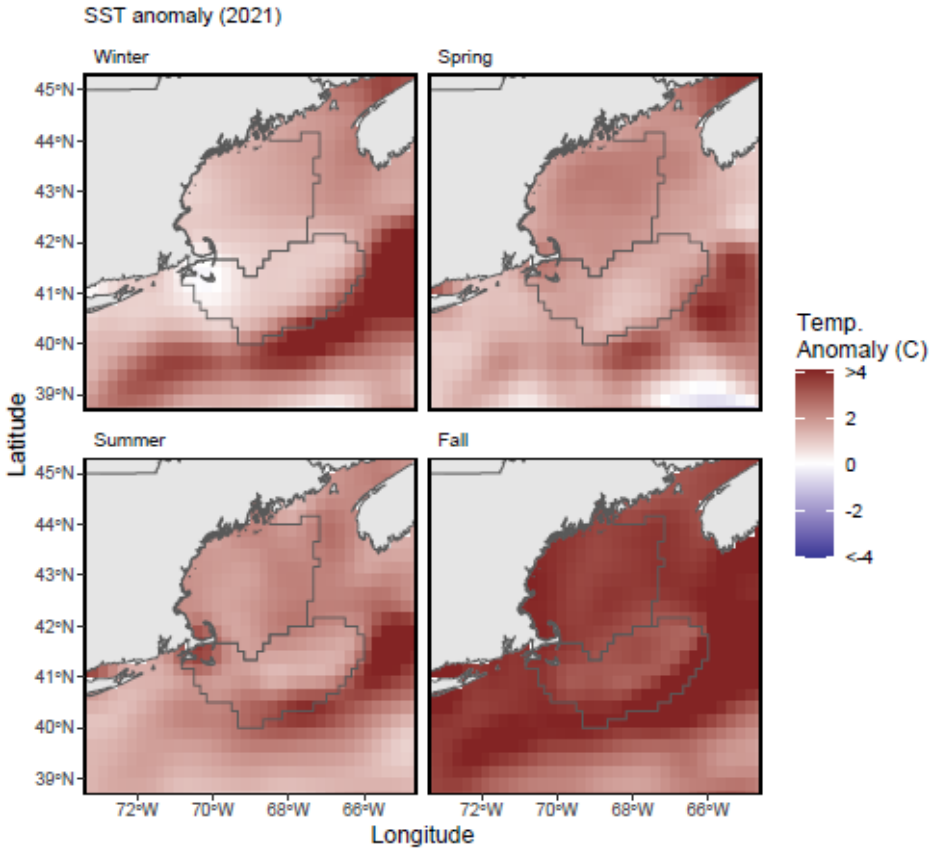


Figure 23: New England (EPUs outlined in grey) seasonal sea surface temperature (SST) time series overlaid onto 2021 seasonal spatial anomalies. Seasons are defined as: Jan-Mar for winter, Apr-Jun for spring, Jul-Sep for summer, and Oct-Dec for fall.

Figure 3.1: Indicator from the NE SoE 2022 report that I wished to duplicate for the GoM.

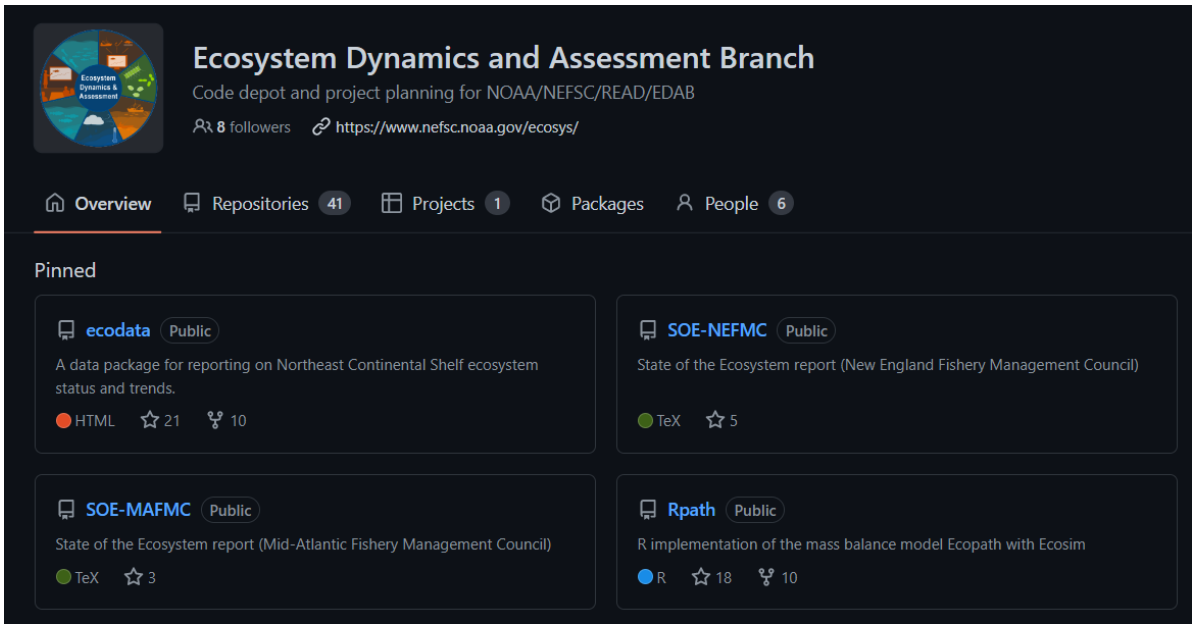


Figure 3.2: Screenshot of the EDAB Github. The group that creates the NE SoE Report.

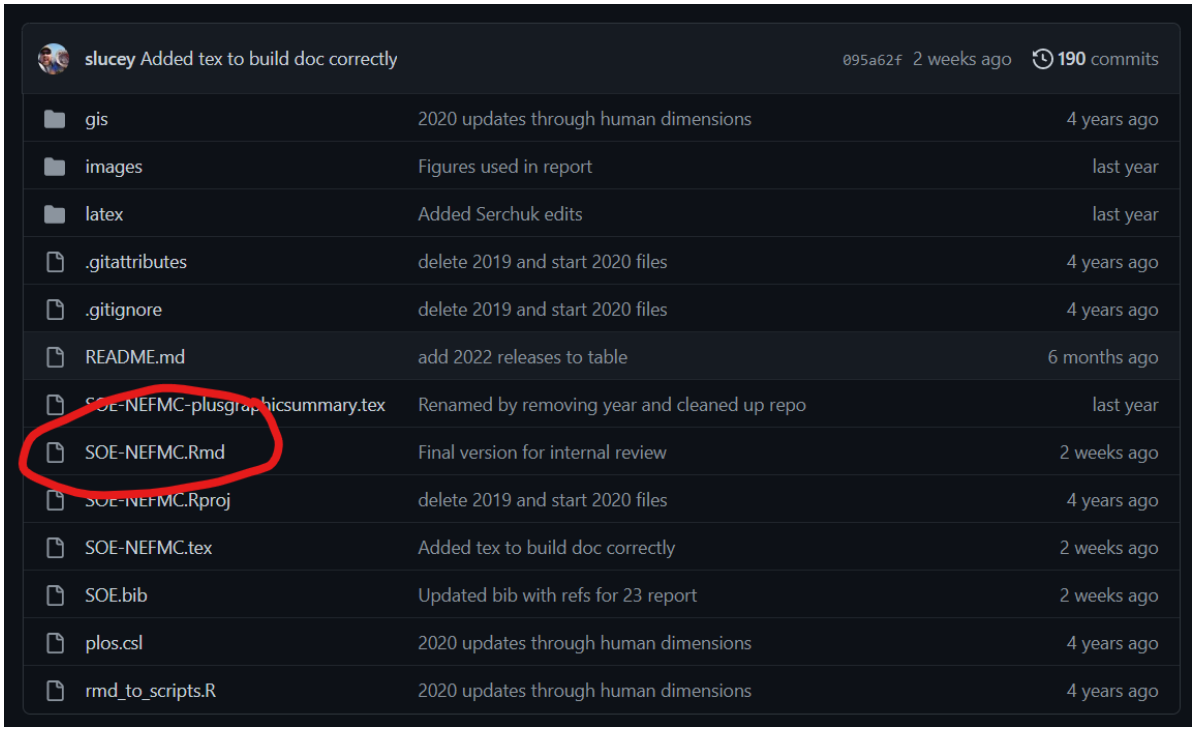
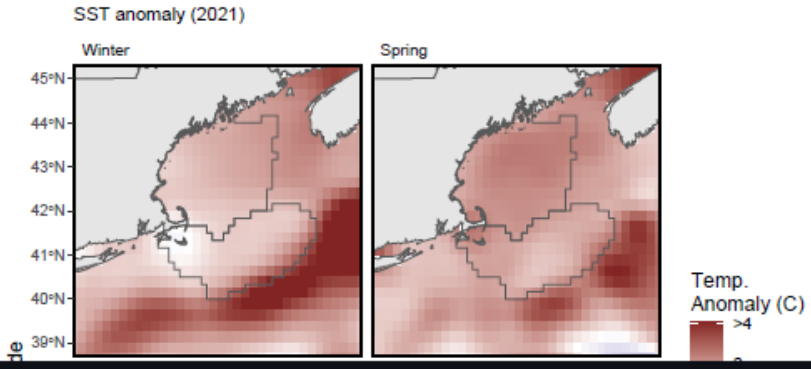


Figure 3.3: The .Rmd file that creates the NE SoE report.

Climate Change Indicators: ocean temperatures, heatwaves, currents, acidification

Ocean temperature and salinity Ocean temperatures continue to warm at both the surface (Fig. 23) and (Fig. 24) throughout the Northeast Shelf including New England. Seasonal sea surface temperatures in 2(above average throughout the year, with some seasons rivaling or exceeding the record warm temperatures (in 2012.




```
### Climate Change Indicators: ocean temperatures, extreme events, currents, acidification

#### Ocean temperature and salinity
The ocean continues to warm, alterning habitat conditions experienced by a wide range of species.  202

```{r seasonal-sst-anom-gridded, fig.cap="New England (EPUs outlined in grey) 2022 seasonal sea surfac
```
```

Figure 3.4: Top: Copied text from NE SoE Report. Bottom: Corresponding text in code.

3.1.3 3. Search for the "seasonal-sst-anom-gridded" code in the ecodata repository (still in the EDAB GitHub)

- We learned when Kim Bastille spoke to us that the "ecodata" package was created to house their data and the functions that sort, format, analyze, and plot their indicators.
- As seen earlier, the ecodata repo is on their GitHub and houses all that code.

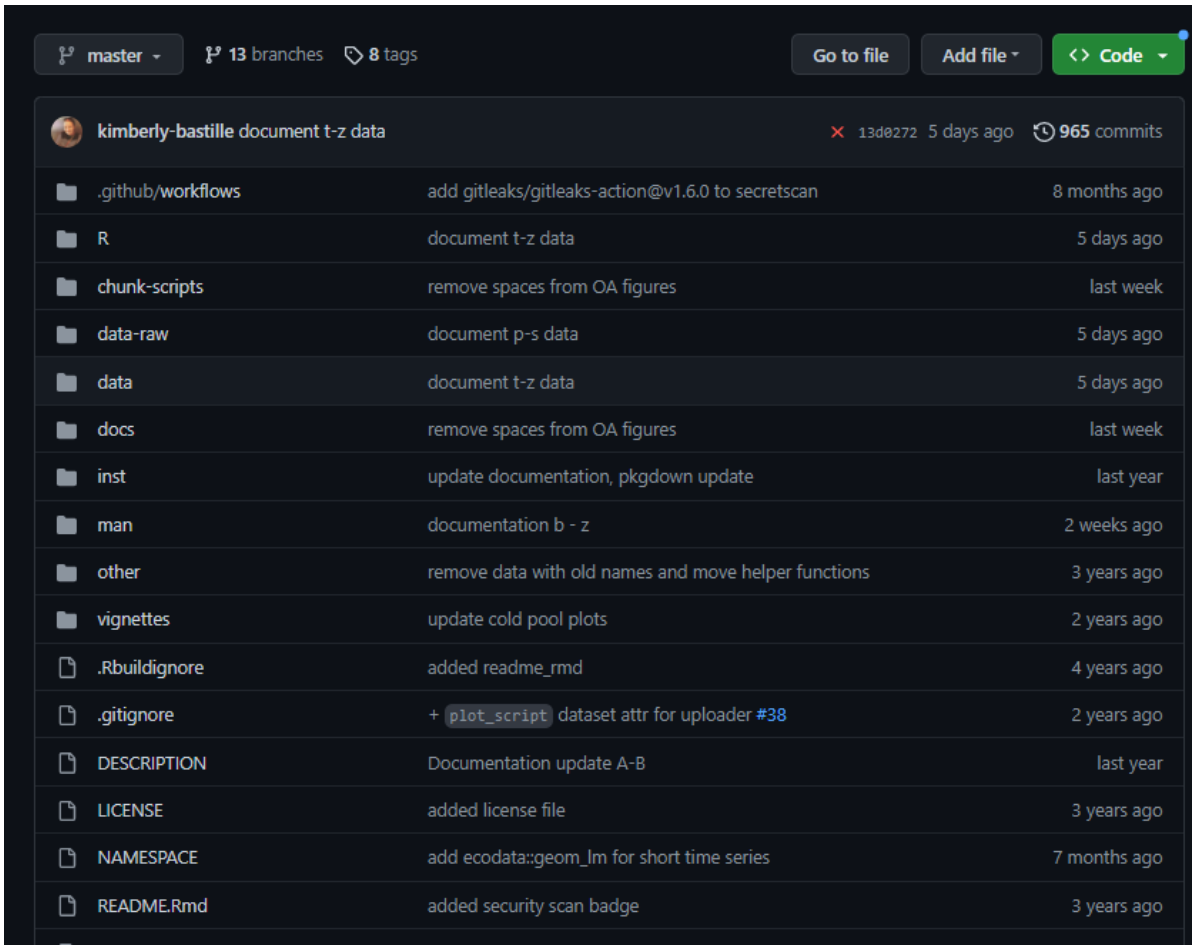


Figure 3.5: A screenshot of the ecodata repo and the choices of folders I had to search through.

- I first went to the "chunk-scripts" folder because Kim had mentioned that this is created using the 'readLines' function to take the script "chunks" for a specific indicator and puts them in this folder. In the folder I found a list of what appears to be all separate indicators and thought this would have all of the code I needed.
- I installed the ecodata package that contains the necessary data and functions to run the code that is in that "LTL_MAB.Rmd-seasonal-sst-anom-gridded.R" file.

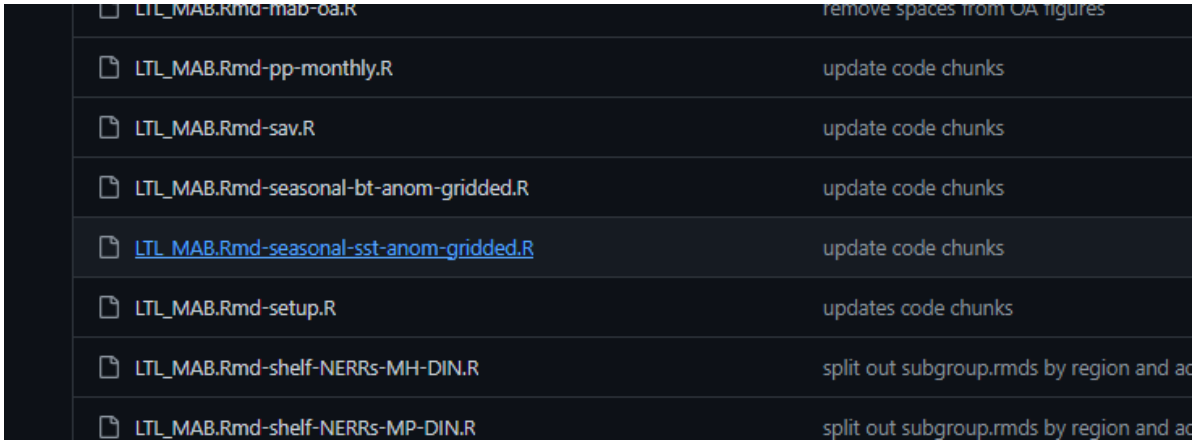


Figure 3.6: List of indicators in the "script-chunks" folder in the ecodata repo.

- However, ran I copy and pasted the code and ran it locally it would not work. I kept getting "Error: x.shade.min" not found. Letting me know that I was missing some piece of the puzzle.
- I continued to search through the ecodata repo to find other locations where the code for the "seasonal-sst-anom-gridded" indicator may be
- I found a docs folder that had LTL files which matched with the "LTL" in the indicator name of the chunks-script and looked there

Figure 3.7: The HTML document I looked at for more information on the indicator I was searching for.

- There is a .Rmd and .html file. The .Rmd file (RMarkdown) file creates the HTML file. I decided to first view the HTML file because it was easier to see the indicators, then used the .Rmd to view the code.

Previewing HTML files from GitHub

TIP: If you just click the .html file you will see the raw HTML code and not the website. However you can preview the rendered HTML website by pasting this text "http://htmlpreview.**github**.io/?" before the url to the LTL_MAB.html document.

- `Preview Text + .html URL = Website preview`
- `Preview Text = http://htmlpreview.github.io/?`
- `HTML Text = https://github.com/NOAA-EDAB/ecodata/blob/master/docs/LTL_MAB.html`
- `Preview Link = http://htmlpreview.github.io/?https://github.com/NOAA-EDAB/ecodata/blob/`

- This document once again shows the indicator that I was looking for, so I looked at the doc for any additional code that I may have been missing from my first try.
- To look at the code I looked at the .Rmd version of that document. Reminder the .Rmd holds the code to run the analysis and the .html holds the code to put it together in a document.
- When I was scrolling through the .Rmd document looking for the chunk addressing SST anomaly, I found code addressing the missing variable that was part of the previous error messages "x.shade.min". It was part of a setup code chunk in the beginning of the document which was not included in the previous code I explored.
- For this indicator I needed that code chunk so I copy/pasted this chunk before the previous code chunk and tried to recreate the indicator from the report.

Replicate First

- TIP: When adapting/manipulating code I recommend first reproducing it as is to make sure the code is working and then making changes to it as you need.

- Combining the setup code and the indicator code gave me this.

```
library(tidyverse)
library(ecodata)
library(sf)

### SETUP CODE FROM LTL_MAB.RmD
# Set lat/lon window for maps
xmin = -77
xmax = -65
ymin = 36
ymax = 45
```

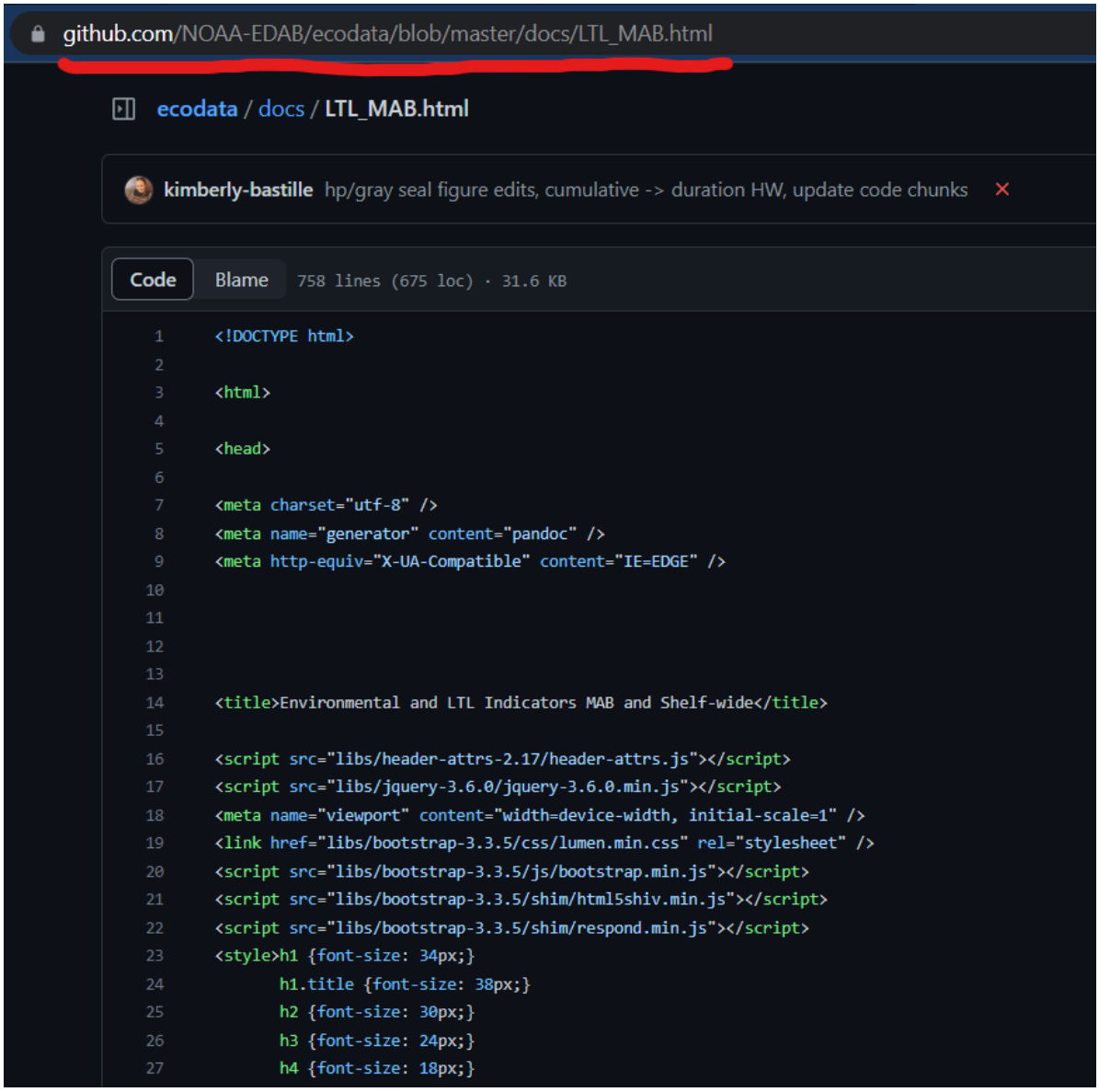
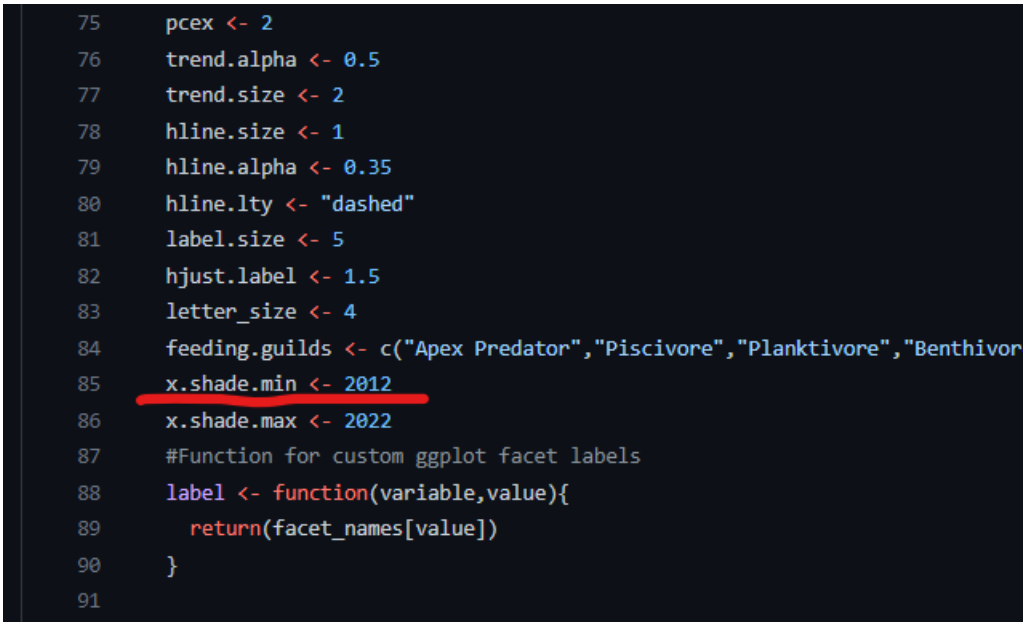


Figure 3.8: The raw HTML code after clicking on the "LTL_MAB.html" in the docs repo. The URL at the top is what you add onto the preview code mentioned in text.



```
75    pcex <- 2
76    trend.alpha <- 0.5
77    trend.size <- 2
78    hline.size <- 1
79    hline.alpha <- 0.35
80    hline.lty <- "dashed"
81    label.size <- 5
82    hjust.label <- 1.5
83    letter_size <- 4
84    feeding.guilds <- c("Apex Predator","Piscivore","Planktivore","Benthivor
85    x.shade.min <- 2012
86    x.shade.max <- 2022
87    #Function for custom ggplot facet labels
88    label <- function(variable,value){
89      return(facet_names[value])
90    }
91
```

Figure 3.9: Setup code chunk at the beginning of a document that helps define some variables for several indicators throughout the document.

```
xlims <- c(xmin, xmax)
ylims <- c(ymin, ymax)
#Time series constants
shade.alpha <- 0.3
shade.fill <- "lightgrey"
lwd <- 1
pcex <- 2
trend.alpha <- 0.5
trend.size <- 2
hline.size <- 1
hline.alpha <- 0.35
hline.lty <- "dashed"
label.size <- 5
hjust.label <- 1.5
letter_size <- 4
feeding.guilds <- c("Apex Predator","Piscivore","Planktivore","Benthivore","Benthos")
x.shade.min <- 2012
x.shade.max <- 2022
#Function for custom ggplot facet labels
label <- function(variable,value){
```

```
  return(facet_names[value])
}

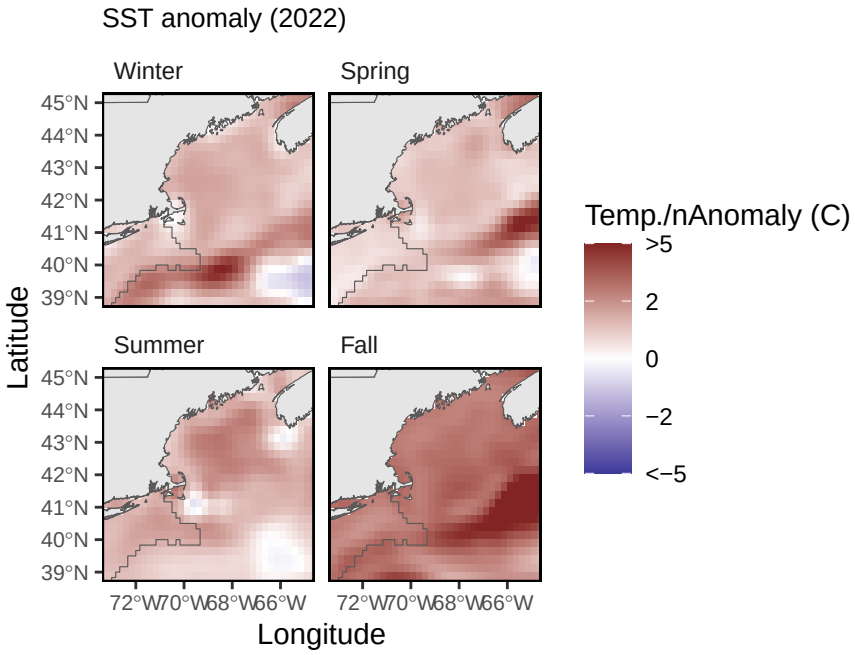
### ORIGINAL CODE FROM "LTL_MAB.Rmd-seasonal-sst-anom-gridded.R" IN SCRIPT-CHUNKS
#EPU shapefile
# ne_epu_sf <- ecodata::epu_sf %>%  dplyr::filter(EPU %in% c("GOM","GB"))
ne_epu_sf <- ecodata::epu_sf[4,4] #MAB SET

# View(ecodata::epu_sf)

#Map line parameters
map.lwd <- 0.4
# Set lat/lon window for maps
xmin = -73
xmax = -65
ymin = 39
ymax = 45
xlims <- c(xmin, xmax)
ylims <- c(ymin, ymax)
sst <- ecodata::seasonal_sst_anomaly_gridded  #WHERE YOU LOAD THE DATA FROM ECODATA
sst$Season <- factor(sst$Season, levels = c("Winter",
                                            "Spring",
                                            "Summer",
                                            "Fall"))
sst<- sst %>% dplyr::mutate(Value = replace(Value, Value > 5, 5))
sst_map <-
  ggplot2::ggplot() +
  ggplot2::geom_tile(data = sst, aes(x = Longitude, y = Latitude,fill = Value)) +
  ggplot2::geom_sf(data = ecodata::coast, size = map.lwd) +
  ggplot2::geom_sf(data = ne_epu_sf, fill = "transparent", size = map.lwd) +
  ggplot2::scale_fill_gradient2(name = "Temp./nAnomaly (C)",
                       low = scales::muted("blue"),
                       mid = "white",
                       high = scales::muted("red"),
                       limits = c(-5,5),
                       labels = c("<-5", "-2", "0", "2", ">5")) +
  ggplot2::coord_sf(xlim = xlims, ylim = ylims) +
  ggplot2::facet_wrap(Season~.) +
  ecodata::theme_map() +
  ggplot2::ggtitle("SST anomaly (2022)") +
  ggplot2::xlab("Longitude") +
```

```
  ggplot2::ylab("Latitude") +
  ggplot2::theme(panel.border = element_rect(colour = "black", fill=NA, size=0.75),
        legend.key = element_blank(),
        axis.title = element_text(size = 11),
        strip.background = element_blank(),
        strip.text=element_text(hjust=0),
        axis.text = element_text(size = 8),
        axis.title.y = element_text(angle = 90))+
  ecodata::theme_title()

sst_map
```



3.1.4 4. Change Indicator to Gulf of Mexico

- Now that I had working code I began examining the code for the pieces that change the geographic area for analyses and plotting to change that to the GoM.
- Of the above code, here are the lines I changed to accommodate to GoM

  The x and y ranges for figure plots (took the limits of the GoM shapefile I had)

```
# Set lat/lon window for maps
xmin = -98.05 #CHANGED
xmax = -80.43 #CHANGED
ymin = 17.40 #CHANGED
ymax = 31.46 #CHANGED
```

I substituted the shapefile that was provided in the indicator text with a GoM version of it. To do this I double checked that it was in the same format as the one used originally. For this use it needed to be the geometry of a polygon of class sf. To do this I used the function st_as_sf() from sf.

```
#EPU shapefile
# ne_epu_sf <- ecodata::epu_sf %>%  dplyr::filter(EPU %in% c("GOM","GB"))
# ne_epu_sf <- ecodata::epu_sf[4,4]


#OUR REGION
GoM_shp<-rgdal::readOGR("C:/Users/brittanytroast/Dropbox/Work/Seaside Chats/Tutorials
GoM_shp_sf<-sf::st_as_sf(GoM_shp)
GoM_shp_sf<-GoM_shp_sf$geometry
```

There was a second set of lat/lon ranges that I also changed. This just overrides the above code, but I changed it anyway.

```
# Set lat/lon window for maps
xmin = -98.05 #CHANGED
xmax = -80.43 #CHANGED
ymin = 17.40 #CHANGED
ymax = 31.46 #CHANGED
```

The last code I found addressing the geographic area of the indicator was in the ggplot code chunk where it calls on the shapefile. Since I have already renamed and loaded in a new shapefile I made sure to swap out the new name (or you could leave it as the same name and skip this step) in the plotting code.

```
  ggplot2::geom_sf(data = GoM_shp_sf, fill = "transparent", size = map.lwd)
```

- With these changes I reran the code hoping to have the new indicator with GoM SST anomalies.

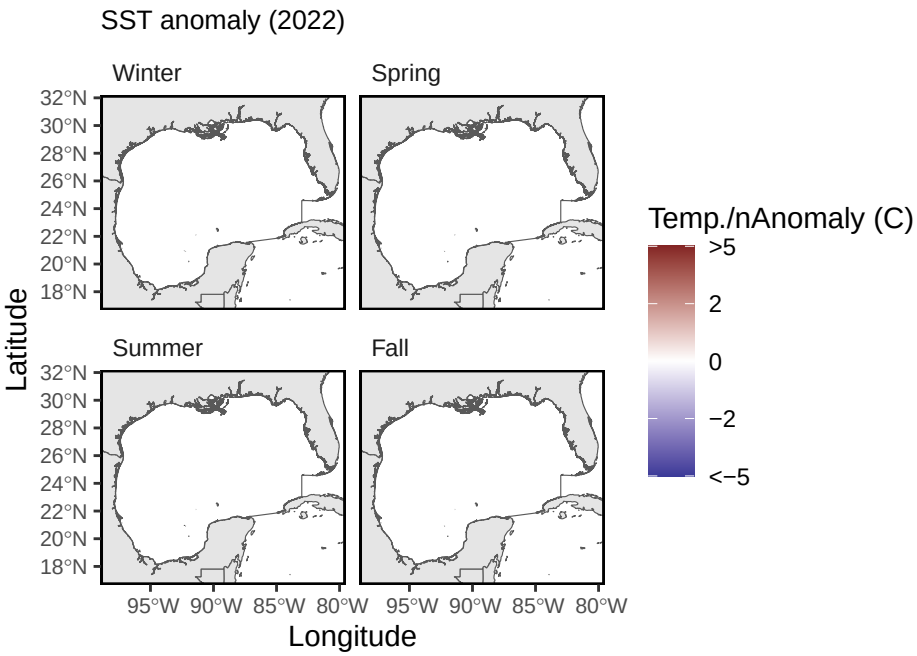
```r
# Set lat/lon window for maps
xmin = -98.05 #CHANGED
xmax = -80.43 #CHANGED
ymin = 17.40 #CHANGED
ymax = 31.46 #CHANGED
xlims <- c(xmin, xmax)
ylims <- c(ymin, ymax)
#Time series constants
shade.alpha <- 0.3
shade.fill <- "lightgrey"
lwd <- 1
pcex <- 2
trend.alpha <- 0.5
trend.size <- 2
hline.size <- 1
hline.alpha <- 0.35
hline.lty <- "dashed"
label.size <- 5
hjust.label <- 1.5
letter_size <- 4
feeding.guilds <- c("Apex Predator","Piscivore","Planktivore","Benthivore","Benthos")
x.shade.min <- 2012
x.shade.max <- 2022
#Function for custom ggplot facet labels
label <- function(variable,value){
  return(facet_names[value])
}


#OUR REGION
GoM_shp<-rgdal::readOGR("C:/Users/brittanytroast/Dropbox/Work/Seaside Chats/Tutorials
```

```
OGR data source with driver: ESRI Shapefile
Source: "C:\Users\brittanytroast\Dropbox\Work\Seaside Chats\Tutorials\Pics\Adapt NE SoE
with 1 features
It has 10 fields
```

```r
GoM_shp_sf<-sf::st_as_sf(GoM_shp)
shp_geom<-GoM_shp_sf$geometry

#Map line parameters
```

```r
map.lwd <- 0.4
# Set lat/lon window for maps
xmin = -98.05 #CHANGED
xmax = -80.43 #CHANGED
ymin = 17.40 #CHANGED
ymax = 31.46 #CHANGED
xlims <- c(xmin, xmax)
ylims <- c(ymin, ymax)

sst <- ecodata::seasonal_sst_anomaly_gridded  #WHERE YOU LOAD THE DATA FROM ECODATA
sst$Season <- factor(sst$Season, levels = c("Winter",
                                            "Spring",
                                            "Summer",
                                            "Fall"))

sst<- sst %>% dplyr::mutate(Value = replace(Value, Value > 5, 5))
sst_map <-
  ggplot2::ggplot() +
  ggplot2::geom_tile(data = sst, aes(x = Longitude, y = Latitude,fill = Value)) +
  ggplot2::geom_sf(data = ecodata::coast, size = map.lwd) +
  ggplot2::geom_sf(data = GoM_shp_sf, fill = "transparent", size = map.lwd) +
  ggplot2::scale_fill_gradient2(name = "Temp./nAnomaly (C)",
                       low = scales::muted("blue"),
                       mid = "white",
                       high = scales::muted("red"),
                       limits = c(-5,5),
                       labels = c("<-5", "-2", "0", "2", ">5")) +
  ggplot2::coord_sf(xlim = xlims, ylim = ylims) +
  ggplot2::facet_wrap(Season~.) +
  ecodata::theme_map() +
  ggplot2::ggtitle("SST anomaly (2022)") +
  ggplot2::xlab("Longitude") +
  ggplot2::ylab("Latitude") +
  ggplot2::theme(panel.border = element_rect(colour = "black", fill=NA, size=0.75),
        legend.key = element_blank(),
        axis.title = element_text(size = 11),
        strip.background = element_blank(),
        strip.text=element_text(hjust=0),
        axis.text = element_text(size = 8),
        axis.title.y = element_text(angle = 90))+
  ecodata::theme_title()
```

```
sst_map
```



- As you can see in the output, we are getting really close with figures of our study area, however, with a glaring lack of actual SST anomaly data.
- From here I took a further look at the SST Anomaly that data that came in the ecodata package and looked at the Lat/Long range to figure out what area it covers and realized that the package did not include the data coverage I needed.

```
head(ecodata::seasonal_sst_anomaly_gridded)
```

```
  Latitude Longitude      Value Season
1   45.875   -63.625 -0.2245555 Winter
2   45.875   -63.375 -0.1810536 Winter
3   45.875   -63.125 -0.1650498 Winter
4   45.875   -62.875 -0.1733716 Winter
5   45.875   -62.625 -0.2262375 Winter
6   45.875   -62.375 -0.2597778 Winter
```

```
range(ecodata::seasonal_sst_anomaly_gridded$Latitude)
```

```
[1] 35.125 45.875
```

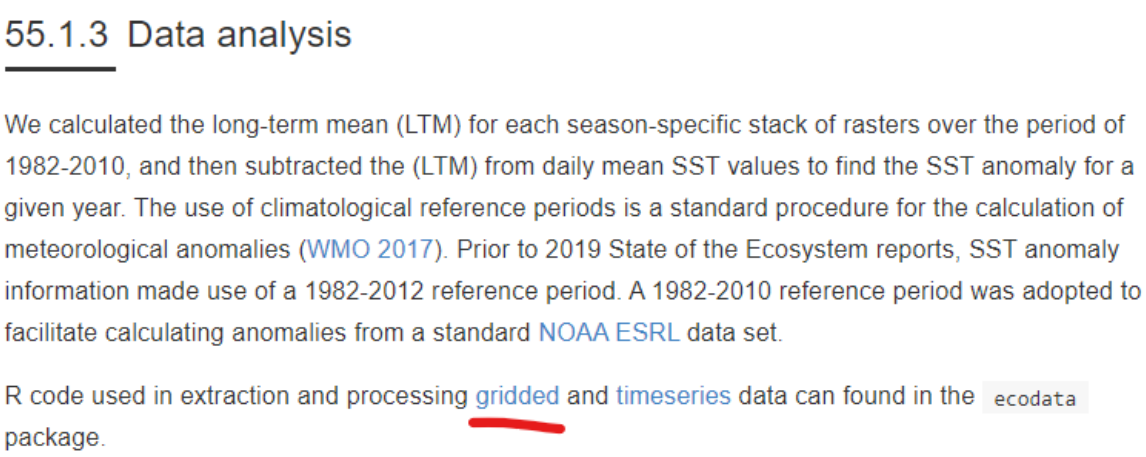
```
range(ecodata::seasonal_sst_anomaly_gridded$Longitude)
```

```
[1] -76.875 -60.125
```

- I then went on a hunt to figure out more about where this data came from and happened upon the NE SoE Technical document which describes every indicator and points of contact, details analysis, and gives data sources.

  NE SoE Technical Document

  Specific Link in Tech Doc to Indicator of Interest



55.1.3 Data analysis

We calculated the long-term mean (LTM) for each season-specific stack of rasters over the period of 1982-2010, and then subtracted the (LTM) from daily mean SST values to find the SST anomaly for a given year. The use of climatological reference periods is a standard procedure for the calculation of meteorological anomalies (WMO 2017). Prior to 2019 State of the Ecosystem reports, SST anomaly information made use of a 1982-2012 reference period. A 1982-2010 reference period was adopted to facilitate calculating anomalies from a standard NOAA ESRL data set.

R code used in extraction and processing gridded and timeseries data can found in the `ecodata` package.

Figure 3.10: Excerpt from NE SoE Technical Document on SST Anomaly Data extraction with links to code to do so.

- This document gave the data sources needed for the analysis and linked the code they used to analyze and format the data for use with the above code. (The gridded link in this section of the tech doc)

  Link to Code for processing the NOAA OISST SST Anomaly Data

3.1.5 5. Process New SST Anomaly Data for the GoM

- Now we just need to follow the steps in the code to get and process the same data for our specified region.
- When examining the code the first thing I noticed was the code saying "here" which linked this code to the ecodata repo

- To get around this I forked the ecodata repo to my computer so that it could access it however this is not necessary
- Upon reviewing the code I realized this was just providing a file directory and since we are providing our own directory we can just ignore those two lines of code (comment them out with a # so that they do not run)

```
ltm.dir <- here::here("data-raw/gridded/sst")
raw.dir <- here::here("data-raw")
crs <- "+proj=longlat +lat_1=35 +lat_2=45 +lat_0=40
```

Figure 3.11: Code showing the here::here code

- In the code there is a chunk warning that you must separately download two data files (with link provided) and alter the code to the directory where you have saved the files.

```
seasonal_sst_anomaly_gridded_day_nc <-"sst.day.mean.2022.nc"
seasonal_sst_anomaly_gridded_ltm_nc <- "sst.day.mean.ltm.1982-2010.nc"
#These data are large files that are not included among ecodata source files. They are accessible
#here: https://www.esrl.noaa.gov/psd/data/gridded/data.noaa.oisst.v2.highres.html
# but are removed after use as they are too large to store on github
sst.2019 <- rast_prep(stack(file.path(ltm.dir, seasonal_sst_anomaly_gridded_day_nc)))
ltm <- rast_prep(stack(file.path(ltm.dir,"ltm/", seasonal_sst_anomaly_gridded_ltm_nc)))
```

Figure 3.12: Portion of code detailing that you need to locally download the SST data since it is too large to host in the ecodata repo.

Link to NOAA OI SST Anomaly data source

- Download the two files as instructed in code
- DOUBLE CHECK that your filepath matches where you have it stored before continuing to next chunk of code. Remove the ltm.dir with your personal file path

- I double checked the rest of the code for any spatial things I may need to change and there was none. There was a chunk of code at the end that addressed heatwaves that I did not need and did not run.
- Running the code produced a dataset of SST anomalies for the GoM in the correct format for using the SoE indicator code.
  - I have provided the code that I used for this data with the same disclaimer that you must download the data separately.
  - Places with ALL CAPS comments are what I changed. They are in the extent, naming the file.dir argument, adding file.dir in the file.path() function, and "commenting out" unnecessary code at the end of the script

```
#Processing for spatial SST anomaly

library(dplyr)
library(raster)
library(sf)
library(ggplot2)
library(ncdf4)
library(reshape2)

rast_prep <- function(r){
  r <- rotate(r) #Rotate
  r <- crop(r, extent(-98.05,-80.43,17.40,31.46)) #Crop (CHANGED FOR GOM)
  return(r)
}


crs <- "+proj=longlat +lat_1=35 +lat_2=45 +lat_0=40+lon_0=-77 +x_0=0 +y_0=0 +datum=NA

seasonal_sst_anomaly_gridded_day_nc <-"sst.day.mean.2022.nc"
seasonal_sst_anomaly_gridded_ltm_nc <- "sst.day.mean.ltm.1982-2010.nc"

#These data are large files that are not included among ecodata source files. They ar
#here: https://www.esrl.noaa.gov/psd/data/gridded/data.noaa.oisst.v2.highres.html
# but are removed after use as they are too large to store on github

file.dir<-"C:/Users/brittanytroast/Documents/GitHub/Testing/Testing/SST Data" #ADDED

sst.2019 <- rast_prep(stack(file.path(file.dir, seasonal_sst_anomaly_gridded_day_nc))
ltm <- rast_prep(stack(file.path(file.dir, seasonal_sst_anomaly_gridded_ltm_nc))) #CH

winter.ltm <- ltm[[1:90]]
spring.ltm <- ltm[[91:181]]
summer.ltm <- ltm[[182:273]]
fall.ltm <- ltm[[274:365]]

winter.anom <- sst.2019[[1:90]] - winter.ltm
spring.anom <- sst.2019[[91:181]] - spring.ltm
summer.anom <- sst.2019[[182:273]] - summer.ltm
fall.anom <- sst.2019[[274:342]] - fall.ltm ### switched to cover just dates included
```

```r
rast_process <- function(r, season){
  r <- raster::stackApply(r, indices = rep(1,nlayers(r)),mean) #Find mean anomaly
  crs(r) <- crs #Add SOE CRS
  ### Remove smoothing steps due to "over smoothing'
  #r <- disaggregate(r, 5) #interpolate step 1 - create higher res grid
  #r <- focal(r, w=matrix(1,nrow=5,ncol=5), fun=mean,
  #           na.rm=TRUE, pad=TRUE) #interpolate step 2 - moving window
  r <- as(r, "SpatialPointsDataFrame") #Convert to ggplot-able object
  r <- as.data.frame(r)
  r <- r %>%
    reshape2::melt(id = c("y","x")) %>%
    dplyr::rename(Latitude = y, Longitude = x) %>%
    dplyr::select(-variable) %>%
    dplyr::mutate(Season = season) %>%
    dplyr::rename(Value = value)

  return(r)
}

#CHANGED NAME TO _GOM SO IT WASNT THE SAME AS BEFORE
seasonal_sst_anomaly_gridded_GoM <-
  rbind(rast_process(winter.anom,season = "Winter"),
      rast_process(spring.anom,season = "Spring"),
      rast_process(summer.anom, season = "Summer"),
      rast_process(fall.anom, season = "Fall")) %>%
  tibble::as_tibble()

#Don't need... here to write metadata
# # metadata ----
# attr(seasonal_sst_anomaly_gridded, "tech-doc_url") <- "https://noaa-edab.github.io/
# attr(seasonal_sst_anomaly_gridded, "data_files")   <- list(
#   seasonal_sst_anomaly_gridded_day_nc = seasonal_sst_anomaly_gridded_day_nc,
#   seasonal_sst_anomaly_gridded_ltm_nc = seasonal_sst_anomaly_gridded_ltm_nc)
# attr(seasonal_sst_anomaly_gridded, "data_steward") <- c(
#   "Kimberly Bastille <kimberly.bastille@noaa.gov>")
# attr(seasonal_sst_anomaly_gridded, "plot_script") <- list(
#   `ltl_MAB_shelf` = "LTL_MAB.Rmd-shelf-seasonal-sst-anomaly-gridded.R",
#   `ltl_NE` = "LTL_NE.Rmd-seasonal-sst-anomaly-gridded.R")
#
# usethis::use_data(seasonal_sst_anomaly_gridded, overwrite = T)
```

```
# #### Get Gridded Daily Max values for Marine Heatwaves
#
# mab_peak_hw <- sst.2019[[210]] - summer.ltm## 7/28/2020
# gb_peak_hw <- sst.2019[[227]] - summer.ltm## 8/14/2020
# gom_peak_hw <- sst.2019[[214]] - summer.ltm## 8/2/2020
#
# rast_process_epu <- function(r, epu){
#   r <- raster::stackApply(r, indices = rep(1,nlayers(r)),mean) #Find mean anomaly
#   crs(r) <- crs #Add SOE CRS
#   ### Remove smoothing steps due to "over smoothing'
#   #r <- disaggregate(r, 5) #interpolate step 1 - create higher res grid
#   #r <- focal(r, w=matrix(1,nrow=5,ncol=5), fun=mean,
#   #           na.rm=TRUE, pad=TRUE) #interpolate step 2 - moving window
#   r <- as(r, "SpatialPointsDataFrame") #Convert to ggplot-able object
#   r <- as.data.frame(r)
#   r <- r %>%
#     reshape2::melt(id = c("y","x")) %>%
#     dplyr::rename(Latitude = y, Longitude = x) %>%
#     dplyr::select(-variable) %>%
#     dplyr::mutate(EPU = epu) %>%
#     dplyr::rename(Value = value)
#
#   return(r)
# }
#
#
# heatwave_peak_date <-
#   rbind(rast_process_epu(mab_peak_hw,epu = "MAB"),
#         rast_process_epu(gb_peak_hw ,epu = "GB"),
#         rast_process_epu(gom_peak_hw, epu = "GOM"))
#
# usethis::use_data(heatwave_peak_date, overwrite = T)
```

3.1.6 6. Final Indicator Plotting

- Now that we have a new dataset for the GoM region all we have to do is run the previous code with the new data. I have saved the smaller GoM SST Anomaly dataset for convenience.

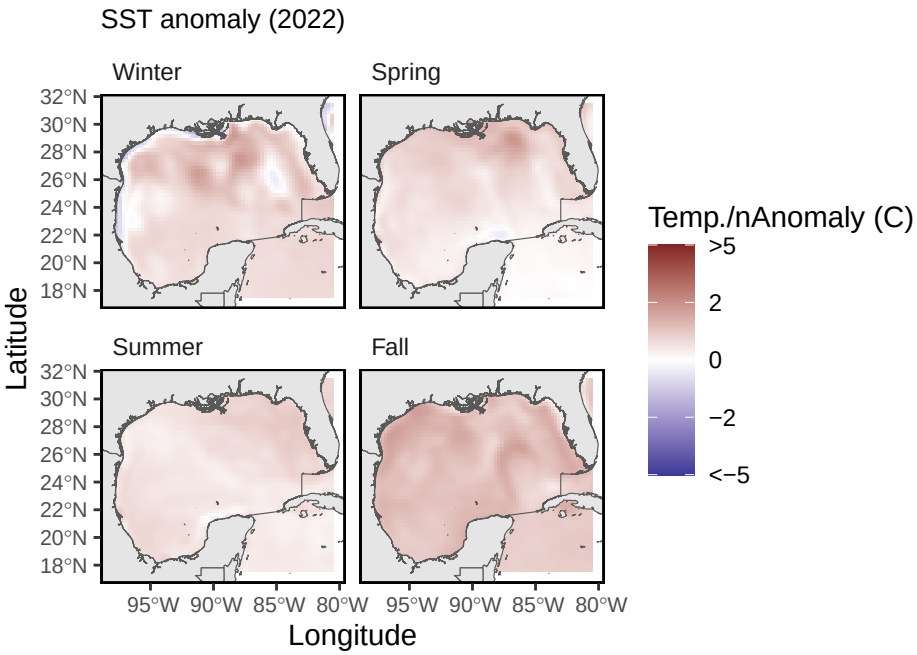
```
# Set lat/lon window for maps
xmin = -98.05 #CHANGED
xmax = -80.43 #CHANGED
ymin = 17.40 #CHANGED
ymax = 31.46 #CHANGED
xlims <- c(xmin, xmax)
ylims <- c(ymin, ymax)
#Time series constants
shade.alpha <- 0.3
shade.fill <- "lightgrey"
lwd <- 1
pcex <- 2
trend.alpha <- 0.5
trend.size <- 2
hline.size <- 1
hline.alpha <- 0.35
hline.lty <- "dashed"
label.size <- 5
hjust.label <- 1.5
letter_size <- 4
feeding.guilds <- c("Apex Predator","Piscivore","Planktivore","Benthivore","Benthos")
x.shade.min <- 2012
x.shade.max <- 2022
#Function for custom ggplot facet labels
label <- function(variable,value){
  return(facet_names[value])
}


#OUR REGION
GoM_shp<-rgdal::readOGR("C:/Users/brittanytroast/Dropbox/Work/Seaside Chats/Tutorials/Pics
```

```
OGR data source with driver: ESRI Shapefile
Source: "C:\Users\brittanytroast\Dropbox\Work\Seaside Chats\Tutorials\Pics\Adapt NE SoE Indi
with 1 features
It has 10 fields
```

```
GoM_shp_sf<-sf::st_as_sf(GoM_shp)
GoM_shp_sf<-GoM_shp_sf$geometry

#Map line parameters
```

```r
map.lwd <- 0.4
# Set lat/lon window for maps
xmin = -98.05 #CHANGED
xmax = -80.43 #CHANGED
ymin = 17.40 #CHANGED
ymax = 31.46 #CHANGED
xlims <- c(xmin, xmax)
ylims <- c(ymin, ymax)

# sst <- ecodata::seasonal_sst_anomaly_gridded  #WHERE YOU LOAD THE DATA FROM ECODATA
sst<-read.csv("C:/Users/brittanytroast/Dropbox/Work/Seaside Chats/Tutorials/Data/SST Anom

sst$Season <- factor(sst$Season, levels = c("Winter",
                                            "Spring",
                                            "Summer",
                                            "Fall"))

sst<- sst %>% dplyr::mutate(Value = replace(Value, Value > 5, 5))
sst_map <-
  ggplot2::ggplot() +
  ggplot2::geom_tile(data = sst, aes(x = Longitude, y = Latitude,fill = Value)) +
  ggplot2::geom_sf(data = ecodata::coast, size = map.lwd) +
  ggplot2::geom_sf(data = GoM_shp_sf, fill = "transparent", size = map.lwd) +
  ggplot2::scale_fill_gradient2(name = "Temp./nAnomaly (C)",
                       low = scales::muted("blue"),
                       mid = "white",
                       high = scales::muted("red"),
                       limits = c(-5,5),
                       labels = c("<-5", "-2", "0", "2", ">5")) +
  ggplot2::coord_sf(xlim = xlims, ylim = ylims) +
  ggplot2::facet_wrap(Season~.) +
  ecodata::theme_map() +
  ggplot2::ggtitle("SST anomaly (2022)") +
  ggplot2::xlab("Longitude") +
  ggplot2::ylab("Latitude") +
  ggplot2::theme(panel.border = element_rect(colour = "black", fill=NA, size=0.75),
        legend.key = element_blank(),
        axis.title = element_text(size = 11),
        strip.background = element_blank(),
        strip.text=element_text(hjust=0),
        axis.text = element_text(size = 8),
```

```
        axis.title.y = element_text(angle = 90))+
  ecodata::theme_title()

sst_map
```



- There you have it! The same indicator from the NE SoE report but adapted for the GoM.

3.2 Steps now that we have more information

After going through the previous process I am more informed and would use a different process if I wanted to adapt a new indicator. I will briefly describe these steps here.

3.2.1 1. Search for indicators in the technical document

- I would start my search for indicators in the technical document. They have many more indicators than the reports do, as all indicators are not published. Plus I will get a good overview of the methods and datasets used in analysis. The technical report also has example figures for reference.

NE SoE Technical Document

3.2.2 2. Replicate Indicator

- I would replicate the indicator using the data provided by the ecodata package and using the associated code in the "chunk-scripts" folder in the ecodata repository.
- If any argument was missing I would search for that argument in the repo (as I had to do previously with the setup code for x.min.shade)

3.2.3 3. Check Data

- Next, I would check the extent of the data provided in the ecodata package and determine if it will work for my study area.
- If the data do not cover my study area I would refer to the information provided by the technical data and follow links to the code that will grab and format the data for my preferred region.

3.2.4 4. Plotting

- With the new data I would triple check the code for the indicator and make sure there are not any geographic or other variables that need to be altered for my region.
- With these changes hopefully I get the desired indicator for my region!

4 Intro to Spatial

4.1 Introduction

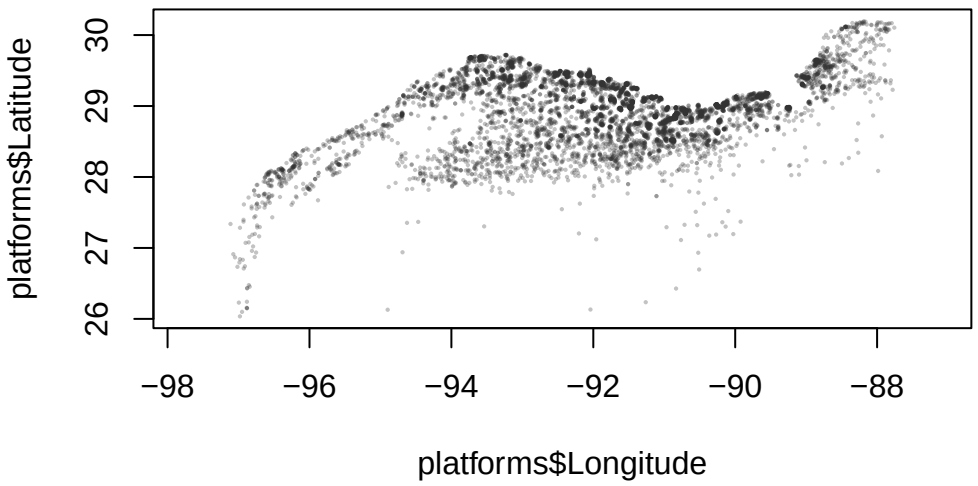
This is a quick tutorial on manipulating spatial data in R. I will focus on 4 valuable packages: leaflet, terra, sf, and spatstat.

First, some R housekeeping by loading needed libraries. I will be keeping this simple by using minimal packages (sorry tidyverse folks).

```
library(cmocean)
library(leaflet)
library(scales)
library(sf)
library(spatstat)
library(terra)
```

Next, lets import some data. I have been working with BOEM's oil rig dataset in the Gulf of Mexico. I removed rows that did not have lon/lat data and then turned it into an sf object using the st_as_sf function in the sf package.

```
platforms <- read.csv('PlatStruc2.csv')
platforms <- platforms[-which(is.na(platforms$Longitude)), ]
platforms_sf <- st_as_sf(platforms,
                         coords = c("Longitude", "Latitude"),
                         agr = "identity"
                         )

plot(platforms$Longitude,
     platforms$Latitude,
     pch = 16,
     cex = .3,
     col = alpha('gray20', .3),
     asp = 1)
```



4.2 Let's rasterize

A important skill is to take data in tabular format (like from a csv) and turn into a raster. This is pretty easy with the rast and rasterize functions in the terra package. First, I need to create an empty raster to supply the values to; however, you cane take an existing raster and create a new one based upon the existing ones spatial dimensions and resolution. For this example, I am essentially creating a 2D histogram by counting, or rather using the length function, the number of oil rigs per raster cell. Any function could be used in the rasterize function such as mean, median, or even a user supplied function. Note, I am using the platforms that I turned into an sf object to to this. Then I check the coordinate reference system. Note that the rast function automatically supply a CRS, you can change this if needed.

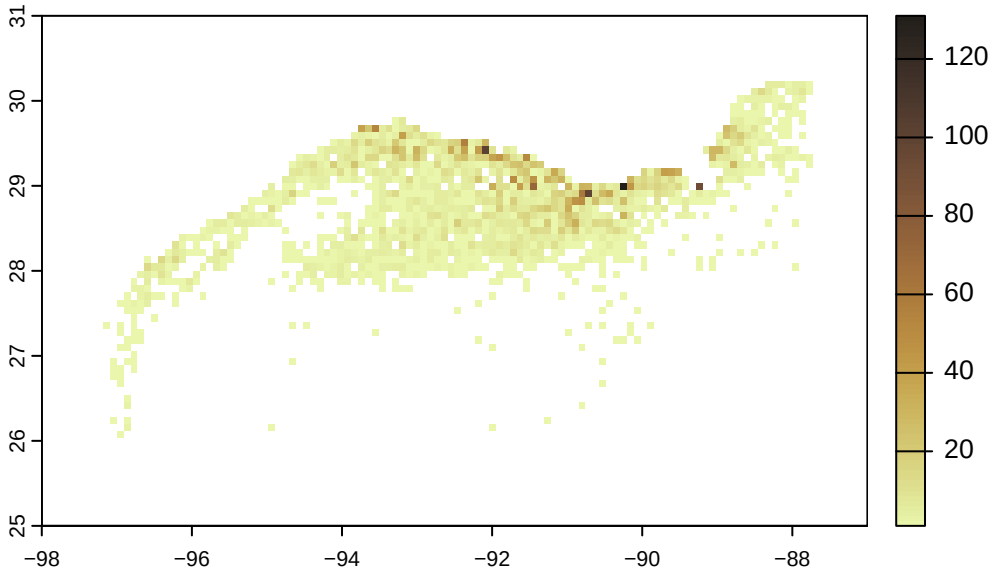
```
r <- rast(
  xmin = (-98), xmax = (-87),
  ymin = (25), ymax = (31),
  ncols = length(87:98) * 10,
  nrows = length(25:31) * 10,
  )
plat_rast <- rasterize(platforms_sf,
                       r,
                       "Area.Code",
```

```
                        fun = length
                        )
crs(plat_rast)
```

```
[1] "GEOGCRS[\"WGS 84 (CRS84)\",\n    DATUM[\"World Geodetic System 1984\",\n        ELLIPSOI
```

Always plot your results to see if they make sense.

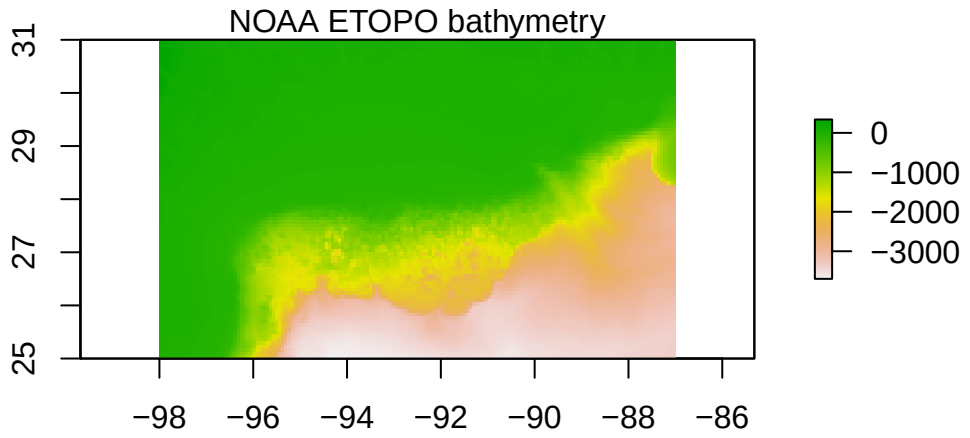
```
plot(plat_rast,
     col = cmocean('turbid')(100))
```



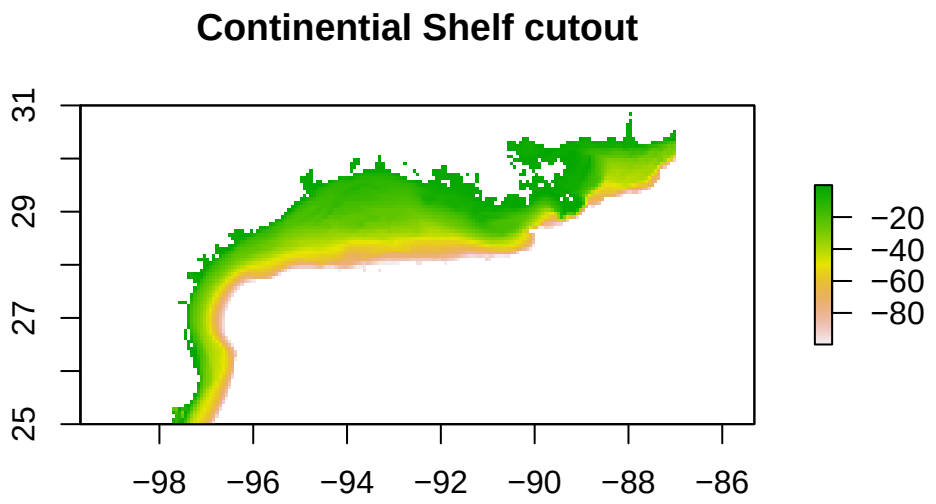
4.3 Customized bathymetric shapefile

Say your interested in creating a shapefile that only encompasses the continential shelf out to 100 m depth. This is easy. First, import bathymetry select the depth range of interest and make it a polygon using the as.polygons function in the terra package. I downloaded a subsetted version of NOAA's ETOPO bathymetry. There are other ways to get bathymetry like THREDDS, ERDDAP, or the marmap r package. Next, I will crop it to the extent of the oil rig data and add a little extra on each side to capture the full extent.

```r
# bathy <- rast('etopo1.nc')
bathy <- marmap::getNOAA.bathy(lon1 = -98, lon2 = -87, lat1 = 25, lat2 = 31)
bathy <- marmap::as.raster(bathy)

plot(bathy)
mtext('NOAA ETOPO bathymetry')
```

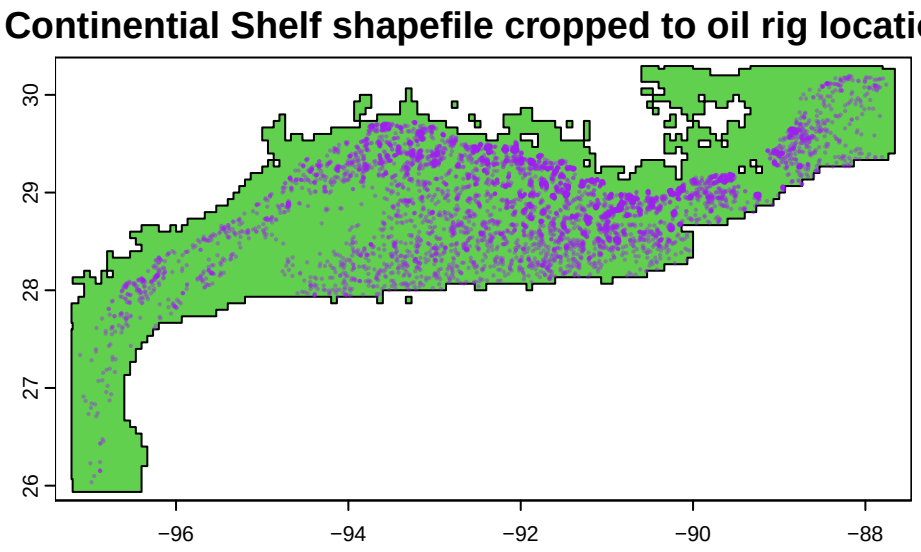



```r
# ### isolate shelf
# bathy[values(bathy$Band1) > 0] <- NA
# bathy[values(bathy$Band1) < (-100)] <- NA

### isolate shelf (new data) #BT EDIT
bathy[bathy@data@values > 0] <- NA
bathy[bathy@data@values < (-100)] <- NA

plot(bathy,
     main = 'Continential Shelf cutout')
```




```
### set all values to 1; then make into shapefile
# bathy[!is.na(bathy@data@values)] <- 1


bathy<-rast(bathy) #BT EDIT
bathy2 <- terra::as.polygons(bathy) #BT EDIT
bathy2 <- terra::aggregate(bathy2) #BT EDIT


bathy_c <- crop(bathy2,
                ext(min(platforms$Longitude,na.rm=T)-.1,
                    max(platforms$Longitude,na.rm=T)+.1,
                    min(platforms$Latitude,na.rm=T)-.1,
                    max(platforms$Latitude,na.rm=T)+.1)
)

st_crs(platforms_sf) <- crs(bathy_c)
ind <- lengths(st_intersects(platforms_sf,st_as_sf(bathy_c)))>0

plot(bathy_c,
     col = 3,
     main = 'Continential Shelf shapefile cropped to oil rig location',
```

```
      new=F)
plot(platforms_sf$geometry[ind],
      add=T,
      pch = 16,
      cex = .3,
      col = alpha('purple', .4))
```

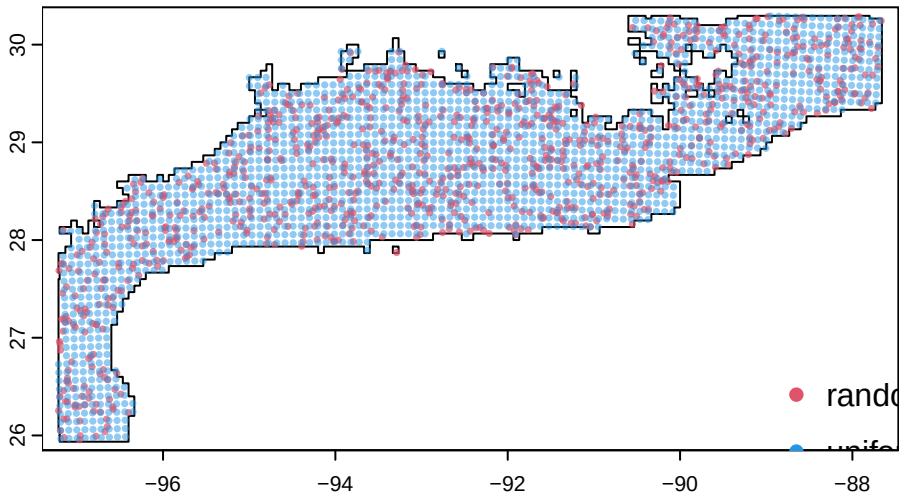


4.4 Random and uniform points

Using the spatstat package we can make random points or uniformly spaced points within our shapefile.

```
gom <- st_as_sf(bathy_c)
ngom <- st_transform(gom, 32615)
wgom <- as.owin(ngom)


### random points
rpts <- runifpoint(1000,wgom)
rpts2 <- st_as_sf(data.frame(lon=rpts$x,lat=rpts$y),coords=c('lon','lat'))
```

```
st_crs(rpts2) <- 32615
rpts2 <- st_transform(rpts2, st_crs(gom))
rpts2 <- st_multipoint(cbind(unlist(rpts2$geometry)[seq(1,nrow(rpts2)*2,2)],
                             unlist(rpts2$geometry)[seq(2,nrow(rpts2)*2,2)]))

### uniformly spaced points
upts <- rsyst(wgom, nx = 100)
upts2 <- st_as_sf(data.frame(lon=upts$x,lat=upts$y),coords=c('lon','lat'))
st_crs(upts2) <- 32615
upts2 <- st_transform(upts2, st_crs(gom))
upts2 <- st_multipoint(cbind(unlist(upts2$geometry)[seq(1,nrow(upts2)*2,2)],
                             unlist(upts2$geometry)[seq(2,nrow(upts2)*2,2)]))


plot(bathy_c)
points(rpts2,pch=16,cex=.5,col=alpha(2,.7))
points(upts2,pch=16,cex=.5,col=alpha(4,.5))
legend(-89,27,c('random pts','uniform pts'),pch=16,col=c(2,4),bty='n')
```



Part III

Second Part

5 Products

References

Knuth, Donald E. 1984. “Literate Programming.” *Comput. J.* 27 (2): 97–111. https://doi.org/10.1093/comjnl/27.2.97.