Sensitivity Raster versus Projected Shapefile

Zoë Kitchel

January, 2022

##Here, we compare using a raster to using a projected shapefile to check for differences in area calculations.

```
library(raster)
library(sf)
library(ncdf4)
library(rmapshaper)
library(tidyverse)
library(diptest)
library(moments)
library(viridis) #colors
library(data.table)
library(hydroTSM) #hypsometric curves
library(gridExtra)
library(maptools)
library(rgdal)
library(rgeos)
library(SpaDES)
library(rnaturalearth)
library(rnaturalearthdata)
library(equate)

etopo_shelf_df <- readRDS("raw_data/etopo_shelf_df.rds")
#bring in bathymetry data frame for shelf regions

#LMEs
LME_spdf <- readOGR("raw_data/LME66/LMEs66.shp") #spatial points data frame with all 66 LMEs


#convert to equal area projection
  #The Lambert azimuthal equal-area projection is a particular mapping from a sphere to a disk. It accu
equalareaprojection <- crs(" +proj=laea ")
```

Convert data frame to raster for bathymetry layer.

```
etopo_shelf_raster <- rasterFromXYZ(etopo_shelf_df, crs = crs(LME_spdf))
```

We will test regions that are likely to give us issues because they include high latitudes.

- For degree shifts, I will use East Atlantic Ocean
- For LME depth profiles, I will use High Arctic Canada/Greenland (LME 66)

###Degree Shifts

Eastern Atlantic

LMEs to include -19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 58, 59, 60, 62

Merge LMEs included in Eastern Atlantic coastline

```
east_atl <- c(19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 58, 59, 60, 62)

east_atl_spdf <- LME_spdf[(LME_spdf$LME_NUMBER) %in% east_atl,]

#get rid of buffer for east atl as well to allow for union

east_atl_spdf_nobuf <- gBuffer(east_atl_spdf, byid=TRUE, width=0)
```

```
## Warning in gBuffer(east_atl_spdf, byid = TRUE, width = 0): Spatial object is not
## projected; GEOS expects planar coordinates
```

```
#dissolve polygons into one along coastline
east_atl_spdf_nobuf_agg <- gUnaryUnion(east_atl_spdf_nobuf)
```

Extract bathymetry data from polygon only to make sure we're limiting to shelf regions shallower than 2000 meters

```
#crop bathymetry layer to LME subset (continental shelf habitat in LMEs)
raster_extent <- crop(etopo_shelf_raster, extent(east_atl_spdf_nobuf_agg))

#which areas of raster fall within borders?
east_atl_spdf_nobuf_agg_mask <- mask(raster_extent, east_atl_spdf_nobuf_agg)

#reclassify so that raster only has values of 1, because the only purpose of pulling in bathymetry in t

east_atl_spdf_nobuf_mask_1s <- reclassify(east_atl_spdf_nobuf_agg_mask, c(-Inf, Inf, 1))
```

###How to calculate area? ####Three options

- raster::area() Raster objects: Compute the approximate surface area of cells in an unprojected (longitude/latitude) Raster object. It is an approximation because area is computed as the height (latitudinal span) of a cell (which is constant among all cells) times the width (longitudinal span) in the (latitudinal) middle of a cell. The width is smaller at the poleward side than at the equator-ward side of a cell. This variation is greatest near the poles and the values are thus not very precise for very high latitudes. If x is a Raster* object: RasterLayer or RasterBrick. Cell values represent the size of the cell in km2, or the relative size if weights=TRUE
- raster::area() SpatialPolygons: Compute the area of the spatial features. Works for both planar and angular (lon/lat) coordinate reference systems. If x is a SpatialPolygons* object: area of each spatial object in squared meters if the CRS is longitude/latitude, or in squared map units (typically meter)
- rgeos::gArea Returns the area of the geometry in the units of the current projection. By definition non-[MULTI]POLYGON geometries have an area of 0. The area of a POLYGON is the area of its shell less the area of any holes. Note that this value may be different from the area slot of the Polygons class as this value does not subtract the area of any holes in the geometry.

#####Now, we will split each coastline raster into latitudinal bins of 1 degree

Eastern Atlantic

name of raster = east_atl_spdf_nobuf_mask_1s

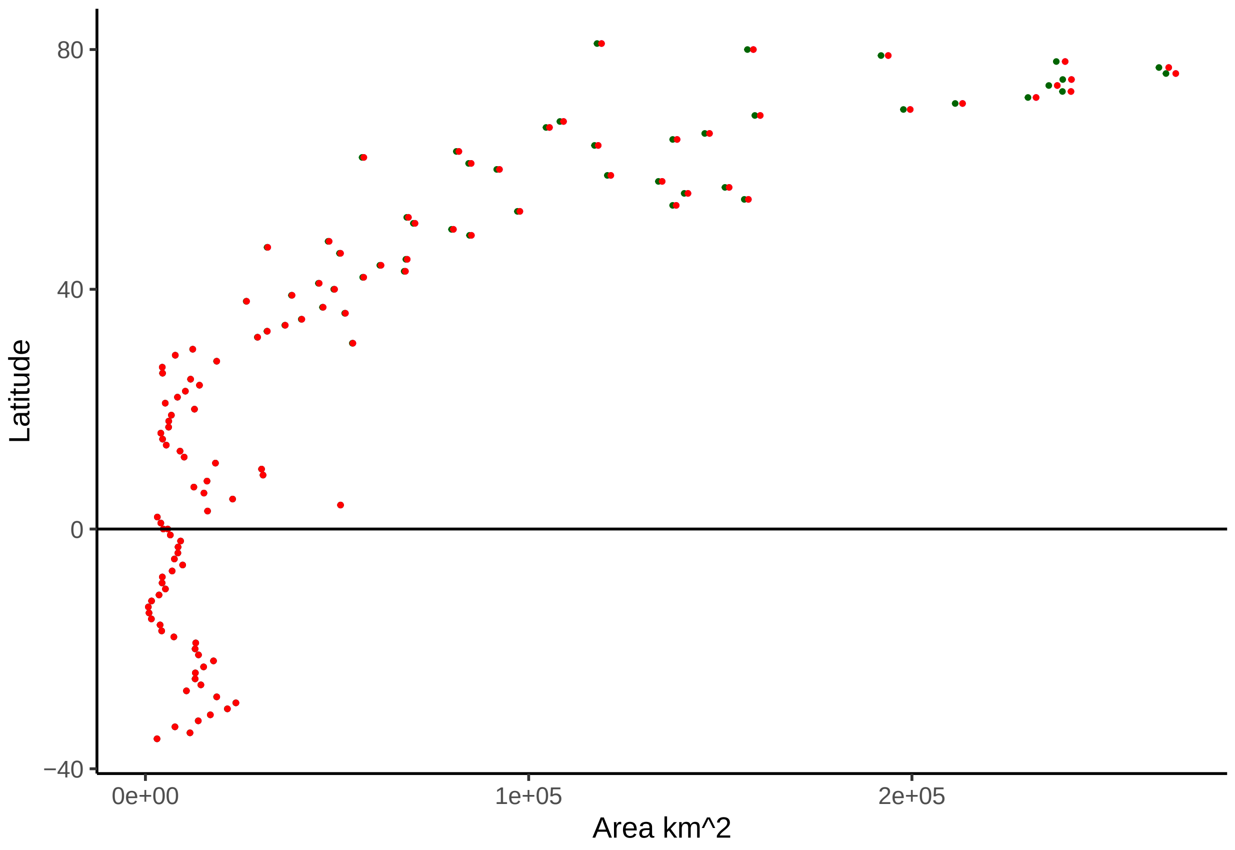
```
#split into northern hemisphere and southern hemisphere
north_extent <- c(xmin(east_atl_spdf_nobuf_mask_1s), xmax(east_atl_spdf_nobuf_mask_1s), 0, ymax(east_at
south_extent <- c(xmin(east_atl_spdf_nobuf_mask_1s), xmax(east_atl_spdf_nobuf_mask_1s), ymin(east_atl_s

#crop east_atl raster above and below 0
east_atl_spdf_shift_agg_north <- crop(east_atl_spdf_nobuf_mask_1s, extent(north_extent))
```

```
east_atl_spdf_shift_agg_south <- crop(east_atl_spdf_nobuf_mask_1s, extent(south_extent))

#all 1 degree  latitude sections for east atlantic
east_atl_north_latitudes <- seq(0, ymax(east_atl_spdf_nobuf_mask_1s), by = 1)
east_atl_south_latitudes <- seq(0, ymin(east_atl_spdf_nobuf_mask_1s), by = -1)
```

Now, use loop to populate data table with area values

```
#setup data table to populate in loop, subtracting because there is one fewer bin than latitude #s
east_atl_shelf_areas <- as.data.table(matrix(nrow = (length(east_atl_north_latitudes)-1+length(east_atl_

east_atl_shelf_areas[, latitude_start := as.numeric(V1)][, latitude_end := as.numeric(V1)][, area_raste

#loop for north
for (i in 1:(length(east_atl_north_latitudes)-1)) {
  #setting up extent for slicing by min and max longitudes, and i to i+1 latitudes
  north_extent <- c(xmin(east_atl_spdf_nobuf_mask_1s), xmax(east_atl_spdf_nobuf_mask_1s), east_atl_nort

  #crop raster segement based on bin extent
  segment_north <- crop(east_atl_spdf_nobuf_mask_1s, extent(north_extent))


  #populate data table with latitudinal bin
  east_atl_shelf_areas[i, "latitude_start"] <- east_atl_north_latitudes[i]
  east_atl_shelf_areas[i, "latitude_end"] <- east_atl_north_latitudes[i+1]

  if(all(is.na(values(segment_north)))) { #if there's no shelf area within a bin, all area = 0

  east_atl_shelf_areas[i, "area_equalareaproj"] <- 0
  east_atl_shelf_areas[i, "area_rasterarea"] <- 0
  east_atl_shelf_areas[i, "area_rgeos_gArea"] <- 0


  print(i)

  } else { #if there is shelf area within the bin, calculate area of slice

    #raster area calculation
      #get sizes of all cells in raster [km2]
    cell_size_raster<-area(segment_north, na.rm=TRUE, weights=FALSE)

    #delete NAs from vector of all raster cells
    cell_size_raster<-cell_size_raster[!is.na(segment_north)]

    #sum all values of cell sizes
    segment_area_raster <- sum(cell_size_raster)

  #populate data table with raster area
  east_atl_shelf_areas[i, "area_rasterarea"] <- segment_area_raster

  #convert to spatial polygons to check area calculations

  #convert segment from raster to polygon, each cell from the raster is an independent polygon, (dissol
  segment_north.sp <- rasterToPolygons(segment_north, dissolve = T)
```

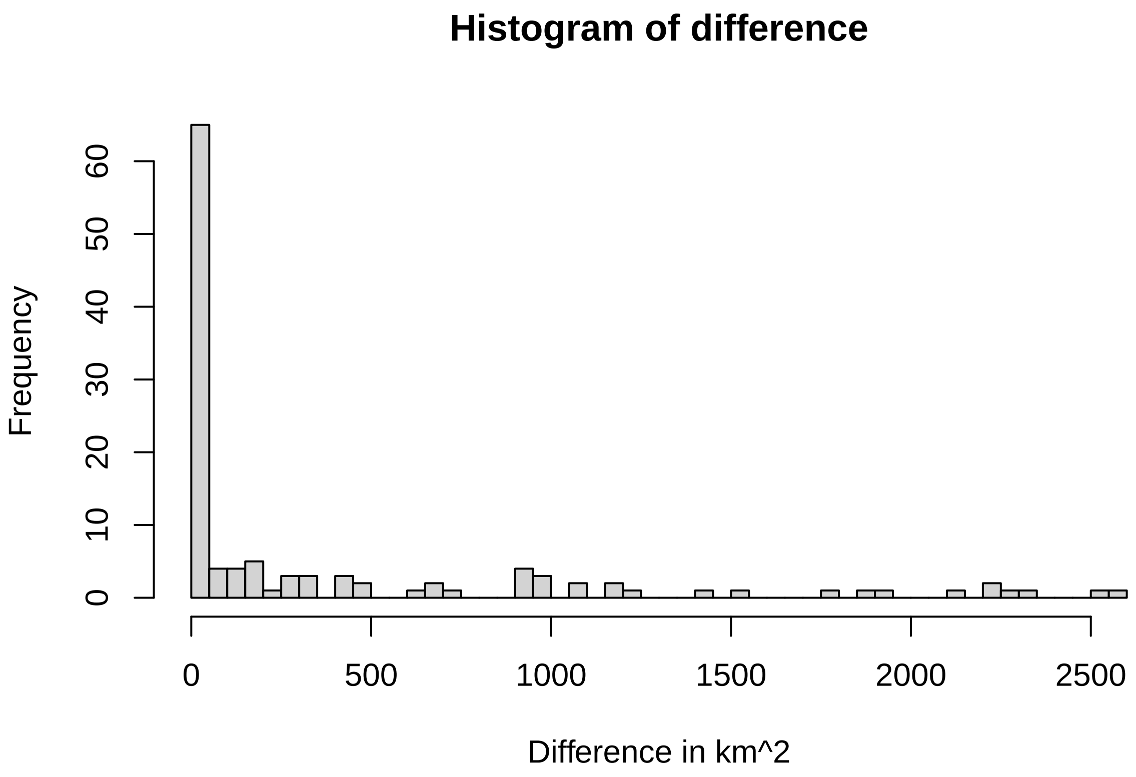
```r
  # NB: If x is a SpatialPolygons* object: area of each spatial object in squared meters if the CRS is

    #project to equal earth area projection
  segment_north.sp.EA <- spTransform(segment_north.sp, CRSobj = equalareaprojection)

  #calculate area of the entire spatial object in m^2
    polygon_size.sp <- area(segment_north.sp.EA)

    #convert from m^2 to km^2
    segment_area_equalarea <- polygon_size.sp/1e6

    #populate data table with polygon area using raster calculation
  east_atl_shelf_areas[i, "area_equalareaproj"] <- segment_area_equalarea

  #and then also use rgeos::gArea

  area_rgeos_gArea <- gArea(segment_north.sp.EA)/1e6

  #populate data table with polygon area using regeos calculation
  east_atl_shelf_areas[i, "area_rgeos_gArea"] <-   area_rgeos_gArea

  print(i)
  }
}

#loop for south
for (i in 1:(length(east_atl_south_latitudes)-1)) {
  south_extent <- c(xmin(east_atl_spdf_nobuf_mask_1s), xmax(east_atl_spdf_nobuf_mask_1s), east_atl_sout

  #raster segment
  segment_south <- crop(east_atl_spdf_nobuf_mask_1s, extent(south_extent))

  #add latitude bin info to data table
    east_atl_shelf_areas[i+(length(east_atl_north_latitudes)-1), "latitude_start"] <- east_atl_south_la
  east_atl_shelf_areas[i+(length(east_atl_north_latitudes)-1), "latitude_end"] <- east_atl_south_latitu


  if(all(is.na(values(segment_south)))) { #if there's no shelf area within a bin, meaning there's no sh

  east_atl_shelf_areas[i+(length(east_atl_north_latitudes)-1), "area_equalareaproj"] <- 0
  east_atl_shelf_areas[i+(length(east_atl_north_latitudes)-1), "area_rasterarea"] <- 0
    east_atl_shelf_areas[i+(length(east_atl_north_latitudes)-1), "area_rgeos_gArea"] <- 0

  print(i)

  } else {

    #raster area calculation
      #get sizes of all cells in raster [km2]
    cell_size_raster<-area(segment_south, na.rm=TRUE, weights=FALSE)

    #delete NAs from vector of all raster cells
    cell_size_raster<-cell_size_raster[!is.na(segment_south)]
    #compute area of all cells in geo_raster

    segment_area_raster <- sum(cell_size_raster)

  #populate data table with raster area
  east_atl_shelf_areas[i+(length(east_atl_north_latitudes)-1), "area_rasterarea"] <- segment_area_raste


  #convert to spatial polygons to check area calculations

  #convert segment from raster to polygon, each cell from the raster is an independent polygon, (dissol
  segment_south.sp <- rasterToPolygons(segment_south, dissolve = T)

    #project to equal earth area projection
  segment_south.sp.EA <- spTransform(segment_south.sp, CRSobj = equalareaprojection)

  #calculate area of the spatial object in m^2
    polygon_size.sp <- area(segment_south.sp.EA)

    #convert from m^2 to km^2
    segment_area_equalarea <- polygon_size.sp/1e6

    #populate data table with area of polygon from raster::area function
  east_atl_shelf_areas[i+(length(east_atl_north_latitudes)-1), "area_equalareaproj"] <- segment_area_equ

  #and then plain and simple also using rgeos::gArea

  area_rgeos_gArea <- gArea(segment_south.sp.EA)/1e6

  #populate data table from rgeos area calculation for projected polygon
  east_atl_shelf_areas[i+(length(east_atl_north_latitudes)-1), "area_rgeos_gArea"] <-   area_rgeos_gArea

  print(i)
  }
}

#compare raster:area calculation, to equal area still using raster::area function, to rgeos::gArea func

ggplot(data = east_atl_shelf_areas) +
  geom_point(aes(x = latitude_start, y = area_rgeos_gArea), color = "purple", size = 0.5) +
  geom_point(aes(x = latitude_start, y = area_rasterarea), color = "darkgreen", size = 0.5) +
  geom_point(aes(x = latitude_start, y = area_equalareaproj), color = "red", size = 0.5) +
  labs(x = "Latitude", y = "Area km^2") +
  coord_flip() +
  geom_vline(xintercept = 0) +
  theme_classic()
```



```
cor(east_atl_shelf_areas[,3:5], use = "complete.obs")
```

What's the maximum difference between polygon generated area and raster generated area?

```
east_atl_shelf_areas[, difference := abs(area_equalareaproj-area_rasterarea)]

east_atl_shelf_areas[,hist(difference, breaks = 50, xlab = "Difference in km^2")]
```

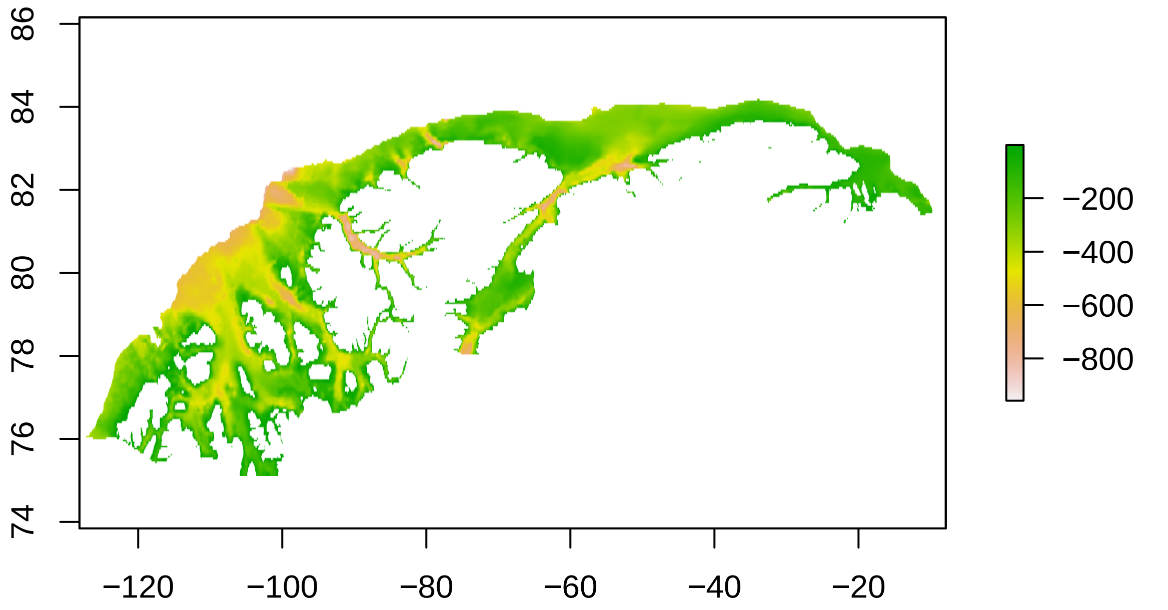


###LME Depth Profiles

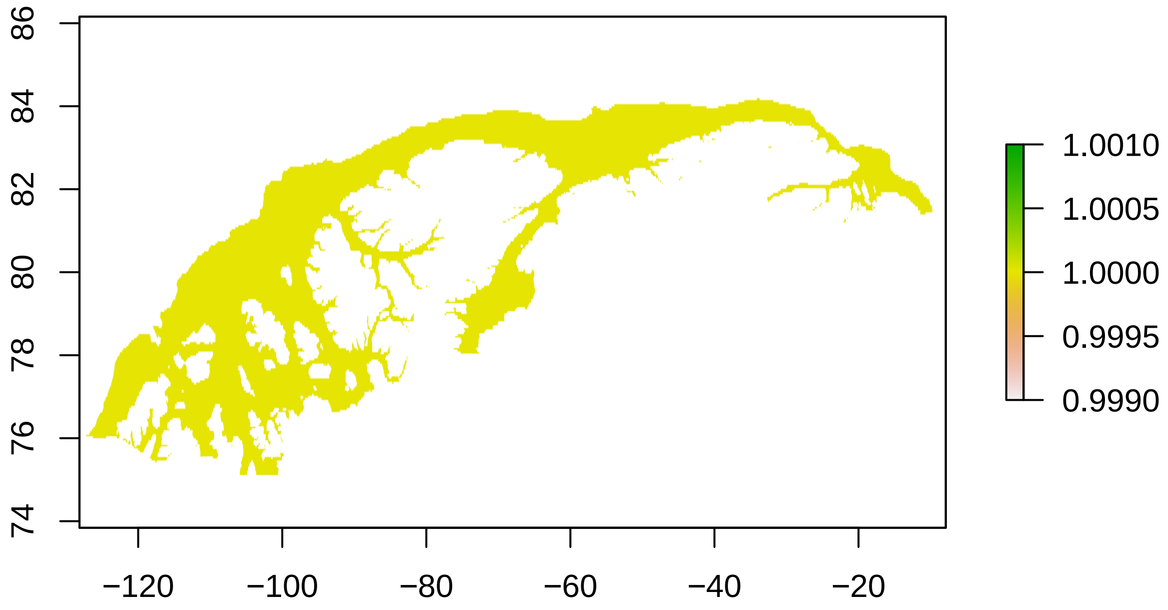
For LME depth profiles, I will use High Arctic Canada/Greenland (LME = 66)

```
 LME_high_arctic <- LME_spdf[LME_spdf@data$LME_NUMBER == 66,]

  #clip raster to LME
  LME_high_arctic_extent <- crop(etopo_shelf_raster, extent(LME_high_arctic))

  #which areas of raster fall within borders?
  LME_high_arctic_mask <- mask(LME_high_arctic_extent, LME_high_arctic)

  LME_high_arctic_mask_1s <- reclassify(LME_high_arctic_mask, c(-Inf, Inf, 1)) #this raster is only 1s

plot(LME_high_arctic)
```

```
plot(LME_high_arctic_mask)
```



```
plot(LME_high_arctic_mask_1s)
```



#####Next, I need to construct hypsometric plots (how does area available vary by depth).

Apply `raster::area` directly to list of raster values

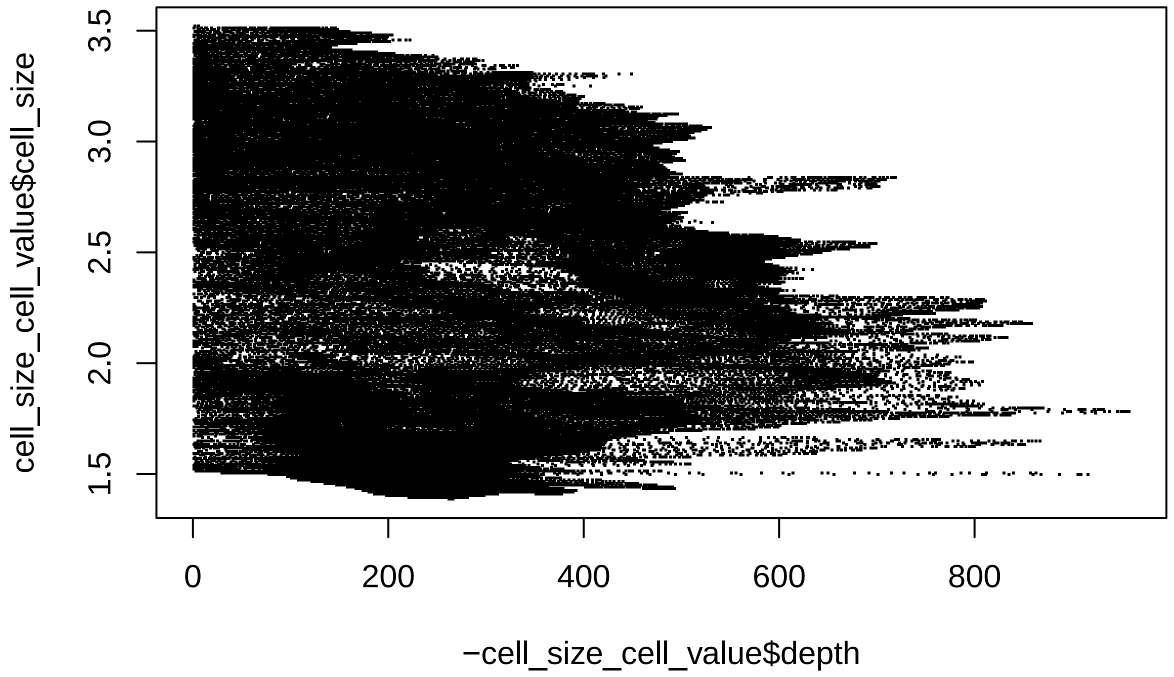
```
#list of values within raster
  high_arctic_bathy_depth_list <- getValues(LME_high_arctic_mask)
  high_arctic_bathy_depth_list <- high_arctic_bathy_depth_list[!is.na(high_arctic_bathy_depth_list)] #g

 #get sizes of all cells in raster [km2] (vector of size of all cells)
    cell_size_high_arctic <-area(LME_high_arctic_mask, na.rm=TRUE, weights=FALSE)

        #delete NAs from vector of all raster cells
        cell_size_high_arctic<-cell_size_high_arctic[!is.na(cell_size_high_arctic)]
    #compute total area [km2] of all cells in geo_raster
   high_arctic_raster_area<-sum(cell_size_high_arctic)

   #467587.9 km^2
```

Assign specific sizes to each grid cell

```
#cell_size_high_arctic # cell sizes
#high_arctic_bathy_depth_list # cell values

cell_size_cell_value <- data.table(cell_size = cell_size_high_arctic, depth = high_arctic_bathy_depth_li

cor(cell_size_cell_value$depth, cell_size_cell_value$cell_size) #Not super correlated
```

```
## [1] 0.09480613
```

```
plot(-cell_size_cell_value$depth, cell_size_cell_value$cell_size, pch = ".") #more likely to have small
```



Sum area across cells with same depth

```
cell_size_cell_value[,sum_area := sum(cell_size), by = depth]

#reduce to frequency table
freq_table_raster <- unique(cell_size_cell_value, by = c("depth", "sum_area"))

freq_table_raster[,cell_size := NULL] #remove cell size column
freq_table_raster[,depth := -depth] #make depth positive

#plot this frequency table
raster_areabycell_plot <- ggplot(data = freq_table_raster, aes(x = depth, y = sum_area, fill = depth)) +
    geom_col(width = 5) +
#    geom_smooth(method = "gam", se = F, color = "black", size = 1) +
    theme_classic() +
    labs(x = "Depth (m)", y = expression(paste("Area (", km^{2},")")), fill = "Depth (m)") +
    scale_y_continuous(expand = c(0, 0)) +
    scale_x_continuous(expand = c(0,0)) +
    scale_fill_gradientn(colors = rev(viridis(5)), limits = c(0, 2000)) +
    guides(position = "bottom", fill = guide_colorbar(reverse = T))
```

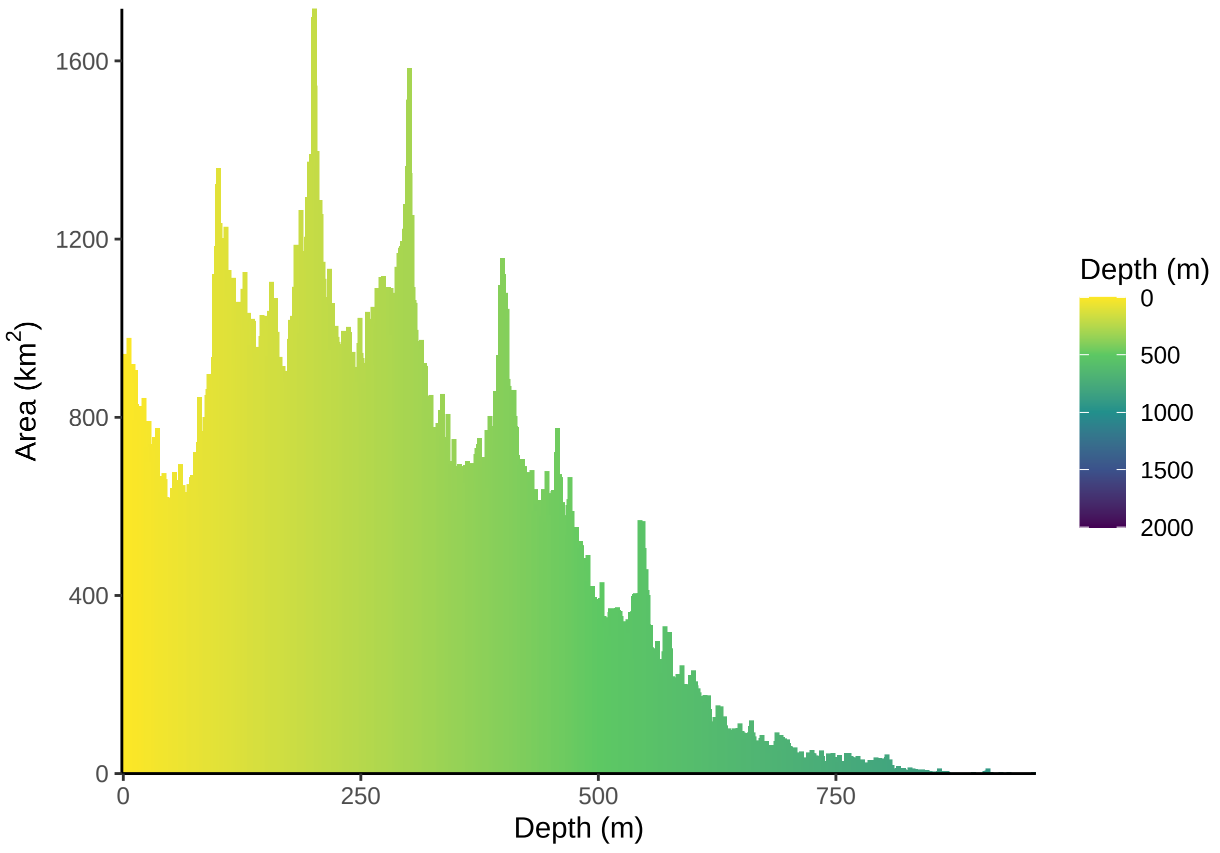
Alternatively, we convert to polygon for equal area projection and then produce frequency table

```
  #raster back to polygon
  LME_high_arctic.sp <- rasterToPolygons(LME_high_arctic_mask, dissolve = T) #dissolve means all same v

  #transform to equal area projection
  LME_high_arctic.EA <- spTransform(LME_high_arctic.sp, equalareaprojection)

#list of values within polygon
LME_high_arctic.EA@data$depth <- -(LME_high_arctic.EA@data$z) #creating depth column in m

LME_high_arctic.EA@data$area_m <- area(LME_high_arctic.EA, dissolve = T) #calculating area in m^2 of ea

LME_high_arctic.EA@data$area_km <- LME_high_arctic.EA@data$area_m/1e6 #convert to km^2 (same units as r

freq_table_polygon <- as.data.table(LME_high_arctic.EA@data)

polygon_plot <- ggplot(data = freq_table_polygon, aes(x = depth, y = area_km, fill = depth)) +
    geom_col(width = 5) +
#     geom_smooth(method = "gam", se = F, color = "black", size = 1) +
    theme_classic() +
    labs(x = "Depth (m)", y = expression(paste("Area (", km^{2},")")), fill = "Depth (m)") +
    scale_y_continuous(expand = c(0, 0)) +
    scale_x_continuous(expand = c(0,0)) +
    scale_fill_gradientn(colors = rev(viridis(5)), limits = c(0, 2000)) +
    guides(position = "bottom", fill = guide_colorbar(reverse = T))
```

Let's compare the two graphs visually

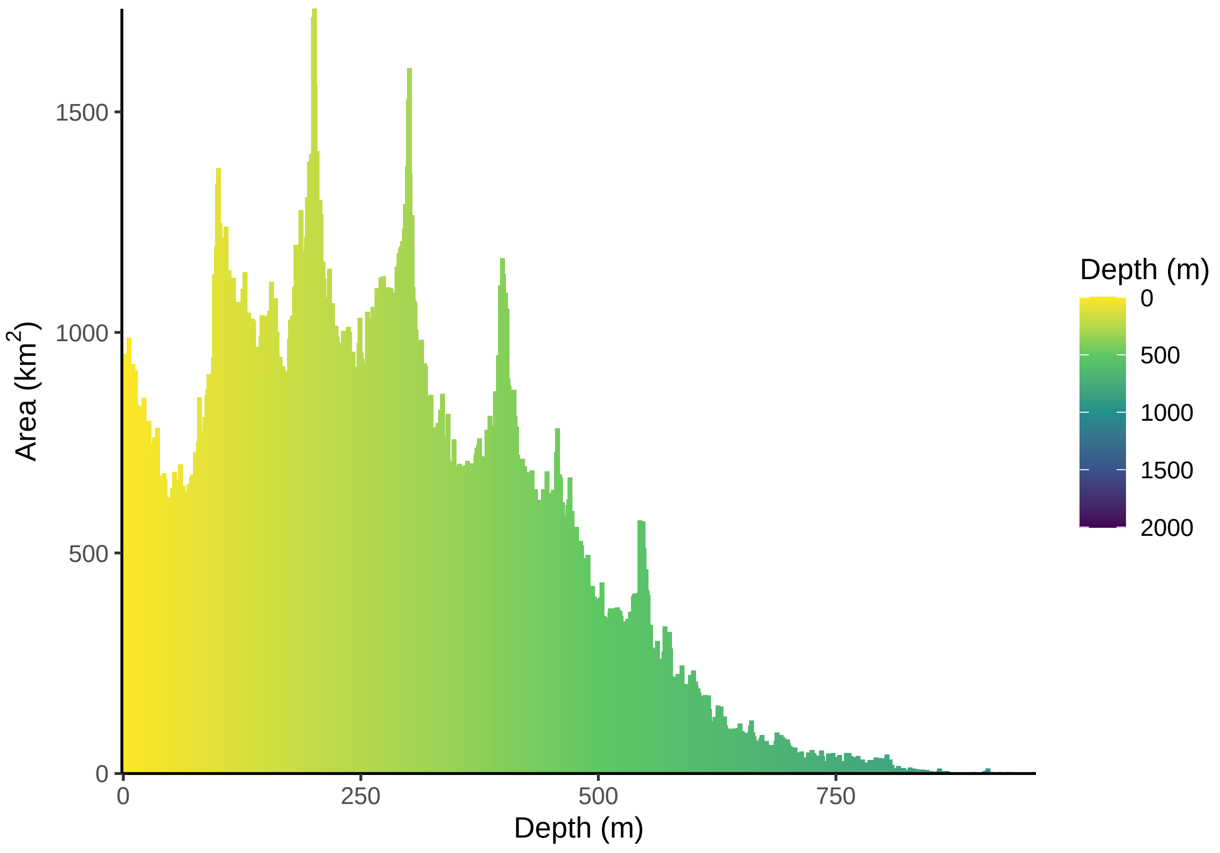
```
raster_areabycell_plot
```

```
## Warning: position_stack requires non-overlapping x intervals
```



```
polygon_plot
```

```
## Warning: position_stack requires non-overlapping x intervals
```



```
#identical shape, slightly different values
```

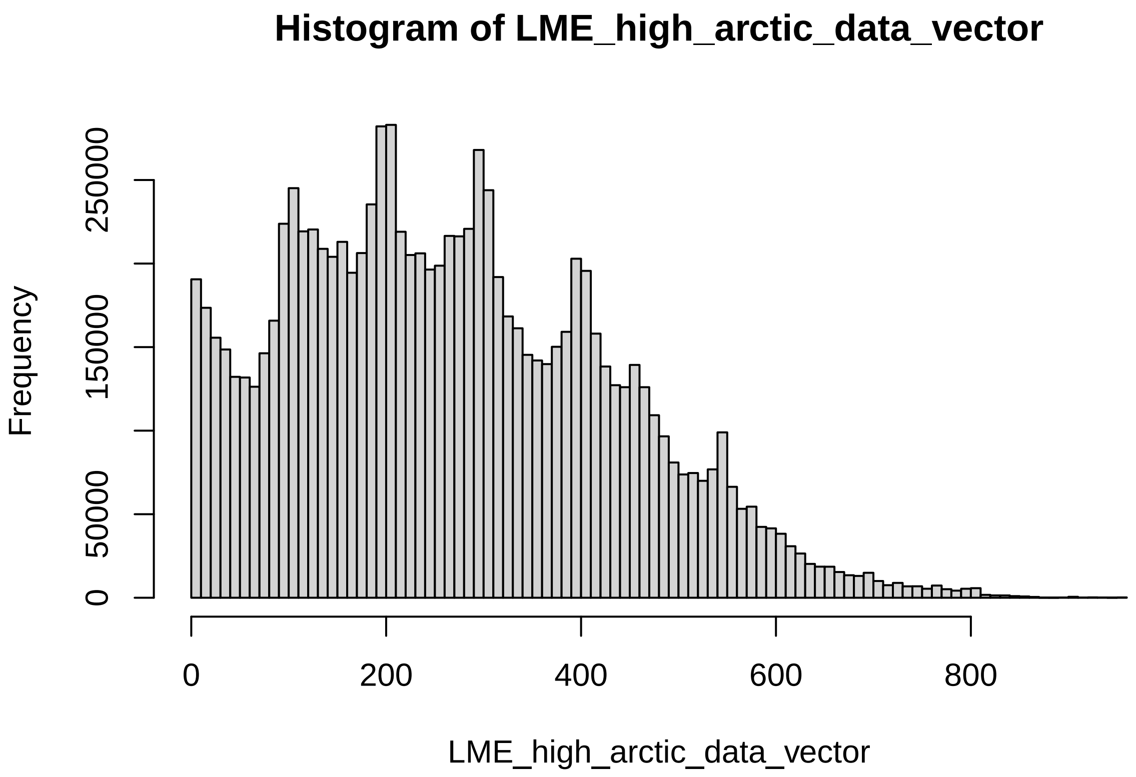
Visually, the shape does not change

But, what about the statistics, and final classification?

For raster taking into consideration bias correction. The output is a frequency table, therefore, in order to calculate modality, we need to generate a vector of values from the frequency table.

For skew etc., we can use the equate package in r to apply calculations directly to frequency table.

```
freq_table_raster[,prop_area :=  sum_area/sum(sum_area)] #calculate proportional area

freq_table_raster[,area_rounddown := round(prop_area * 10000000, 0)]

#empty vector
LME_high_arctic_data_vector <- vector()

#populate vector
for (i in 1:nrow(freq_table_raster)){
add <- rep(freq_table_raster[i,depth], freq_table_raster[i,area_rounddown])
LME_high_arctic_data_vector <- append(LME_high_arctic_data_vector, add, after = length(LME_high_arctic_

}

#plotting
hist(LME_high_arctic_data_vector, 100)
```



Perform calculations on raster.

```
freq_table_unbiased_raster.ft <- as.freqtab(freq_table_raster[,1:2])

dip.test_unbiased_raster <- dip.test(LME_high_arctic_data_vector, simulate.p.value = TRUE, B = 2000)
#this takes a while, probably more than 30 minutes
  p.value_dip.test_unbiased_raster <- dip.test_unbiased_raster$p.value
  skew_dip.test_unbiased_raster <- skew.freqtab(freq_table_unbiased_raster.ft)
  mean_unbiased_raster <- mean(freq_table_unbiased_raster.ft)
  max_depth_unbiased_raster <- max(freq_table_unbiased_raster.ft)
  median_depth_unbiased_raster <- median(freq_table_unbiased_raster.ft)


saveRDS(dip.test_unbiased_raster, file = "output/dip.test_unbiased_raster.rds")
```

Finally, repeat the preceding procedure but for projected polygon and associated depth frequency table.

```
freq_table_polygon[,prop_area :=  area_m/sum(area_m)]

freq_table_polygon[,area_rounddown := round(prop_area * 10000000, 0)]

#empty vector
LME_high_arctic_data_vector_polygon <- vector()

#populate vector
for (i in 1:nrow(freq_table_polygon)){
add <- rep(freq_table_polygon[i,depth], freq_table_polygon[i,area_rounddown])
LME_high_arctic_data_vector_polygon <- append(LME_high_arctic_data_vector_polygon, add, after = length(

}
```

Convert to actual frequency table using equate package, and calculate all statistics to describe hypsometric curve.

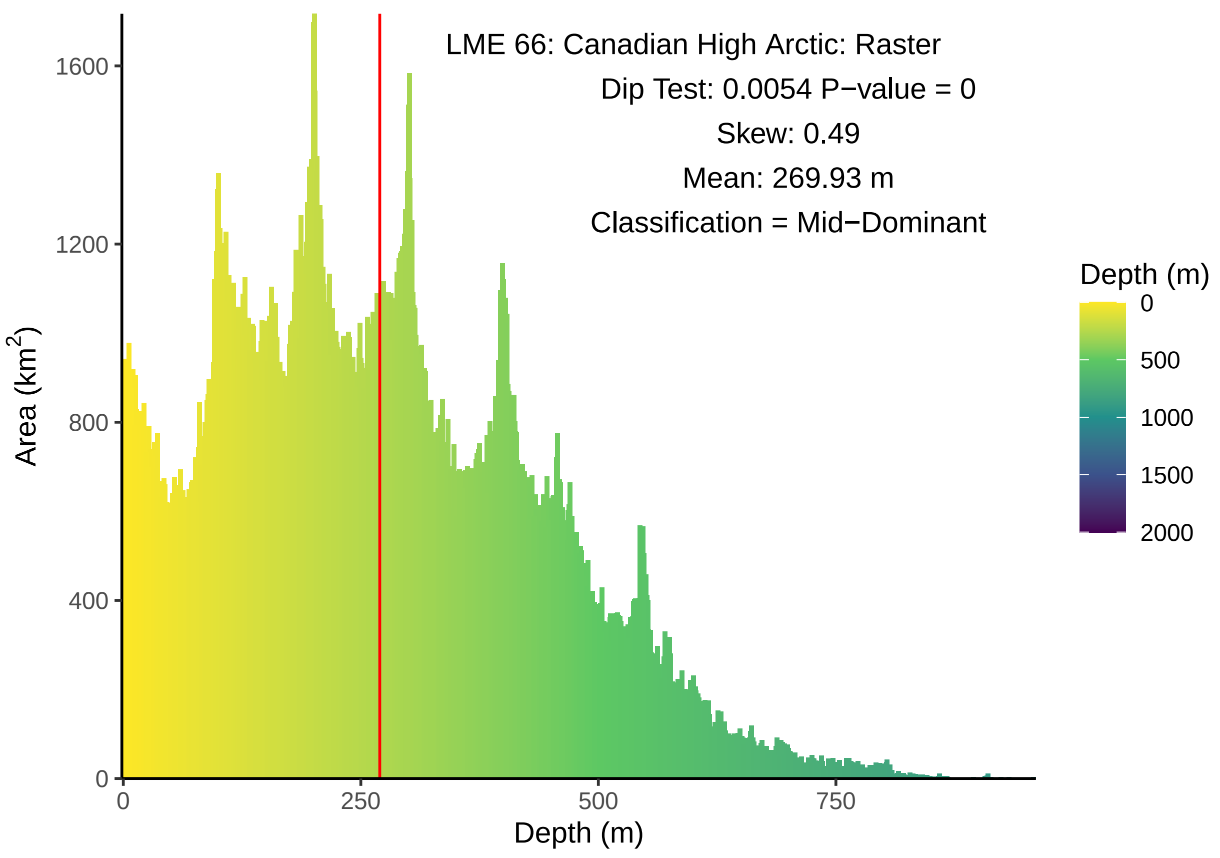
```
freq_table_polygon.ft <- as.freqtab(freq_table_polygon[,c(2,4)])

dip.test_polygon <- dip.test(LME_high_arctic_data_vector_polygon, simulate.p.value = TRUE, B = 2000)
save(dip.test_polygon, file = "higharctic_dip.test_polygon.RData")
  p.value_dip.test_polygon <- dip.test_polygon$p.value
  skew_dip.test_polygon <- skew.freqtab(freq_table_polygon.ft)
  mean_polygon <- mean(freq_table_polygon.ft)
  max_depth_polygon <- max(freq_table_polygon.ft)
  median_depth_polygon <- median(freq_table_polygon.ft)



saveRDS(dip.test_polygon, file = "output/dip.test_polygon.rds")
```

Re-plot both methods with all the statistics, and classification based on:

Multimodal: diptest value > 0.01, p-value <0.05; else Mid-Dominant: -1 < skew value < 1 Shallow Dominant: skew value > 1 Deep Dominant: skew value < -1

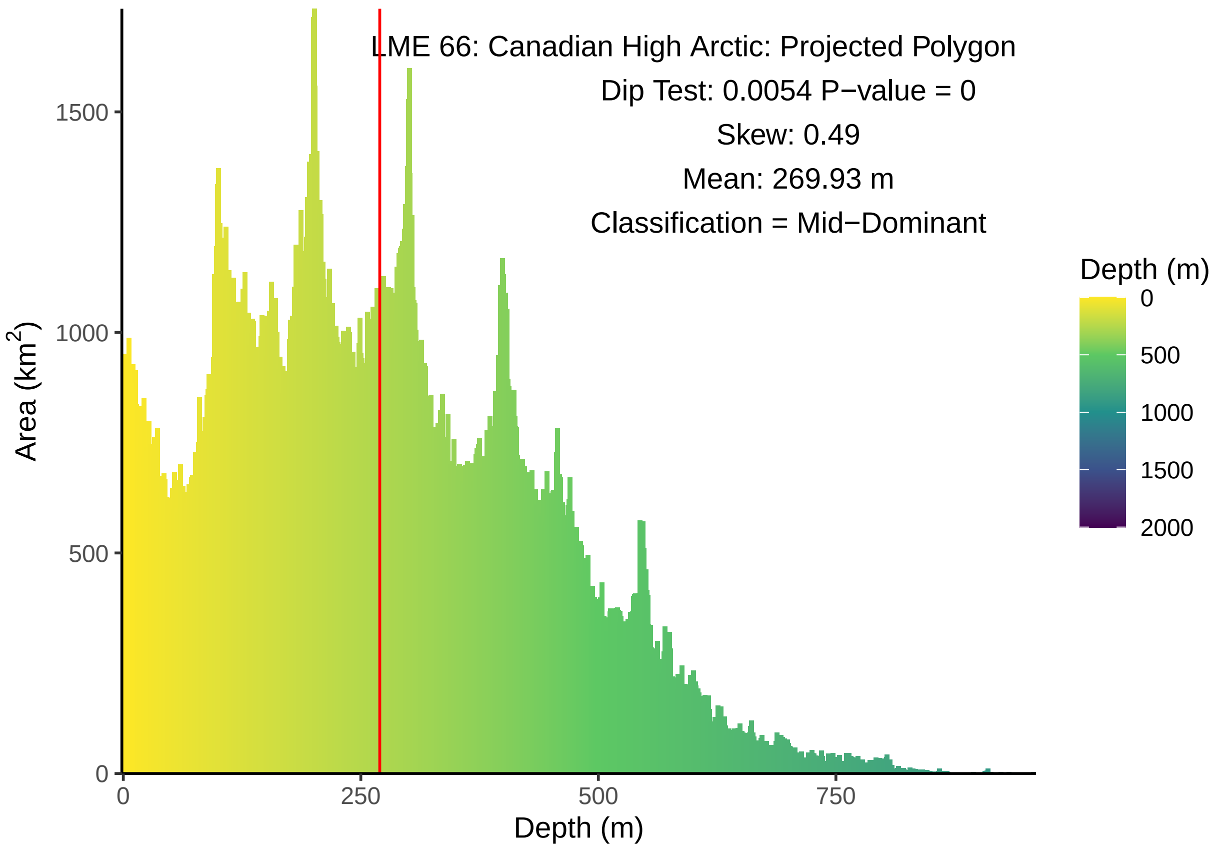
```
#plot for raster
raster_areabycell_plot +
annotate("text", x = 600, y = 1650, label = "LME 66: Canadian High Arctic: Raster") +
  annotate("text", x = 700, y = 1550, label = paste0("Dip Test: ", signif(dip.test_unbiased_raster$stati
  annotate("text", x = 700, y = 1450, label = paste0("Skew: ", round(skew_dip.test_unbiased_raster,2)))
  annotate("text", x = 700, y = 1350, label = paste0("Mean: ",round(mean_unbiased_raster,2), " m")) +
  annotate("text", x = 700, y = 1250, label = "Classification = Mid-Dominant") +
  geom_vline(xintercept = mean_unbiased_raster, color = "red")
```

```
## Warning: position_stack requires non-overlapping x intervals
```



```r
#projected polygon
polygon_plot +
annotate("text", x = 600, y = 1650, label = "LME 66: Canadian High Arctic: Projected Polygon") +
  annotate("text", x = 700, y = 1550, label = paste0("Dip Test: ", signif(dip.test_polygon$statistic,2)
  annotate("text", x = 700, y = 1450, label = paste0("Skew: ", round(skew_dip.test_polygon,2))) +
  annotate("text", x = 700, y = 1350, label = paste0("Mean: ",round(mean_polygon,2), " m")) +
  annotate("text", x = 700, y = 1250, label = "Classification = Mid-Dominant") +
  geom_vline(xintercept = mean_polygon, color = "red")
```

```
## Warning: position_stack requires non-overlapping x intervals
```



Note that there is no change in classification between the two methods.