methods bpow

Aurore A. Maureaud

2022

Brief description: This document presents the combination of steps performed either on R, or on the geospatial software (ArcGIS, but reproducible on the open source QGIS) to create the biogeographic layer of the benthic provinces of the world (bpow).

Data

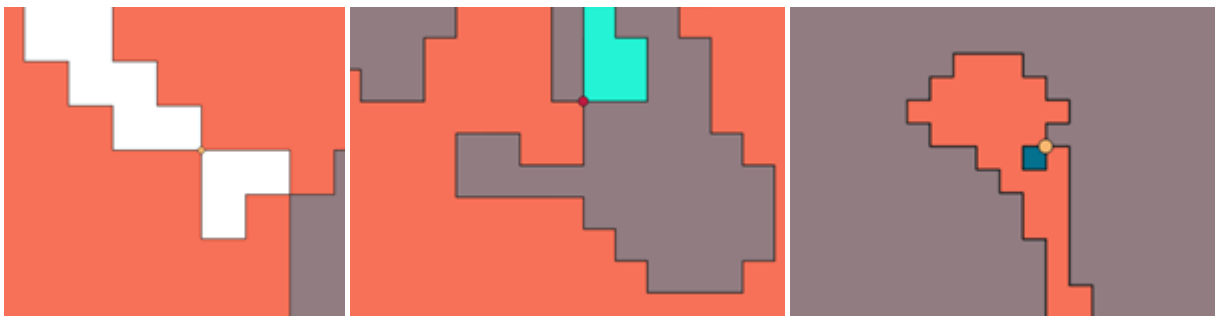
- Marine Ecoregions of the World: 232 ecoregions, 62 provinces, and 12 realms, Spalding et al., 2007 (Spalding et al. 2007)
- Deep sea GOODS provinces: 14 bathyal and 14 abyssal provinces Watling et al., 2013 (Watling et al. 2013)
- Hadal provinces Belyaev, 1989 (Belyaev 1989)
- Global depth raster from GEBCO (Group 2020)
- Land polygons with major islands from Natural Earth at 10m resolution, version 4.1.0

Analysis steps

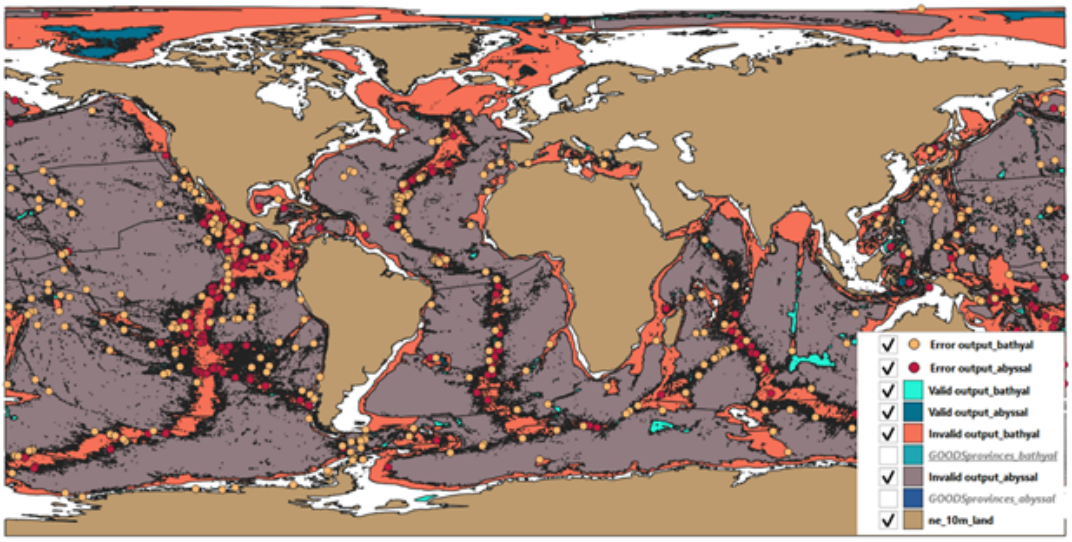
Part 1: check validity of DSP geometries

Performed on ArcGIS Pro

- Step #1: Both the bathyal and abyssal layers were checked because we noticed some wrong geometries in it when trying to perform geometric operations on Arc/QGIS. A lot of points were identified with wrong geometries because of self-intersections.



- Step #2: The validity of geometries was checked with the command "Check validity" of QGIS `Vector → Geometry Tools → Check Validity → Input Layer: BATHYAL/ABYSSAL`



Part 2: correcting geometries for the DSP shapefile layers

Performed on R

- Step #1: check validity of geometries on R with the sf function `st_is_valid()`to check what geometric problem is occurring. All issues are due to self-intersections on both objects
- Step #2: correct the geometric issues in both layers with the sf R function `st_make_valid()`
- Step #3: files saved with the function with the sf R function `st_write()`

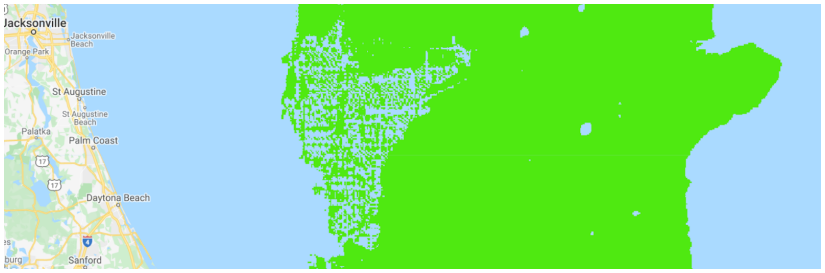
```
# load libraries
library(sf)
library(tidyverse)
library(ggplot2)
library(raster)
library(exactextractr)
library(rgdal)
library(fasterize)
sf::sf_use_s2(FALSE)
library(polyclip)
library(tiff)
library(raster)
library(geosphere)
library(fasterize)
library(smoothie)
library(exactextractr)


### Abyssal & Bathyal provinces from Watling et al., 2013
### Related publication: https://www.sciencedirect.com/science/article/abs/pii/S0079661112001693?via%3D
### Access to shapefiles given by the author
abyssal <- st_read(dsn = "data/goods_provinces", layer = "GOODSprovinces_abyssal")
bathyal <- st_read(dsn = "data/goods_provinces", layer = "GOODSprovinces_bathyal")


### A. Work on abyssal self intersections
abyssal_validation <- st_is_valid(abyssal, reason=TRUE, NA_on_exception = FALSE)
```

```
View(abyssal_validation) # 189 abyssal with self-ring intersection

abyssal_corrected <- st_make_valid(abyssal)

abyssal_validation2 <- st_is_valid(abyssal_corrected, reason=TRUE, NA_on_exception = FALSE)
unique(abyssal_validation2) # all valid geometries


### B. Work on bathyal self intersections
bathyal_validation <- st_is_valid(bathyal, reason=TRUE, NA_on_exception = FALSE)
View(bathyal_validation) # 341 bathyal with self-ring intersection

bathyal_corrected <- st_make_valid(bathyal)

bathyal_validation2 <- st_is_valid(bathyal_corrected, reason=TRUE, NA_on_exception = FALSE)
unique(bathyal_validation2) # all valid geometries


### C. Save new files
st_write(abyssal_corrected, dsn = "outputs/deep_sea/GOODSprovinces_abyssal_Rfix.shp")
st_write(bathyal_corrected, dsn = "outputs/deep_sea/GOODSprovinces_bathyal_Rfix.shp")
```

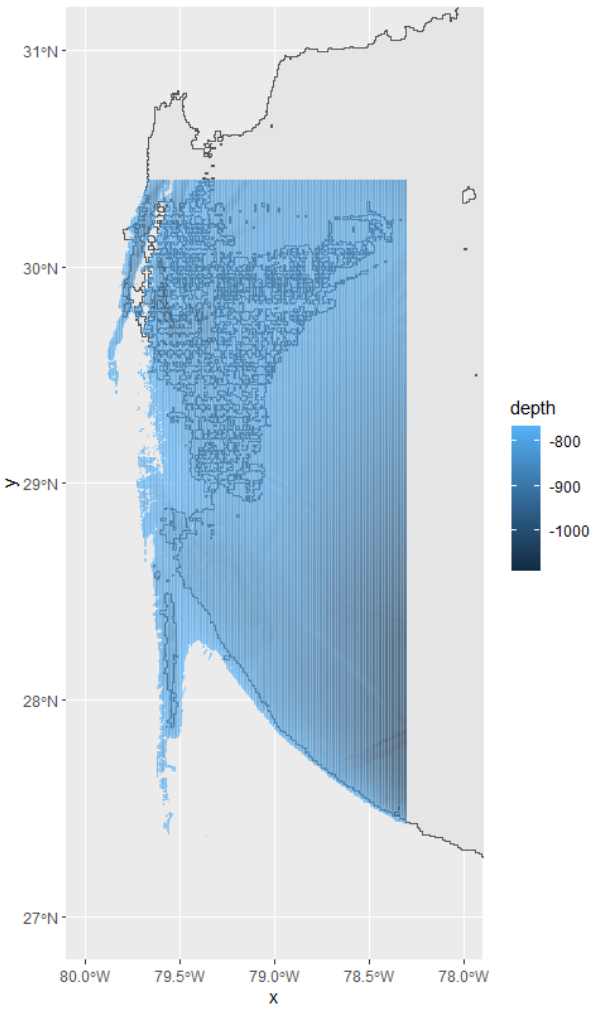
Part 3: Removing irregularities from the bathyal shapefile

Performed on R

- Step #1: identification of areas: off the Florida coast, because the threshold of 800m is arbitrary and this is a "transition" zone



- Step #2: fixed on R, using GEBCO and adding additional areas to the bathyal layer
- Step #3: all polygons intersecting the depth raster (in blue below) are merged via `st_union()` to create one larger replacing polygons, added to the bathyal layer



- Step #4: geometry checked again and file saved with the function with the sf R function`st_write()`

```
# load libraries
library(sf)
library(tidyverse)
library(ggplot2)
library(raster)
library(exactextractr)
library(rgdal)
library(fasterize)
sf::sf_use_s2(FALSE)
library(polyclip)
library(tiff)
library(raster)
library(geosphere)
library(fasterize)
library(smoothie)
library(exactextractr)

rm(list = ls())

# load new bathyal shapefile
```

```
bathyal <- st_read("outputs/deep_sea/GOODSprovinces_bathyal_Rfix.shp")

# fix florida region irregularities - object 17014 - will fix problems with
# coast intersection
# ggplot(bathyal) + geom_sf() +
#   coord_sf(xlim = c(-80,-78), ylim = c(28,31))
#
# ggplot(bathyal[17014,]) + geom_sf()
# bbox_one <- st_bbox(bathyal[17014,])
# load depth raster
depth_n90_s0_w90_e0 <- raster("data/gebco_2020_ascii/gebco_2020_n90.0_s0.0_w-90.0_e0.0.asc")
depth <- crop(depth_n90_s0_w90_e0, y = extent(-80,-78.3,27,30.4))

depth[depth>-765] <- NA
plot(depth)

# extract necessary polygons
depth_spdf <- as(depth, "SpatialPixelsDataFrame")
depth_df <- as.data.frame(depth_spdf)
colnames(depth_df) <- c("depth", "x", "y")

# select polygons from shapefile and depth raster
depth_pts <- data.frame(rasterToPoints(depth))
depth_pts_sf <- st_as_sf(depth_pts, coords = c("x","y"), crs = st_crs(bathyal))
select_polygons <- st_intersects(depth_pts_sf, bathyal)
select_polygons <- unique(unlist(select_polygons))
select_polygons <- bathyal[select_polygons,]

ggplot(select_polygons) + geom_sf() +
  coord_sf(xlim = c(-80,-78), ylim = c(27,31)) +
  geom_tile(data = depth_df, aes(x = x, y = y, fill = depth), alpha = 0.5, color = NA)

new_poly <- rasterToPolygons(aggregate(depth, fact = 2, fun = mean))
new_poly <- st_as_sf(new_poly, crs = st_crs(bathyal))
new_poly <- new_poly %>%
  mutate(fix = "fix") %>%
  group_by(fix) %>%
  summarize(geometry = st_union(geometry)) %>%
  mutate(ID = bathyal[17014,]$ID,
         Province = bathyal[17014,]$Province,
         Name = bathyal[17014,]$Name) %>%
  dplyr::select(-fix)

ggplot(new_poly) + geom_sf(fill = "red", alpha= 0.5) +
  geom_sf(data = select_polygons, fill = "blue") +
  coord_sf(xlim = c(-80,-78), ylim = c(27,31))

new_poly <- rbind(new_poly, select_polygons) %>%
  group_by(Province,Name) %>%
  summarize(geometry = st_union(geometry)) %>%
  mutate(ID = 17014)

ggplot(new_poly) + geom_sf(fill = 'red') +
```

```
  #geom_sf(data = select_polygons, fill = "blue") +
  coord_sf(xlim = c(-80,-78), ylim = c(27,31))

# save the new polygon
bathyal <- bathyal %>%
  filter(!ID %in% select_polygons$ID)
bathyal_fixed <- rbind(bathyal, new_poly)

# check validity
bathyal_validation <- st_is_valid(bathyal_fixed, reason=TRUE, NA_on_exception = FALSE)
unique(bathyal_validation) # 341 bathyal with self-ring intersection

bathyal_corrected <- st_make_valid(bathyal_fixed)

# save new bathyal shapefile
st_write(bathyal_corrected, dsn = "outputs/deep_sea/GOODSprovinces_bathyal_Rfix_irregularities.shp",
         append = T)
```

Part 4: Merging the bathyal and abyssal shapefile layers

Performed on ArcGIS Pro

- Step #1: Check that the layers are perfectly complementary to each other by manual visualization, and they are not

- Step #2: Creating perfectly complementary abyssal and bathyal layers, by taking the difference between the two layers. All grid cells that are shared between the abyssal and bathyal habitats will be associated with the bathyal layer, because they may represent more potential habitat than abyssal areas.

  Operation: geometric difference between the bathyal layer and the abyssal layer

  `QGIS: Vector → Geoprocessing tools → Difference → Input Layer: bathyal, Overlay Layer: abyssal`

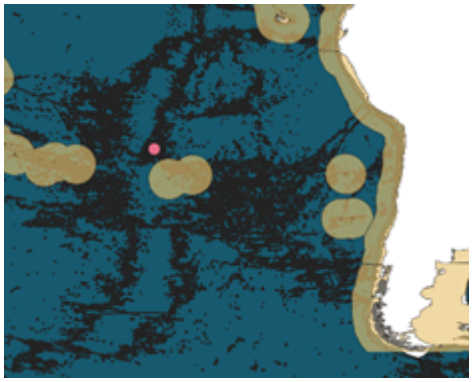
- `ArcGIS: Geoprocessing → Pairwise Erase, input = abyssal, erase = bathyal, output = p4s2`

- Step #3: Creating one shapefile with the abyssal and bathyal layers which I will then call the deepsea layer.

  Operation: merging layers of the two shapefiles to form one deep sea shapefile

  `QGIS: Vector → Data Management Tools → Merge layers → Input Layers: ABYSSAL + BATHYAL`
  `ArcGIS: Merge, input = p4s2, input = GOODSprovinces_bathyal_irregularities, output = GOODSprovinces_p4s3`

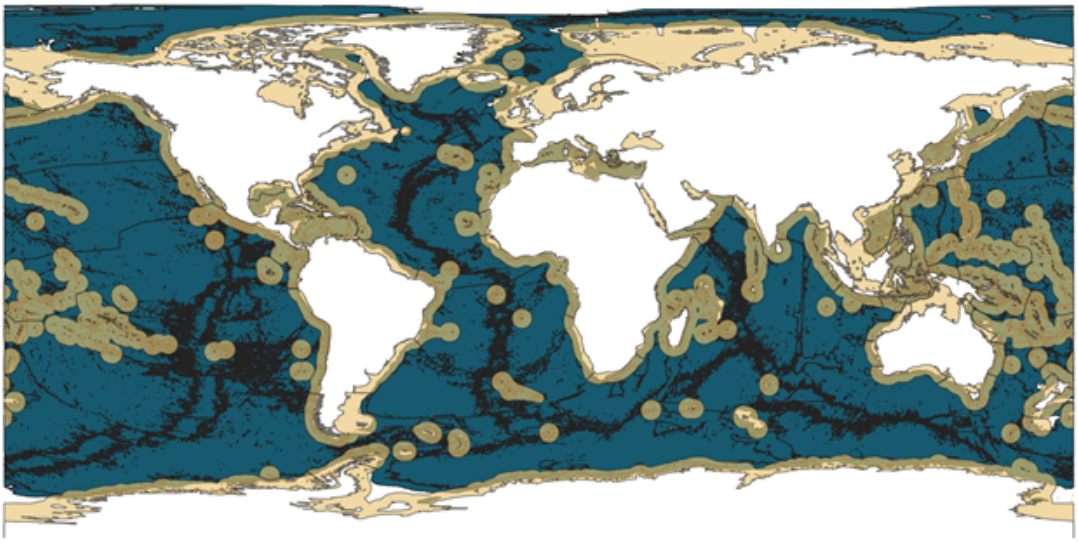
- Step #4: Correcting invalid geometry manually



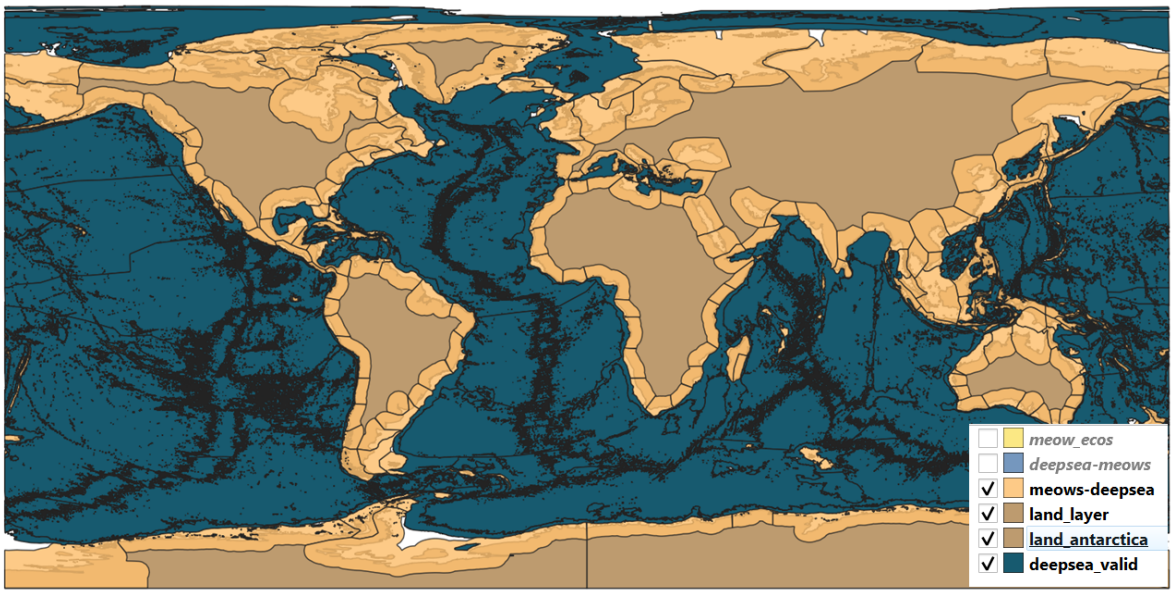
Part 5: Merging the coastal and deep sea shapefile layers

Performed on ArcGIS Pro

- Step #1: Here, two options are possible: (i) removing marine ecoregion areas and keeping them associated with the deep-sea layers (ii) removing areas from the deep sea areas and keeping them associated with the ecoregions. Land areas are identified with the natural earth file because the coastal ecoregions already include more than coastlines and don't need to include the best coastline resolution.



Operation (i): geometric difference between the MEOW layer and the DEEPSEA layer



QGIS: Vector → Geoprocessing Tools → Difference → Input Layer: MEOW, Overlay Layer: DEEPSEA ArcGIS: Geoprocessing → Pairwise Erase, input = meows_ecos, erase = GOODSprovinces_p4s3, output = provinces_p5s1

- Step #2: Creating a shapefile for joining the layers MEOW and DEEPSEA that after step#1 should be perfectly complementary (no spatial overlap) that I will then call the SEAFLOOR shapefile layer.

  QGIS: Vector → Data Management Tools → Merge Vector Layers → Input Layers: MEOWS-DEEPSEA, DEEPSEA

  ArcGIS: Geoprocessing → Merge, input = provinces_p5s1, input = GOODSprovinces_p4s3, output = provinces_p5s2

- Step #3: extract provinces_p5s2 as a shapefile
- Step #4: fix the format of provinces_p5s2 in R

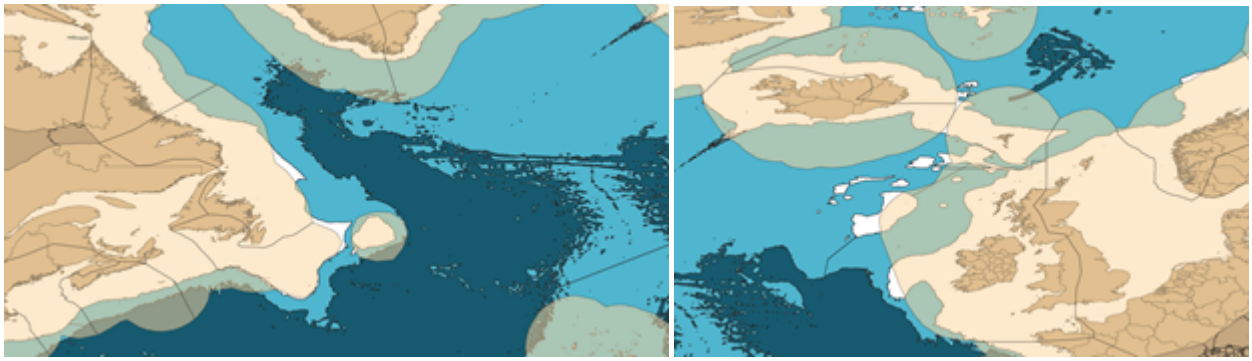
```
# load libraries
library(sf)
library(tidyverse)
library(ggplot2)
library(rgdal)
library(fasterize)
sf::sf_use_s2(FALSE)

# loadlayer after 1st processing on Arc
eco <- st_read("outputs/arcpro/post-processing_1/Provinces_P5S2.shp") %>%
  mutate(type = case_when(MERGE_SRC == "provinces_p5s1" ~ "coastal",
                          MERGE_SRC == "p4s2" ~ "abyssal",
                          MERGE_SRC == "GOODSprovinces_bathyal_Rfix_irregularities" ~ "bathyal"),
         prov_n = PROVINCE,
         prov_id = PROV_CODE,
         prov_id = ifelse(type %in% c("abyssal", "bathyal"), PROVINCE, prov_id),
         prov_n = ifelse(type %in% c("abyssal","bathyal"), Name, prov_n),
         eco_n = ECOREGION,
         eco_id = ECO_CODE_X,
         eco_id = ifelse(eco_id ==0, NA, eco_id),
```

```
          rlm_id = RLM_CODE,
          rlm_n = REALM) %>%
  dplyr::select(ID, type, prov_n, prov_id, eco_n, eco_id, rlm_n, rlm_id)

st_write(eco, dsn = "outputs/bpow/bpow_p5s4.shp")
```

Part 6: Identify remaining areas

Performed on ArcGIS Pro

- Step #1: two types of areas will not yet be characterized in the SEAFLOOR layer, (i) hadal regions below 6,500m, where we should not really observe species, (ii) regions between 200 and 800m that don't belong to ecoregions. The bathyal layer starts at 800m and onwards, so there are remaining zones to associate with a region.



- Step #2: get a shapefile with all missing polygons: (merge of seafloor shapefile with a low resolution land layer (does not matter since meows have a bunch of land areas already)

```
-ArcGIS: Pairwise erase, input = ne_10m_land, erase = Provinces_P5S2, output =
Provinces_P5S2_PairwiseErase
-ArcGIS: Merge, input = Provinces_P5S2_PairwiseErase, input = Provinces_P5S2, output =
Provinces_P5S2_merge
```

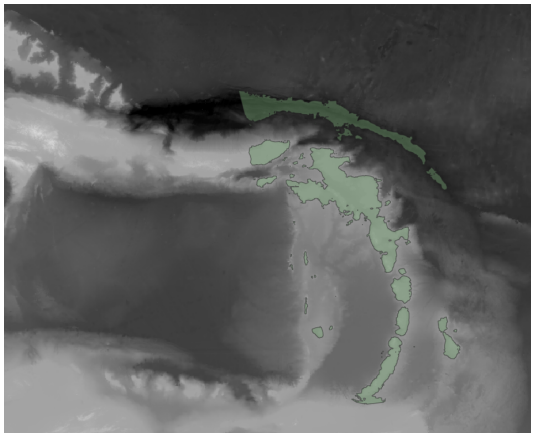
- Step #3: get polygons that are uncharacterized

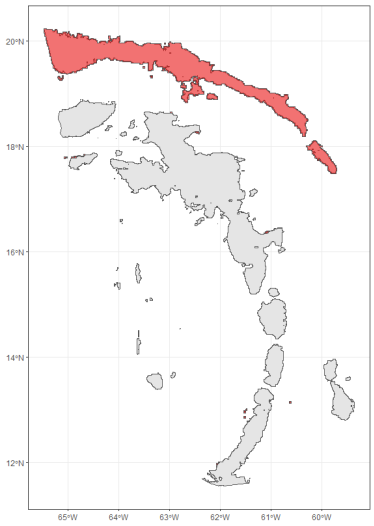
```
-ArcGIS: Create feature, extent of -180;180;-90,90
-ArcGIS: Pairwise erase, input = extent, erase = Provinces_P5S2_PairwiseErase, output =
holes_p6s2
```

Part 7: Remove hadal and classify provinces

Performed on R

- Step #1: Identification of hadal trenches of the world are done based on rough identification of relevant coastal ecoregions based on (Belyaev 1989) and (Jamieson et al. 2010).
- Step #2: For each identified ecoregion from Step #1, we use the corresponding depth raster from GEBCO to identify trenches (below 6,500m deep) and create new polygons from there. For instance, on the figure below, the ecoregion "Eastern Caribbean" from provinces_p5s2 include deep area on the northern sites.





- Step #3: All polygons are merged back to the provinces shapefile, where coastal ecoregions do not include hadal trenches anymore, and they are added as separate geometries.

```
rm(list = ls())

# load libraries
library(sf)
library(tidyverse)
library(ggplot2)
library(raster)
library(exactextractr)
library(rgdal)
library(fasterize)
sf::sf_use_s2(FALSE)
library(units)
library(ggtern)
library(rnaturalearth)
library(rnaturalearthdata)
library(tricolore)
library(ggtern)
world <- ne_countries(scale = "medium", returnclass = "sf")
library(tiff)
library(RColorBrewer)
library(egg)

### Libraries
library(raster)
library(geosphere)
library(fasterize)
library(smoothie)
library(exactextractr)


# load biogeography layer
eco <- st_read("outputs/bpow/bpow_p5s4.shp")

holes <- st_read(dsn = "outputs/arcpro/post-processing_1", layer = "holes_p6s2_correct")
```

```
eco <- st_transform(eco, crs = st_crs(holes))
rm(holes)

# Load all depth shapefiles
depth_n0_s90_w180_e90 <- raster("data/gebco_2020_ascii/gebco_2020_n0.0_s-90.0_w-180.0_e-90.0.asc")
depth_n0_s90_w90_e0 <- raster("data/gebco_2020_ascii/gebco_2020_n0.0_s-90.0_w-90.0_e0.0.asc")
depth_n0_s90_w0_e90 <- raster("data/gebco_2020_ascii/gebco_2020_n0.0_s-90.0_w0.0_e90.0.asc")
depth_n0_s90_w90_e180 <- raster("data/gebco_2020_ascii/gebco_2020_n0.0_s-90.0_w90.0_e180.0.asc")
depth_n90_s0_w180_e90 <- raster("data/gebco_2020_ascii/gebco_2020_n90.0_s0.0_w-180.0_e-90.0.asc")
depth_n90_s0_w90_e0 <- raster("data/gebco_2020_ascii/gebco_2020_n90.0_s0.0_w-90.0_e0.0.asc")
depth_n90_s0_w0_e90 <- raster("data/gebco_2020_ascii/gebco_2020_n90.0_s0.0_w0.0_e90.0.asc")
depth_n90_s0_w90_e180 <- raster("data/gebco_2020_ascii/gebco_2020_n90.0_s0.0_w90.0_e180.0.asc")

obj = c('depth_n90_s0_w180_e90', 'depth_n90_s0_w90_e0',
        'depth_n90_s0_w0_e90', 'depth_n90_s0_w90_e180',
        "depth_n0_s90_w180_e90", "depth_n0_s90_w90_e0",
        'depth_n0_s90_w0_e90', 'depth_n0_s90_w90_e180')
x.min = c(-180, -90, 0, 90, -180, -90, 0, 90)
x.max = c(-90, 0, 90, 180, -90, 0, 90, 180)
y.min = c(0, 0, 0, 0, -90, -90, -90, -90)
y.max = c(90, 90, 90, 90, 0, 0, 0, 0)

depth_files <- data.frame(obj) %>%
  mutate(xmin = x.min,
         xmax = x.max,
         ymin = y.min,
         ymax = y.max)
select_files_r <- raster(nrows = 2, ncols = 4, xmn = -180, xmx = 180,
                         ymn = -90, ymx = 90) %>%
  rasterToPolygons() %>%
  st_as_sf()


################################################################################
#### Remove hadal regions
################################################################################
eco_no_hadal <- eco

hadal_ecoregions <- c(46,47,48,51,53,54,122, #hadal 1
                      121,127,128,129, #hadal 2
                      123,125, #hadal 3
                      130,134,135,136,148, #hadal 4
                      146,157,195,196, #hadal 5
                      171,175,176,177,178, #hadal 6
                      111,119,131,132,120, #hadal 7
                      63,64,65,68, #hadal 8
                      60,166,167,168, #hadal 9
                      219,220) #hadal 10

# adapted from Belyaev 1989 & Jamieson, 2010
had_n <- data.frame(c("Aleutian-Japan",
                      "Philippine",
                      "Mariana",
```

```
                      "Bougainville-New Hebrides",
                      "Tonga-Kermadec",
                      "Peru-Chile",
                      "Java",
                      "Puerto Rico",
                      "Middle America",
                      "Southern Antilles"))

hadal <- data.frame(had_n) %>%
  mutate(type = "hadal",
         ID = c(180463:180472),
         prov_n = NA_character_,
         prov_id = NA,
         eco_n = NA_character_,
         eco_id = NA,
         rlm = NA_character_,
         rlm_id = NA,
         had_id = c(1:nrow(had_n))) %>%
  rename(had_n = `c..Aleutian.Japan....Philippine....Mariana....Bougainville.New.Hebrides...`)
hadal_poly <- data.frame()

abyssal <- data.frame(unique(eco[eco$type=="abyssal",]$prov_n)) %>%
  mutate(type = "abyssal",
         ID = c(180473:180486),
         prov_id = unique(eco[eco$type =="abyssal",]$prov_id),
         eco_n = NA_character_,
         eco_id = NA,
         rlm = NA_character_,
         rlm_id = NA,
         had_id = NA,
         had_n = NA_character_) %>%
  rename( prov_n = `unique.eco.eco.type.....abyssal.....prov_n.`)
abyssal_poly <- data.frame()


for(h in 1:length(hadal_ecoregions)){

  print(h)
  hadal_one <- eco[which(eco$eco_id == hadal_ecoregions[h]),]

  if(st_is_valid(hadal_one)==FALSE){
    hadal_one <- st_make_valid(hadal_one)
  }

  # extract lat/long boundaries
  bbox_one <- st_bbox(hadal_one)

  bbox_r <- raster(nrow=1, ncol=1, xmn = bbox_one[1], xmx = bbox_one[3],
                   ymn = bbox_one[2], ymx = bbox_one[4])

  overlap_rr <- coverage_fraction(bbox_r, select_files_r)
  select_depth <- c()
  for(r in 1:8){
```

```
  if(values(overlap_rr[[r]])==1){select_depth[length(select_depth)+1] <- depth_files$obj[r]}
  if(values(overlap_rr[[r]])>0 && values(overlap_rr[[r]]<1)){select_depth[length(select_depth)+1] <- d
}

if(length(select_depth)>1){
  for(k in 1:length(select_depth)){
    depth_k <- crop(get(select_depth[k]), y = extent(bbox_one[1],bbox_one[3],bbox_one[2],bbox_one[4]))
    if(k==1){depth <- depth_k}
    else{depth <- merge(depth, depth_k)}
    rm(depth_k)
  }
}
if(length(select_depth)==1){
  depth <- crop(get(select_depth), y = extent(bbox_one[1],bbox_one[3],bbox_one[2],bbox_one[4]))
}

depth_poly <- exact_extract(depth, hadal_one, include_xy = T)

# save the hadal polygon
new_r <- aggregate(depth, fact = 2, fun = min)
new_r[new_r>(-6500)] <- NA

overlap_ri <- coverage_fraction(new_r, hadal_one)[[1]]
overlap_ri[overlap_ri==0] <- NA
overlap_ri[overlap_ri>0] <- 1
new_ri <- new_r*overlap_ri
if(length(unique(new_ri))!=0){
  new_poly <- rasterToPolygons(new_ri)
  new_poly <- st_as_sf(new_poly, crs = st_crs(eco)) %>%
    mutate(ID = 1) %>%
    group_by(ID) %>%
    summarize(geometry = st_union(geometry))

  # ggplot(hadal_one) + geom_sf() + theme_bw() +
  #   geom_sf(data = new_poly, fill = "red", alpha = 0.5)
  # ggplot(new_poly) + geom_sf()

  if(st_is_valid(new_poly)==FALSE){
    new_poly <- st_make_valid(new_poly)
  }
  corr_poly <- st_difference(hadal_one, new_poly)

  # modify coastal ecoregion
  st_geometry(eco_no_hadal[which(eco_no_hadal$eco_id == hadal_ecoregions[h]),]) <- st_geometry(corr_po

  if (hadal_ecoregions[h] %in% c(46,47,48,51,53,54,122)){
    new_poly <- new_poly %>%
      dplyr::select(-ID)
    new_poly <- st_as_sf(cbind(hadal[1,],new_poly))
    hadal_poly <- rbind(hadal_poly, new_poly)
  } else if (hadal_ecoregions[h] %in% c(121,127,128,129)){
    new_poly <- new_poly %>%
      dplyr::select(-ID)
```

```
    new_poly <- st_as_sf(cbind(hadal[2,],new_poly))
    hadal_poly <- rbind(hadal_poly, new_poly)
  } else if (hadal_ecoregions[h] %in% c(123,125)){
    new_poly <- new_poly %>%
      dplyr::select(-ID)
    new_poly <- st_as_sf(cbind(hadal[3,],new_poly))
    hadal_poly <- rbind(hadal_poly, new_poly)
  } else if (hadal_ecoregions[h] %in% c(130,134,135,136,148)){
    new_poly <- new_poly %>%
      dplyr::select(-ID)
    new_poly <- st_as_sf(cbind(hadal[4,],new_poly))
    hadal_poly <- rbind(hadal_poly, new_poly)
  } else if (hadal_ecoregions[h] %in% c(146,157,195,196)){
    new_poly <- new_poly %>%
      dplyr::select(-ID)
    new_poly <- st_as_sf(cbind(hadal[5,],new_poly))
    hadal_poly <- rbind(hadal_poly, new_poly)
  } else if (hadal_ecoregions[h] %in% c(171,175,176,177,178)){
    new_poly <- new_poly %>%
      dplyr::select(-ID)
    new_poly <- st_as_sf(cbind(hadal[6,],new_poly))
    hadal_poly <- rbind(hadal_poly, new_poly)
  } else if (hadal_ecoregions[h] %in% c(111,119,131,132,120)){
    new_poly <- new_poly %>%
      dplyr::select(-ID)
    new_poly <- st_as_sf(cbind(hadal[7,],new_poly))
    hadal_poly <- rbind(hadal_poly, new_poly)
  } else if (hadal_ecoregions[h] %in% c(63,64,65,68)){
    new_poly <- new_poly %>%
      dplyr::select(-ID)
    new_poly <- st_as_sf(cbind(hadal[8,],new_poly))
    hadal_poly <- rbind(hadal_poly, new_poly)
  } else if (hadal_ecoregions[h] %in% c(60,166,167,168)){
    new_poly <- new_poly %>%
      dplyr::select(-ID)
    new_poly <- st_as_sf(cbind(hadal[9,],new_poly))
    hadal_poly <- rbind(hadal_poly, new_poly)
  } else if (hadal_ecoregions[h] %in% c(219,220)){
    new_poly <- new_poly %>%
      dplyr::select(-ID)
    new_poly <- st_as_sf(cbind(hadal[10,],new_poly))
    hadal_poly <- rbind(hadal_poly, new_poly)}

}

# remove abyssal from coastal
new_r2 <- aggregate(depth, fact = 2, fun = min)
new_r2[new_r2>(-3500)] <- NA
new_r2[new_r2<(-6500)] <- NA

overlap_ri2 <- coverage_fraction(new_r2, hadal_one)[[1]]
overlap_ri2[overlap_ri2==0] <- NA
overlap_ri2[overlap_ri2>0] <- 1
```

```
new_ri2 <- new_r2*overlap_ri2

if(length(unique(new_ri2))!=0 & length(unique(new_ri))!=0){
  new_poly2 <- rasterToPolygons(new_ri2)
  new_poly2 <- st_as_sf(new_poly2, crs = st_crs(eco)) %>%
    mutate(ID = 1) %>%
    group_by(ID) %>%
    summarize(geometry = st_union(geometry))

  if(st_is_valid(new_poly2)==FALSE){
    new_poly2 <- st_make_valid(new_poly2)
  }
  corr_poly2 <- st_difference(corr_poly, new_poly2)

  # modify coastal ecoregion
  st_geometry(eco_no_hadal[which(eco_no_hadal$eco_id == hadal_ecoregions[h]),]) <- st_geometry(corr_p

  if (hadal_ecoregions[h] %in% c(46,47,48,51,53,54)){
    new_poly2 <- new_poly2 %>%
      dplyr::select(-ID)
    new_poly2 <- st_as_sf(cbind(abyssal[2,],new_poly2))
    abyssal_poly <- rbind(abyssal_poly, new_poly2)
    # two abyssal regions for 135
  } else if (hadal_ecoregions[h] %in% c(121,122,123,125,127,129,130)){
    new_poly2 <- new_poly2 %>%
      dplyr::select(-ID)
    new_poly2 <- st_as_sf(cbind(abyssal[3,],new_poly2))
    abyssal_poly <- rbind(abyssal_poly, new_poly2)
  } else if (hadal_ecoregions[h] %in% c(128,131)){
    new_poly2 <- new_poly2 %>%
      dplyr::select(-ID)
    new_poly2 <- st_as_sf(cbind(abyssal[4,],new_poly2))
    abyssal_poly <- rbind(abyssal_poly, new_poly2)
  } else if (hadal_ecoregions[h] %in% c(134,136,146,148,157,195,196)){
    new_poly2 <- new_poly2 %>%
      dplyr::select(-ID)
    new_poly2 <- st_as_sf(cbind(abyssal[6,],new_poly2))
    abyssal_poly <- rbind(abyssal_poly, new_poly2)
  } else if (hadal_ecoregions[h] %in% c(60,166,167,168,171,175,176,177,178,111)){
    new_poly2 <- new_poly2 %>%
      dplyr::select(-ID)
    new_poly2 <- st_as_sf(cbind(abyssal[14,],new_poly2))
    abyssal_poly <- rbind(abyssal_poly, new_poly2)
  } else if (hadal_ecoregions[h] %in% c(119,120,132)){
    new_poly2 <- new_poly2 %>%
      dplyr::select(-ID)
    new_poly2 <- st_as_sf(cbind(abyssal[7,],new_poly2))
    abyssal_poly <- rbind(abyssal_poly, new_poly2)
  } else if (hadal_ecoregions[h] %in% c(63,64,65,68)){
    new_poly2 <- new_poly2 %>%
      dplyr::select(-ID)
    new_poly2 <- st_as_sf(cbind(abyssal[12,],new_poly2))
    abyssal_poly <- rbind(abyssal_poly, new_poly2)
```

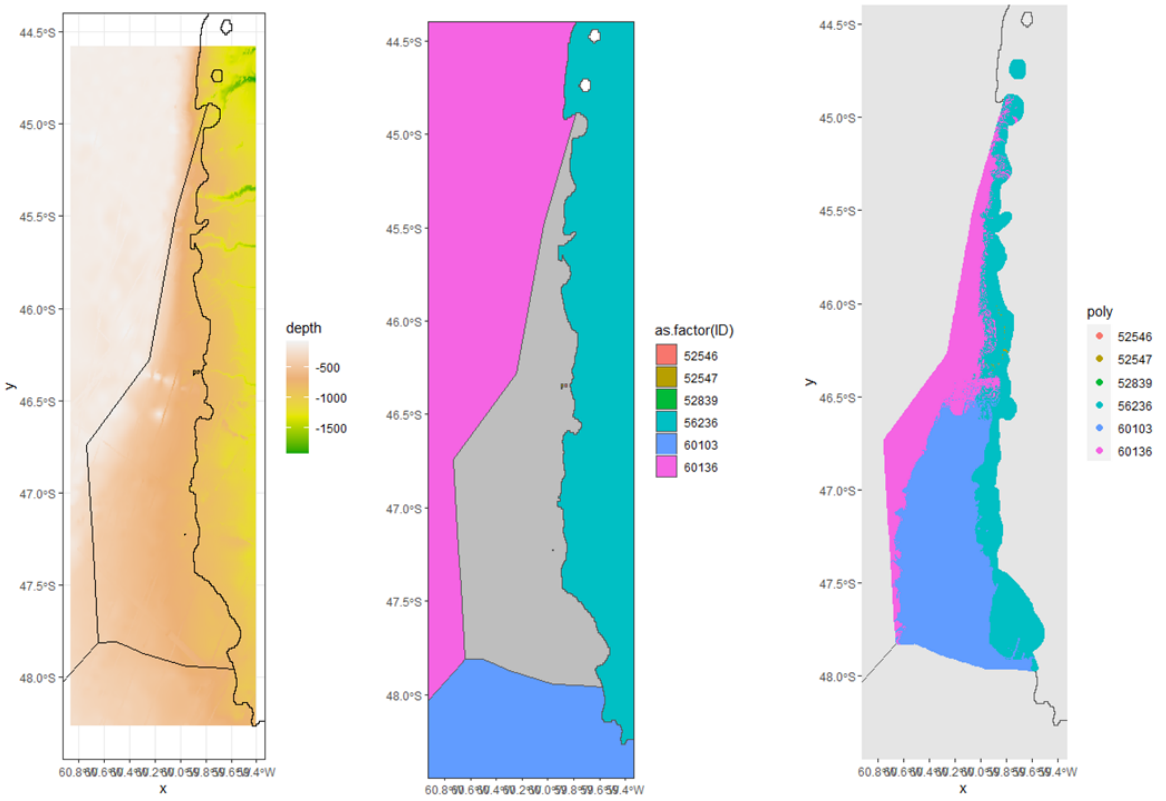
```
    } else if (hadal_ecoregions[h] %in% c(219,220)){
      new_poly2 <- new_poly2 %>%
        dplyr::select(-ID)
      new_poly2 <- st_as_sf(cbind(abyssal[8,],new_poly2))
      abyssal_poly <- rbind(abyssal_poly, new_poly2)
    } else if (hadal_ecoregions[h]==135){
      new_poly2 <- new_poly2 %>%
        dplyr::select(-ID) %>%
        st_cast(to = "POLYGON") %>%
        mutate(latitude = st_coordinates(st_centroid(geometry))[,2])
      new_poly2_a <- new_poly2 %>%
        filter(latitude>(-2)) %>%
        dplyr::select(-latitude) %>%
        st_union()
      new_poly2_b <- new_poly2 %>%
        filter(latitude<(-2)) %>%
        dplyr::select(-latitude) %>%
        st_union()
      new_poly2_a <- st_as_sf(cbind(abyssal[3,],new_poly2_a))
      new_poly2_b <- st_as_sf(cbind(abyssal[6,],new_poly2_b))
      abyssal_poly <- rbind(abyssal_poly, new_poly2_a, new_poly2_b)
      rm(new_poly2_a, new_poly2_b)
    }

    rm(corr_poly, new_poly, overlap_ri, depth_poly, depth, new_ri, new_r,
       select_depth, overlap_rr, bbox_r, bbox_one, hadal_one,
       new_ri2, new_poly2, new_r2, overlap_ri2)
  }
  save.image("outputs/hadal/remove_hadal_abyssal_from_coastal.RData")

}

load("outputs/hadal/remove_hadal_abyssal_from_coastal.RData")

eco_hadal <- st_make_valid(eco_no_hadal) %>%
  mutate(had_id = NA,
         had_n = NA_character_) %>%
  dplyr::select(ID, type, prov_n, prov_id, eco_n, eco_id, rlm_n, rlm_id, had_n, had_id, geometry)

hadal_poly <- st_make_valid(st_as_sf(hadal_poly)) %>%
  rename(rlm_n = rlm) %>%
  group_by(type, prov_n, prov_id, eco_n, eco_id, rlm_n, rlm_id, ID, had_id, had_n) %>%
  summarize(geometry = st_union(geometry)) %>%
  dplyr::select(ID, type, prov_n, prov_id, eco_n, eco_id, rlm_n, rlm_id, had_n, had_id, geometry)

abyssal_poly <- st_make_valid(st_as_sf(abyssal_poly)) %>%
  rename(rlm_n = rlm) %>%
  group_by(type, prov_n, prov_id, eco_n, eco_id, rlm_n, rlm_id, ID, had_id, had_n) %>%
  summarize(geometry = st_union(geometry)) %>%
  dplyr::select(ID, type, prov_n, prov_id, eco_n, eco_id, rlm_n, rlm_id, had_n, had_id, geometry)

eco_hadal <- rbind(eco_hadal, hadal_poly, abyssal_poly)
# transform geometry collection into multipolygon
```

```
st_geometry(eco_hadal[60226,]) <- st_geometry(st_collection_extract(eco_hadal[60226,], type = "POLYGON"))
st_geometry(eco_hadal[100307,]) <- st_geometry(st_collection_extract(eco_hadal[100307,], type = "POLYGON"))
st_geometry(eco_hadal[140385,]) <- st_geometry(st_collection_extract(eco_hadal[140385,], type = "POLYGON"))
st_geometry(eco_hadal[180461,]) <- st_geometry(st_collection_extract(eco_hadal[180461,], type = "POLYGON"))

eco_hadal <- eco_hadal %>%
  dplyr::select(-ID) %>%
  distinct()
eco_hadal$ID <- 1:nrow(eco_hadal)

# save file
st_write(obj = eco_hadal, dsn="outputs/hadal/provinces_p7s3_abyssal_corr.shp",
         append = FALSE)
```

Part 8: Characterize missing regions

Performed on R

- Step #1: associate the depth raster corresponding to a box around the polygon
- Step #2: find the polygons of the seafloor in that box
- Step #3: if there is only one polygon, associate the missing polygon the that polygon
- Step #4: if there are multiple polygons, calculate the closest distance based on 3D (longitude, latitude, depth (a)) with the function mcNNindex and associate the closest polygon based on shortest distance around each point of the raster of the missing polygon (b). Because some areas are transition areas, the boundaries between the polygons are not clean (see figure (c)), so the resulting raster is aggregated at a lower resolution, as well as the grid cell polygon associations (d). Finally, polygons are created and associated with the ID of the closest polygon from the seafloor shapefile.




```
rm(list = ls())

### Libraries
library(sf)
sf::sf_use_s2(FALSE)
library(tidyverse)
library(ggplot2)
library(rgdal)
library(RColorBrewer)
library(raster)
library(geosphere)
library(fasterize)
library(smoothie)
library(exactextractr)
library(Morpho)
library(rgl)
library(units)
library(foreach)


################################################################################
### Load data & fix shapefile attributes
################################################################################
# load provinces_p7s3
seafloor_meow_deepsea <- st_read("outputs/hadal/provinces_p7s3_abyssal_corr.shp")

# load holes shapefile
holes <- st_read("outputs/arcpro/post-processing_1/holes_p6s2_correct.shp") %>%
```

```
  st_cast("POLYGON") %>%
  mutate(Shape_Area = drop_units(st_area(geometry)),
         Shape_Leng = drop_units(st_length(geometry))) %>%
  dplyr::select(-Id)
holes$ID <- c(1:nrow(holes))

st_write(obj = holes, dsn = "outputs/holes/holes_p6s2_id.shp")
# for(i in 1:nrow(holes)){
#   png(file = paste0("figures/holes/",i,".png"))
#   print(ggplot(holes[i,]) + geom_sf())
#   dev.off()
# }

holes <- holes %>%
  filter(ID != 1553)

# Load all depth shapefiles
depth_n0_s90_w180_e90 <- raster("data/gebco_2020_ascii/gebco_2020_n0.0_s-90.0_w-180.0_e-90.0.asc")
depth_n0_s90_w90_e0 <- raster("data/gebco_2020_ascii/gebco_2020_n0.0_s-90.0_w-90.0_e0.0.asc")
depth_n0_s90_w0_e90 <- raster("data/gebco_2020_ascii/gebco_2020_n0.0_s-90.0_w0.0_e90.0.asc")
depth_n0_s90_w90_e180 <- raster("data/gebco_2020_ascii/gebco_2020_n0.0_s-90.0_w90.0_e180.0.asc")
depth_n90_s0_w180_e90 <- raster("data/gebco_2020_ascii/gebco_2020_n90.0_s0.0_w-180.0_e-90.0.asc")
depth_n90_s0_w90_e0 <- raster("data/gebco_2020_ascii/gebco_2020_n90.0_s0.0_w-90.0_e0.0.asc")
depth_n90_s0_w0_e90 <- raster("data/gebco_2020_ascii/gebco_2020_n90.0_s0.0_w0.0_e90.0.asc")
depth_n90_s0_w90_e180 <- raster("data/gebco_2020_ascii/gebco_2020_n90.0_s0.0_w90.0_e180.0.asc")

obj = c('depth_n90_s0_w180_e90', 'depth_n90_s0_w90_e0',
        'depth_n90_s0_w0_e90', 'depth_n90_s0_w90_e180',
        "depth_n0_s90_w180_e90", "depth_n0_s90_w90_e0",
        'depth_n0_s90_w0_e90', 'depth_n0_s90_w90_e180')
x.min = c(-180, -90, 0, 90, -180, -90, 0, 90)
x.max = c(-90, 0, 90, 180, -90, 0, 90, 180)
y.min = c(0, 0, 0, 0, -90, -90, -90, -90)
y.max = c(90, 90, 90, 90, 0, 0, 0, 0)

depth_files <- data.frame(obj) %>%
  mutate(xmin = x.min,
         xmax = x.max,
         ymin = y.min,
         ymax = y.max)
select_files_r <- raster(nrows = 2, ncols = 4, xmn = -180, xmx = 180,
                         ymn = -90, ymx = 90) %>%
  rasterToPolygons() %>%
  st_as_sf()


################################################################################
### Loop to correct for missing areas
################################################################################

seafloor_meow_deepsea_filled <- data.frame()
problems <- c()
```

```
for(h in 1:nrow(holes)) {
  print(h)

  hole_one <- holes[h,]

  # extract lat/long boundaries
  bbox_one <- st_bbox(hole_one)

  # create bounding box for the raster shapefile & polygons
  x_range <- abs(abs(bbox_one$xmax) - abs(bbox_one$xmin))
  y_range <- abs(abs(bbox_one$ymax) - abs(bbox_one$ymin))
  new_bbox <- as.vector(bbox_one)
  new_bbox[1] <- new_bbox[1] -0.1*x_range
  new_bbox[2] <- new_bbox[2] -0.1*y_range
  new_bbox[3] <- new_bbox[3] +0.1*x_range
  new_bbox[4] <- new_bbox[4] +0.1*y_range

  if(new_bbox[1]<(-180)){new_bbox[1] <- (-180)}
  if(new_bbox[2]<(-90)){new_bbox[2] <- (-90)}
  if(new_bbox[3]>180){new_bbox[3] <- 180}
  if(new_bbox[4]>90){new_bbox[4] <- 90}

  # get the depth raster(s) around that zone
  # intersect between the new_bbox and the select files
  new_bbox_r <- raster(nrow=1, ncol=1, xmn = new_bbox[1], xmx = new_bbox[3],
                       ymn = new_bbox[2], ymx = new_bbox[4])

  overlap_rr <- coverage_fraction(new_bbox_r, select_files_r)
  select_depth <- c()
  for(r in 1:8){
    if(values(overlap_rr[[r]])==1){select_depth[length(select_depth)+1] <- depth_files$obj[r]}
    if(values(overlap_rr[[r]])>0 && values(overlap_rr[[r]]<1)){select_depth[length(select_depth)+1] <- d
  }

  # extract the zone around the bbox from the raster
  if(length(select_depth)>0){
    if(length(select_depth)>1){
      for(k in 1:length(select_depth)){
        depth_k <- crop(get(select_depth[k]), y = extent(new_bbox[1],new_bbox[3],new_bbox[2],new_bbox[4]
        if(k==1){depth <- depth_k}
        else{depth <- merge(depth, depth_k)}
        rm(depth_k)
      }
    }
    if(length(select_depth)==1){
      depth <- crop(get(select_depth), y = extent(new_bbox[1],new_bbox[3],new_bbox[2],new_bbox[4]))
    }

    while((dim(depth)[1]*dim(depth)[2])<10){
      new_bbox[1] <- new_bbox[1] -0.3*x_range
      new_bbox[2] <- new_bbox[2] -0.3*y_range
      new_bbox[3] <- new_bbox[3] +0.3*x_range
      new_bbox[4] <- new_bbox[4] +0.3*y_range
```

```
  if(length(select_depth)>1){
    for(k in 1:length(select_depth)){
      depth_k <- crop(get(select_depth[k]), y = extent(new_bbox[1],new_bbox[3],new_bbox[2],new_bbox
      if(k==1){depth <- depth_k}
      if(k>1){depth <- merge(depth, depth_k)}
      rm(depth_k)
    }
  }
  if(length(select_depth)==1){
    depth <- crop(get(select_depth), y = extent(new_bbox[1],new_bbox[3],new_bbox[2],new_bbox[4]))
  }
}

# print final dimensions
print(dim(depth)[1]*dim(depth)[2])

# extract necessary polygons
depth_spdf <- as(depth, "SpatialPixelsDataFrame")
depth_df <- as.data.frame(depth_spdf)
colnames(depth_df) <- c("depth", "x", "y")

# select polygons from shapefile and depth raster
depth_pts <- data.frame(rasterToPoints(depth))
depth_pts_sf <- st_as_sf(depth_pts, coords = c("x","y"), crs = st_crs(seafloor_meow_deepsea))
select_polygons <- st_intersects(depth_pts_sf, seafloor_meow_deepsea) # that line takes time
select_polygons <- unique(unlist(select_polygons))
select_polygons <- seafloor_meow_deepsea[select_polygons,]

if(nrow(select_polygons)==1){
  new_poly <- select_polygons %>% st_drop_geometry()
  st_geometry(new_poly) <- st_geometry(holes[h,])
  seafloor_meow_deepsea_filled <- rbind(seafloor_meow_deepsea_filled, new_poly)
  save(seafloor_meow_deepsea_filled, file="outputs/holes/results_fill_holes.RData")
  save.image("outputs/holes/envt.RData")
  print("One polygon")
}

if(nrow(select_polygons)>1 && dim(depth)[2]==1){
  new_poly <- select_polygons[1,] %>% st_drop_geometry()
  st_geometry(new_poly) <- st_geometry(holes[h,])
  seafloor_meow_deepsea_filled <- rbind(seafloor_meow_deepsea_filled, new_poly)
  save(seafloor_meow_deepsea_filled, file="outputs/holes/results_fill_holes.RData")
  save.image("outputs/holes/envt.RData")
  print("Column polygon")}

if(nrow(select_polygons)>1 && dim(depth)[2]>1){
  select_polygons_merged <- st_union(select_polygons) # a bit long to run, only 3 polygons
  xx <- coverage_fraction(depth, select_polygons_merged)

  depths_empty <- as.data.frame(as(xx[[1]], "SpatialPixelsDataFrame")) %>%
    rename(coverage = layer) %>%
    filter(coverage<1) %>%
    left_join(depth_pts, by=c('x','y'))
```

```
depths_full <- as.data.frame(as(xx[[1]], "SpatialPixelsDataFrame")) %>%
  rename(coverage = layer) %>%
  filter(coverage==1) %>%
  left_join(depth_pts, by=c('x','y'))

# query will be depth_pts_empty transformed in 3D matrix
depth_pts_empty_df <- depths_empty %>%
  rename(z = names(depths_empty)[4]) %>%
  mutate(z = z) %>%
  dplyr::select(x,y,z)
depth_pts_empty_mat <- data.matrix(depth_pts_empty_df)

# target will be depth_pts transformed in 3D matrix
depth_pts_df <- depths_full %>%
  rename(z = names(depths_empty)[4]) %>%
  mutate(z = z) %>%
  dplyr::select(x,y,z)
depth_pts_mat <- data.matrix(depth_pts_df)

# 3D NNA from the R morpho package
# https://www.rdocumentation.org/packages/Morpho/versions/2.9/topics/mcNNindex
xx <- mcNNindex(target = depth_pts_mat, query = depth_pts_empty_mat, k=1)
depth_pts_empty_closest <- data.frame(cbind(depth_pts_empty_mat, xx)) %>%
  mutate(x_closest = NA,
         y_closest = NA,
         poly = NA_character_)

types <- c(select_polygons$ID)
pts <- st_as_sf(depth_pts_df, coords = c("x","y"), crs= st_crs(select_polygons))
pts$id_pt <- 1:nrow(pts)
tt <- unlist(st_intersects(pts, select_polygons, sparse=T))
pts <- st_join(pts, select_polygons) %>%
  st_drop_geometry()

depth_pts_empty_closest <- depth_pts_empty_closest %>%
  mutate(x_closest = depth_pts_empty_closest$x[depth_pts_empty_closest$V4],
         y_closest = depth_pts_empty_closest$y[depth_pts_empty_closest$V4],
         poly = types[tt[depth_pts_empty_closest$V4]])

# rasterize the closest points and aggregate a higher scale
# create new grid with lower resolution
# same for rasters
if(nrow(depth)<4 | ncol(depth)<4 ){new_r <- depth
} else {
  new_r <- aggregate(depth, fact = 4, fun = mean)
}

overlap_ri <- coverage_fraction(new_r, holes[h,])[[1]]
overlap_ri[overlap_ri==0] <- NA
overlap_ri[overlap_ri>0] <- 1
new_ri <- new_r*overlap_ri

new_poly <- rasterToPolygons(new_ri)
```

```
      new_poly <- st_as_sf(new_poly, crs = st_crs(seafloor_meow_deepsea_filled))
      new_poly$ID <- c(1:nrow(new_poly))

      depth_pts_empty_closest_sf <- st_as_sf(depth_pts_empty_closest, coords = c('x','y'),
                                             crs = st_crs(new_poly))
      new_poly_closest <- st_join(new_poly, depth_pts_empty_closest_sf, left=T) %>%
        st_drop_geometry() %>%
        group_by(ID, poly) %>% summarise(n=length(poly)) %>% slice_max(n, with_ties = FALSE)

      new_poly <- left_join(new_poly, new_poly_closest, by = "ID") %>%
        filter(n>0) %>%
        group_by(poly) %>%
        summarize(geometry = st_union(geometry))

      # get attributes from seafloor shapefile
      select_polygons <- select_polygons %>%
        filter(ID %in% new_poly$poly) %>%
        st_drop_geometry()
      new_poly <- new_poly %>%
        mutate(poly = as.numeric(as.vector(poly)))
      new_poly <- left_join(select_polygons, new_poly, by=c("ID"="poly"))
      new_poly <- st_as_sf(new_poly, crs = st_crs(seafloor_meow_deepsea))
      seafloor_meow_deepsea_filled <- rbind(seafloor_meow_deepsea_filled, new_poly)
      save(seafloor_meow_deepsea_filled, file="outputs/holes/results_fill_holes.RData")
      save.image("outputs/holes/envt.RData")
      print("Multi polygons")

      rm(new_poly_closest, new_r, new_ri, types, depth_pts_empty_closest,
         depth_pts_empty_df, depth_pts_empty_mat,
         depths_empty, depths_full,select_polygons_merged,
         xx, depth_pts_mat)

    }

    rm(depth_pts, depth_pts_sf, select_polygons, depth_spdf, depth_df, depth,
       overlap_rr, new_bbox_r, select_depth, new_bbox, x_range, y_range, hole_one,
       bbox_one)

  } else{problems[length(problems)+1] <- h
  print("length(select_depth) not positive")
  save.image("outputs/holes/envt.RData")}

}

save(seafloor_meow_deepsea_filled, file = "outputs/holes/seafloor_meow_deepsea_filled.RData")
```

- Step #5: finalize the shapefile

```
rm(list = ls())

### Libraries
library(sf)
sf::sf_use_s2(FALSE)
library(tidyverse)
```

```
library(ggplot2)
library(rgdal)
library(RColorBrewer)
library(raster)
library(geosphere)
library(fasterize)
library(smoothie)
library(exactextractr)
library(Morpho)
library(rgl)
library(units)
library(rnaturalearth)
library(rnaturalearthdata)
world <- ne_countries(scale = "medium", returnclass = "sf")


### load holes transformed into polygons with regions from e. ran on the HPC
load("outputs/holes/seafloor_meow_deepsea_filled_1_to_10.RData")
seafloor_meow_deepsea_final <- seafloor_meow_deepsea_filled
load("outputs/holes/seafloor_meow_deepsea_filled_11_to_200.RData")
seafloor_meow_deepsea_final <- rbind(seafloor_meow_deepsea_final, seafloor_meow_deepsea_filled)
load("outputs/holes/seafloor_meow_deepsea_filled_201_to_400.RData")
seafloor_meow_deepsea_final <- rbind(seafloor_meow_deepsea_final, seafloor_meow_deepsea_filled)
load("outputs/holes/seafloor_meow_deepsea_filled_401_to_600.RData")
seafloor_meow_deepsea_final <- rbind(seafloor_meow_deepsea_final, seafloor_meow_deepsea_filled)
load("outputs/holes/seafloor_meow_deepsea_filled_601_to_800.RData")
seafloor_meow_deepsea_final <- rbind(seafloor_meow_deepsea_final, seafloor_meow_deepsea_filled)
load("outputs/holes/seafloor_meow_deepsea_filled_801_to_1000.RData")
seafloor_meow_deepsea_final <- rbind(seafloor_meow_deepsea_final, seafloor_meow_deepsea_filled)
load("outputs/holes/seafloor_meow_deepsea_filled_1001_to_1200.RData")
seafloor_meow_deepsea_final <- rbind(seafloor_meow_deepsea_final, seafloor_meow_deepsea_filled)
load("outputs/holes/seafloor_meow_deepsea_filled_1201_to_1400.RData")
seafloor_meow_deepsea_final <- rbind(seafloor_meow_deepsea_final, seafloor_meow_deepsea_filled)
load("outputs/holes/seafloor_meow_deepsea_filled_1401_to_1600.RData")
seafloor_meow_deepsea_final <- rbind(seafloor_meow_deepsea_final, seafloor_meow_deepsea_filled)
load("outputs/holes/seafloor_meow_deepsea_filled_1601_to_1800.RData")
seafloor_meow_deepsea_final <- rbind(seafloor_meow_deepsea_final, seafloor_meow_deepsea_filled)
load("outputs/holes/seafloor_meow_deepsea_filled_1801_to_2000.RData")
seafloor_meow_deepsea_final <- rbind(seafloor_meow_deepsea_final, seafloor_meow_deepsea_filled)
load("outputs/holes/seafloor_meow_deepsea_filled_2001_to_2232.RData")
seafloor_meow_deepsea_final <- rbind(seafloor_meow_deepsea_final, seafloor_meow_deepsea_filled)


# load provinces_p7s3
seafloor_meow_deepsea <- st_read("outputs/hadal/provinces_p7s3_abyssal_corr.shp") %>%
  mutate(prov_id = ifelse(type == "hadal", had_id, prov_id),
         prov_n = ifelse(type == "hadal", had_n, prov_n),
         type_id = paste0(type,"-",prov_id)) %>%
  dplyr::select(-had_id, -had_n, -eco_id, -eco_n, -rlm_n, -rlm_id, -ID)

seafloor_meow_deepsea_final <- seafloor_meow_deepsea_final  %>%
  mutate(prov_id = ifelse(type == "hadal", had_id, prov_id),
         prov_n = ifelse(type == "hadal", had_n, prov_n),
```

```
         type_id = paste0(type,"-",prov_id)) %>%
  dplyr::select(-had_id, -had_n, -eco_id, -eco_n, -rlm_n, -rlm_id, -ID)

# creation of the bpow shapefile
bpow <- rbind(seafloor_meow_deepsea, seafloor_meow_deepsea_final) %>%
  mutate(source = case_when(type == "coastal" ~ "Spalding et al., (2007)",
                            type %in% c("bathyal","abyssal") ~ "Watling et al., (2013)",
                            type == "hadal" ~ "adapted from Belyaev (1989)")) %>%
  group_by(type, type_id, prov_n, prov_id, source) %>%
  summarize(geometry = st_union(geometry)) %>%
  ungroup()
bpow$ID <- 1:100

xx <- st_is_valid(bpow) # all true so shapefile valid!

### problem with 20 and 24
new_1 <- bpow[20,] %>%
  st_cast(geometry, to = "POLYGON")
pb_1 <- bpow[20,] %>%
  st_cast(geometry, to = "POLYGON") %>%
  st_centroid() %>%
  st_coordinates()
new_1 <- cbind(new_1, pb_1) %>%
  filter(!(X<0 & X>(-90))) %>%
  dplyr::select(-X, -Y) %>%
  group_by(type, type_id, prov_n, prov_id, source, ID) %>%
  summarize(geometry = st_union(geometry)) %>%
  ungroup()
append_1 <- st_difference(bpow[20,],new_1)
st_geometry(bpow[20,]) <- st_geometry(new_1)

new_2 <- bpow[24,] %>%
  st_cast(geometry, to = "POLYGON")
pb_2 <- bpow[24,] %>%
  st_cast(geometry, to = "POLYGON") %>%
  st_centroid() %>%
  st_coordinates()
new_2 <- cbind(new_2, pb_2) %>%
  filter(Y<0) %>%
  dplyr::select(-X, -Y) %>%
  group_by(type, type_id, prov_n, prov_id, source, ID) %>%
  summarize(geometry = st_union(geometry)) %>%
  ungroup()
append_2 <- st_difference(bpow[24,],new_2)
st_geometry(bpow[24,]) <- st_geometry(new_2)

append_1 <- append_1 %>%
  dplyr::select(type, type_id, prov_n, prov_id, source, ID)
append_2 <- append_2 %>%
  dplyr::select(type, type_id, prov_n, prov_id, source, ID)
geom_23 <- st_union(append_1,append_2)
geom_23 <- st_union(geom_23, bpow[23,])
st_geometry(bpow[23,]) <- st_geometry(geom_23)
```

```
### check and plot all regions
for(i in 1:100){
  png(filename = paste0("figures/all_regions/",i,"_",bpow$type[i],".png"),
      width = 16*200, height = 10*200, res = 200)
  print(ggplot(world) + geom_sf(fill = "grey", color = NA) +
          geom_sf(data = bpow[i,], fill = "blue", color = "black") +
          ggtitle(bpow$prov_n[i]))
  dev.off()
}


st_write(obj = bpow, dsn = "outputs/bpow/bpow_p8s5_abyssal_corr.shp",
         append = FALSE)
```

Part 9: Quality checks

Performed on ArcGIS Pro

- Step #1: Remove unnecessary parts of the geometry

```
-ArcGIS: Pariwise Dissolve, input = bpow_s8p5, by=ID
-QGIS: Dissolve
```

```
# load libraries
library(sf)
library(tidyverse)
library(ggplot2)

bpow <- st_read(dsn = "/Users/auroremaureaud/Documents/bpow_p9_post_op.shp")

### A. Work on abyssal self intersections
bpow_validation <- st_is_valid(bpow, reason=TRUE, NA_on_exception = FALSE)
unique(bpow_validation) # 189 abyssal with self-ring intersection

bpow_corrected <- st_make_valid(bpow)

bpow_validation2 <- st_is_valid(bpow_corrected, reason=TRUE, NA_on_exception = FALSE)
unique(bpow_validation2) # all valid geometries

# bpow_corrected2 <- st_make_valid(bpow_corrected)
#
# bpow_validation3 <- st_is_valid(bpow_corrected2, reason=TRUE, NA_on_exception = FALSE)
# unique(bpow_validation3) # all valid geometries
#
# bpow_corrected3 <- st_make_valid(bpow_corrected2)
# bpow_validation4 <- st_is_valid(bpow_corrected3, reason = TRUE, NA_on_exception = FALSE)

st_write(bpow_corrected, dsn = "outputs/bpow/bpow_p9_abyssal_corr.shp",
         append=F)
```

+==========+
+----------+

- Step #2: Fix the boundaries of the layer

-ArcGIS:

+==========+
+----------+

Part 10: Remove land

Performed on ArcGIS Pro

-ArcGIS: Pairwise Erase, input = land, erase = bpow_s8p5 |
-QGIS: Difference with ne_10land layer from natural earth |

```
# load libraries
library(sf)
library(tidyverse)
library(ggplot2)


bpow <- st_read(dsn = "/Users/auroremaureaud/Documents/bpow_p10.shp")
attributes <- st_read("outputs/bpow/bpow_p8s5_abyssal_corr.shp") %>%
  st_drop_geometry()

bpow <- left_join(bpow, attributes, by = "ID")

st_write(bpow, dsn = "outputs/bpow/bpow_p10_attributes.shp",
         append=F)
```

References

Belyaev, G. M. 1989. *Deep Sea Ocean Trenches and Their Fauna.* Nauka, Moscow.

Group, GEBCO Bathymetric Compilation. 2020. "The GEBCO_2020 Grid - a Continuous Terrain Model of the Global Oceans and Land." British Oceanographic Data Centre, National Oceanography Centre, NERC, UK. https://doi.org/10.5285/A29C5465-B138-234D-E053-6C86ABC040B9.

Jamieson, Alan J., Toyonobu Fujii, Daniel J. Mayor, Martin Solan, and Imants G. Priede. 2010. "Hadal Trenches: The Ecology of the Deepest Places on Earth." *Trends in Ecology & Evolution* 25 (3): 190–97. https://doi.org/10.1016/j.tree.2009.09.009.

Spalding, Mark D., Helen E. Fox, Gerald R. Allen, Nick Davidson, Zach A. Ferdaña, Max Finlayson, Benjamin S. Halpern, et al. 2007. "Marine Ecoregions of the World: A Bioregionalization of Coastal and Shelf Areas." *BioScience* 57 (7): 573–83. https://doi.org/10.1641/B570707.

Watling, Les, John Guinotte, Malcolm R. Clark, and Craig R. Smith. 2013. "A Proposed Biogeography of the Deep Ocean Floor." *Progress in Oceanography* 111 (April): 91–112. https://doi.org/10.1016/j.pocean.2012.11.003.