Sensitivity Raster versus Projected Shapefile

Code

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

Hide

```
library(raster)
```

Hide

```
library(sf)
```

Hide

```
library(ncdf4)
library(rmapshaper)
```

Hide

```
library(tidyverse)
```

Hide

```
library(diptest)
library(moments)
library(viridis) #colors
```

Hide

```
library(data.table)
```

Hide

```
library(hydroTSM) #hypsometric curves
```

Hide

```
library(gridExtra)
```

Hide

```
library(maptools)
```

Hide

```
library(rgdal)
```

Hide

```
library(rgeos)
```

Hide

```
library(SpaDES)
```

Hide

```
library(rnaturalearth)
library(rnaturalearthdata)
library(equate)
```

Hide

```
etopo_shelf_df <- readRDS("~/Documents/grad school/Rutgers/Repositories/shelf_habitat_di
stribution/etopo_shelf_df.rds")
#bring in bathymetry data frame for shelf regions

#LMEs
LME_spdf <- readOGR("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     accurately represents area in all regions of the sphere, but it does not
accurately represent angles.
equalareaprojection<- crs(" +proj=laea ")
```

Convert data frame to raster for bathymetry layer.

Hide

```


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

#reclassify all values <2000m in depth to 1 instead of actual depth
etopo_shelf_raster_1s<- reclassify(etopo_shelf_raster,cbind(-Inf, Inf, 1))
```

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

###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

Hide

```
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)
```

Hide

```
#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 above 2000 meters

Hide

```
#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 bathymetry
 in this analyses is to lmit to <2000m

#which areas of raster fall within borders?
east_atl_spdf_nobuf_mask_1s <- reclassify(east_atl_spdf_nobuf_agg_mask, c(-Inf, Inf, 1))
```

How to calculate area?

- 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˚

Eastern Atlantic

name of raster = east_atl_spdf_nobuf_mask

Hide

```
#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_atl_spdf_nobuf_mask_1s))
south_extent <- c(xmin(east_atl_spdf_nobuf_mask_1s), xmax(east_atl_spdf_nobuf_mask_1s),
 ymin(east_atl_spdf_nobuf_mask_1s), 0)

#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° 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

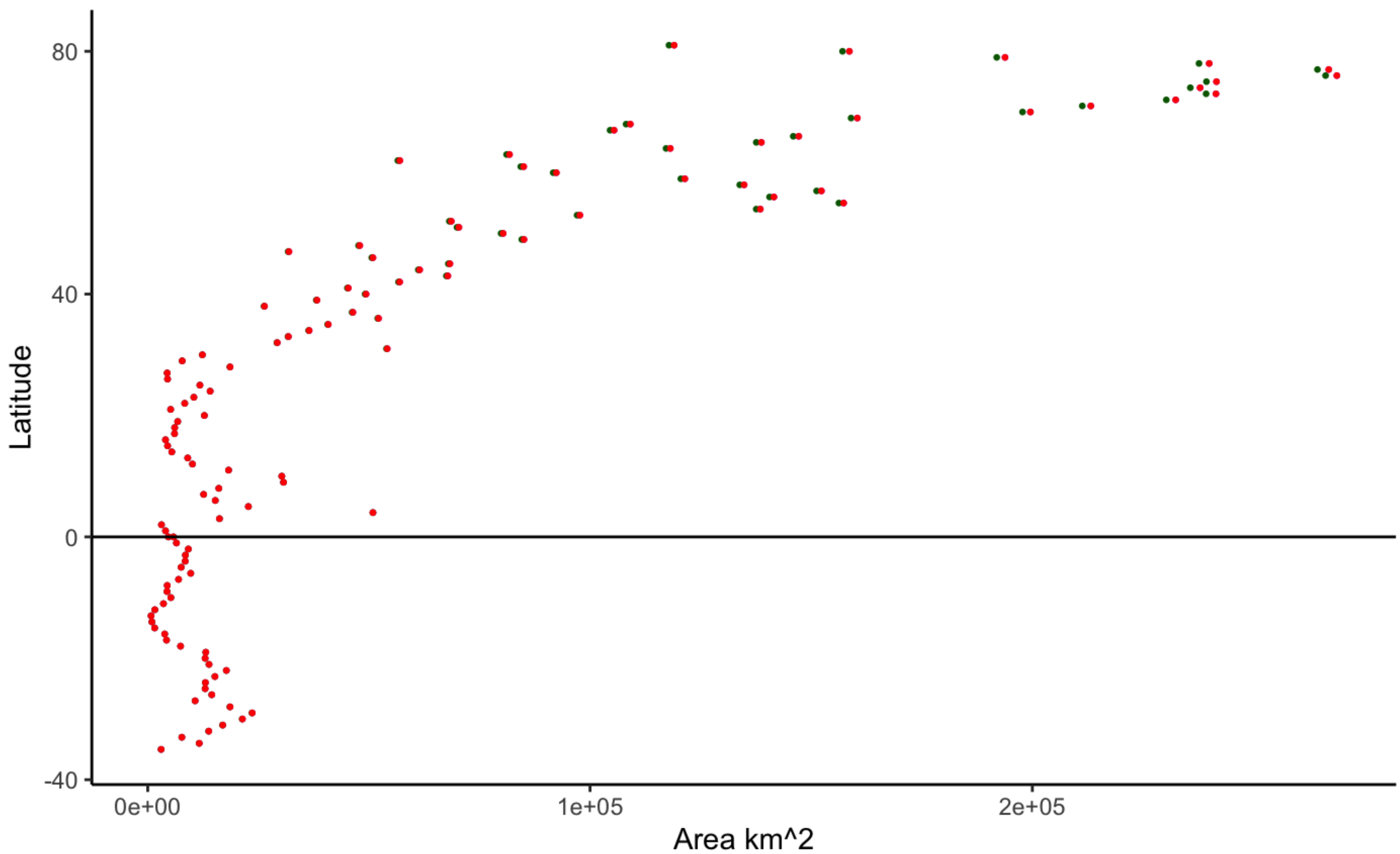
Hide

```
#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_south_latitudes)-1)))

east_atl_shelf_areas[, latitude_start := as.numeric(V1)][, latitude_end := as.numeric(V
1)][, area_rasterarea := as.numeric(V1)][, area_equalareaproj := as.numeric(V1)][, area_
rgeos_gArea := as.numeric(V1)][, V1 := NULL]

#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_1
s), east_atl_north_latitudes[i], east_atl_north_latitudes[i+1])

  #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 ar
ea = 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 p
olygon, (dissolve means all cells with a value of 1 are a single polygon if connected)
```

```
  segment_north.sp <- rasterToPolygons(segment_north, dissolve = T)

  #  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)

    #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_1
s), east_atl_south_latitudes[i+1], east_atl_south_latitudes[i]) #order= xmin, xmax, ymi
n, ymax)

  #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"] <- ea
st_atl_south_latitudes[i]
  east_atl_shelf_areas[i+(length(east_atl_north_latitudes)-1), "latitude_end"] <- east_a
tl_south_latitudes[i+1]


  if(all(is.na(values(segment_south)))) { #if there's no shelf area within a bin, meanin
g there's no shelf area at that latitude

  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"] <- seg
ment_area_raster


  #convert to spatial polygons to check area calculations

  #convert segment from raster to polygon, each cell from the raster is an independent p
olygon, (dissolve means all cells with a value of 1 are a single polygon if connected)
  segment_south.sp <- rasterToPolygons(segment_south, dissolve = T)

  #  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)

    #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_equalarea

  #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 rg
eos::gArea function for polygons
```

```
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()
```



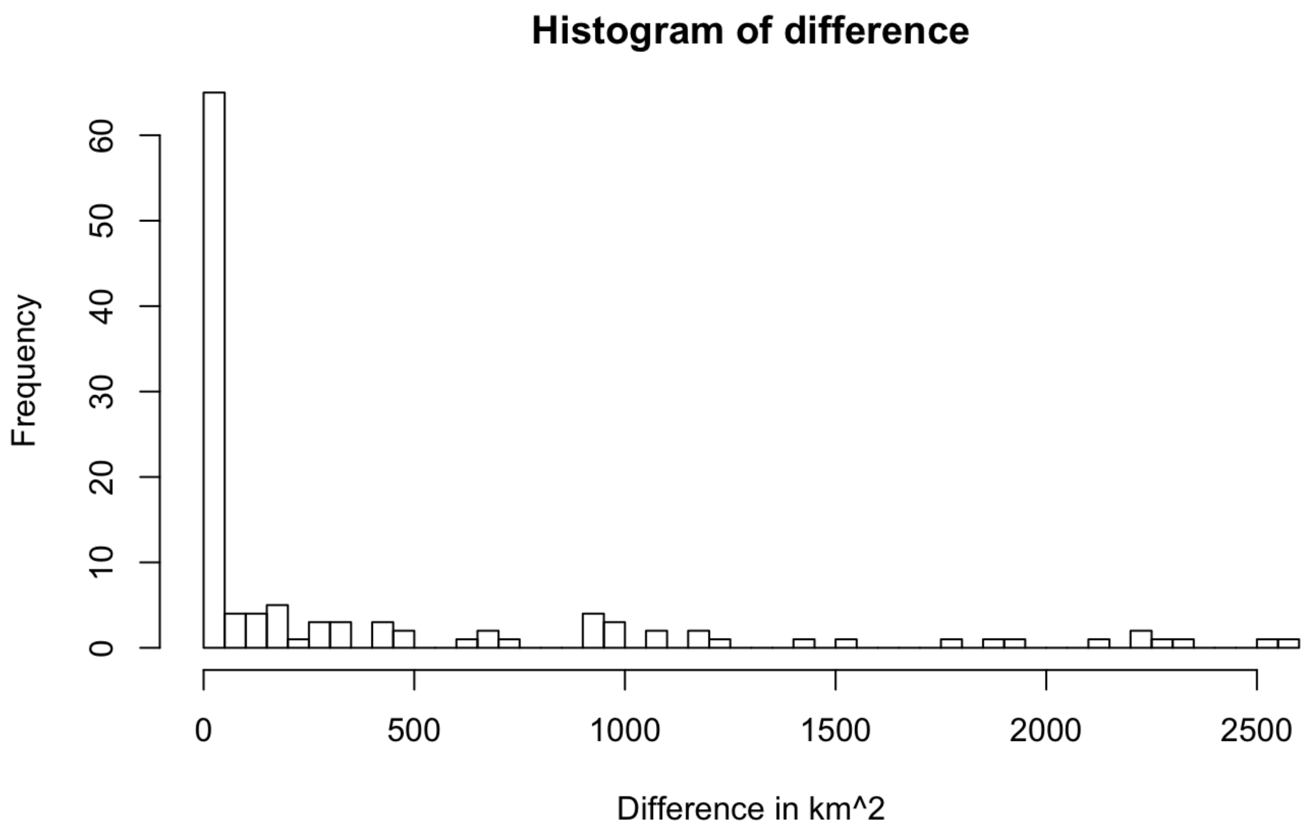
Hide

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

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

Hide

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

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

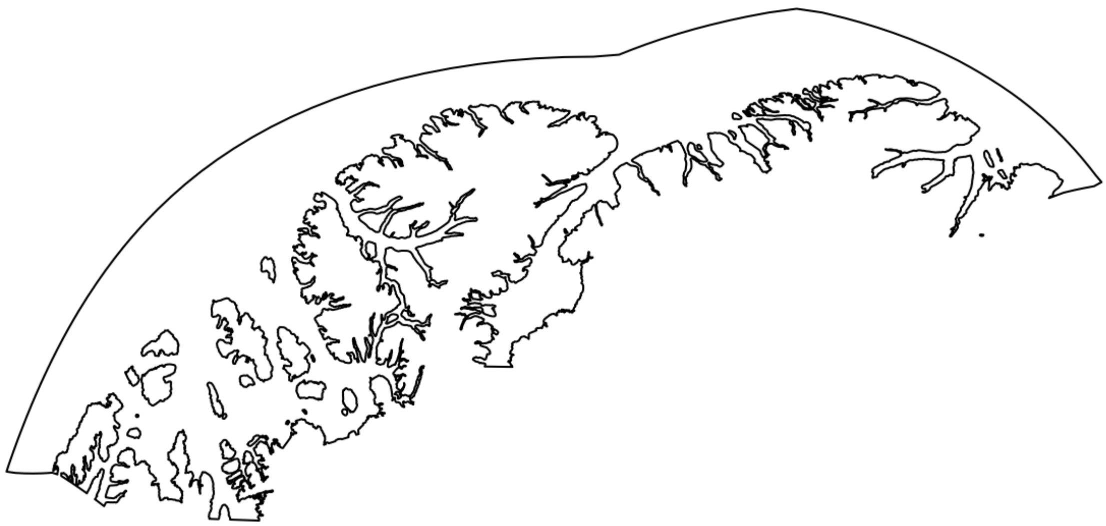


###LME Depth Profiles

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

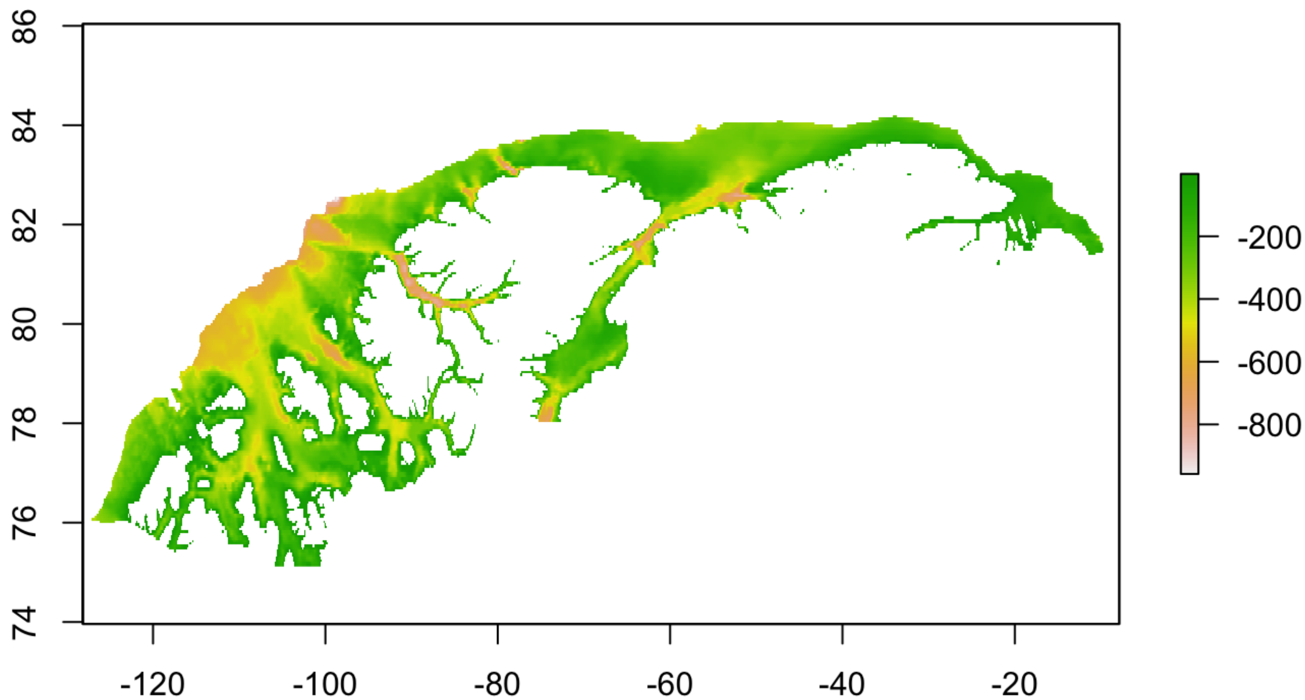
Hide

```
 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 ras
ter is only 1s

plot(LME_high_arctic)
```



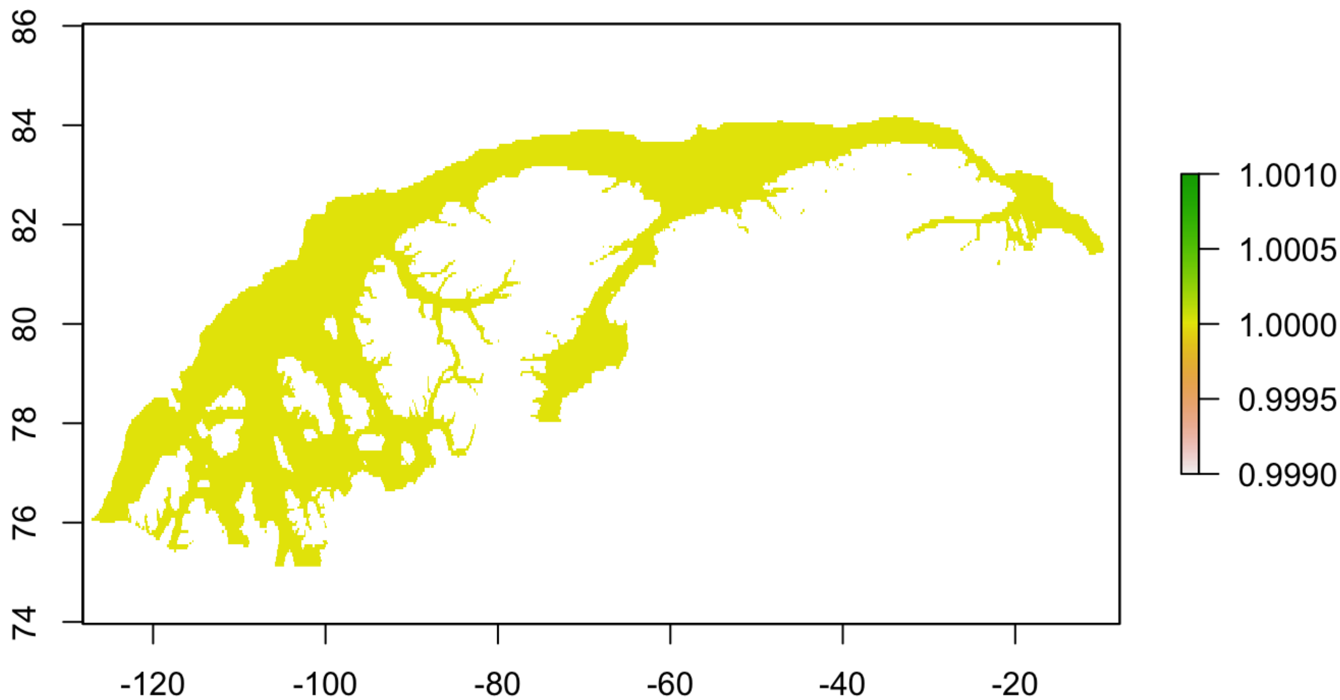
Hide

```
plot(LME_high_arctic_mask)
```



Hide

```
plot(LME_high_arctic_mask_1s)
```

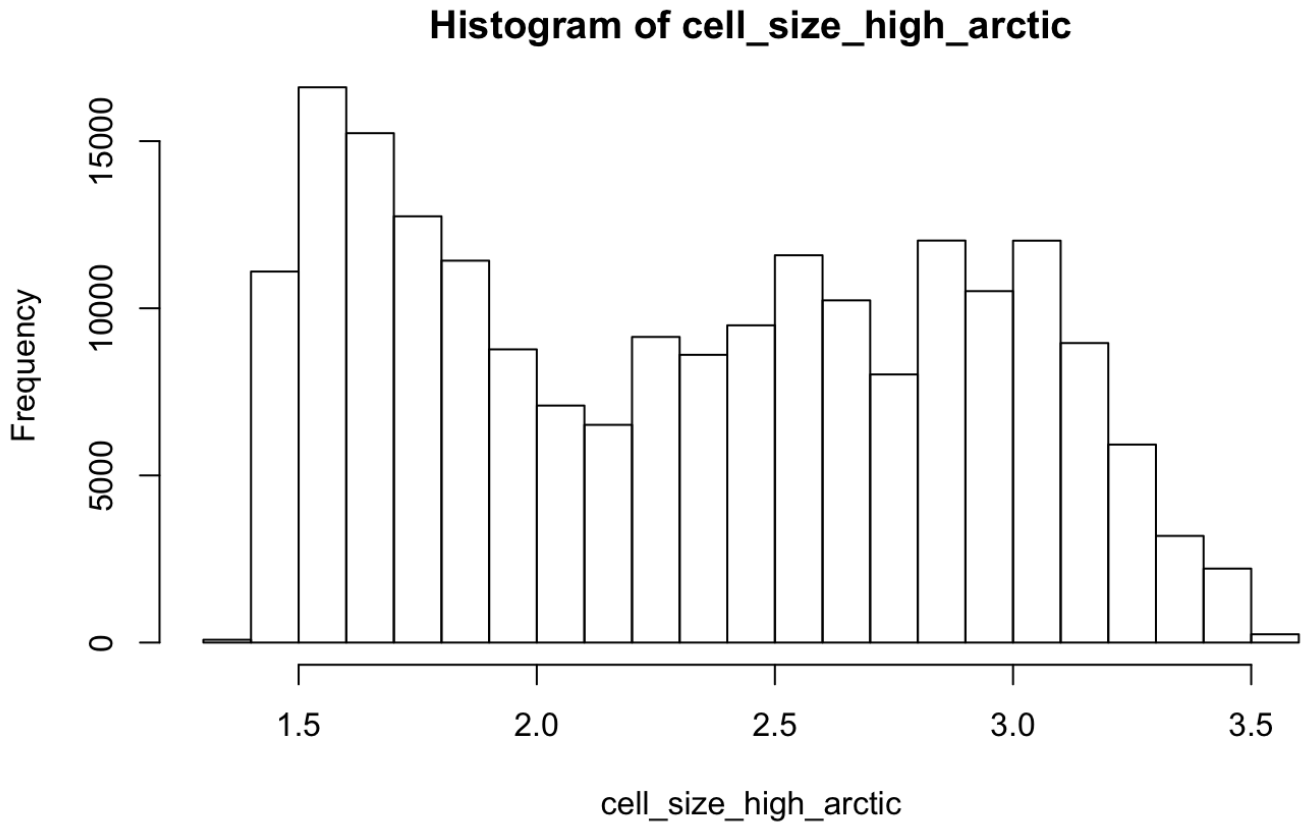


Next, I need to construct hypsometric plots (how does area available vary by depth). One problem is that a value of 30 m in one cell does not necessarily mean the same as a 30 m value in another cell if those cells are different sizes. Therefore, I will compare outputs from different techniques.

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

Hide

```
 #get sizes of all cells in raster [km2]
   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)]

   #cell size varies quite tremendously across high_arctic Sea!! (1 km^2 to 3.5 km^2)
       #histogram to visualize variability in cell size
       hist(cell_size_high_arctic)
```



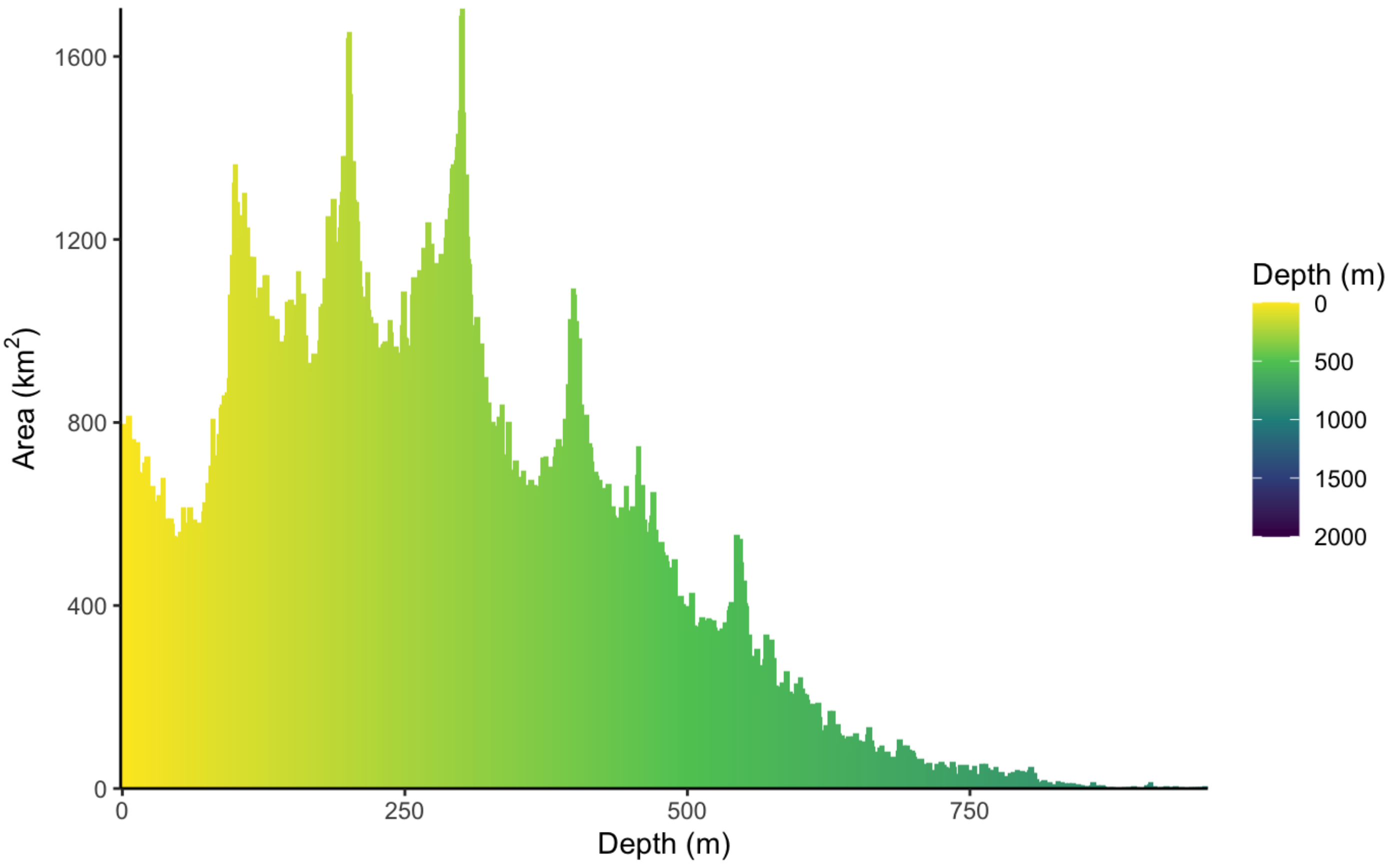
Hide

```
 #compute total area [km2] of all cells in geo_raster
high_arctic_raster_area<-sum(cell_size_high_arctic)
```

I will just pretend all cells are the same size and construct hypsometric curve directly from vector of raster values.

Hide

```
    #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)] #get rid of NA's

  high_arctic_bathy_depth_list_pos <- -1*high_arctic_bathy_depth_list #make all depth va
lues positive


  #cells of each depth value in data table
  high_arctic_bathy_bydepth <- data.table(table(high_arctic_bathy_depth_list_pos)) #make
frequency table of depths

  colnames(high_arctic_bathy_bydepth) <- c("depth", "count")

  #convert columns to numeric
  high_arctic_bathy_bydepth[,depth := as.numeric(depth)][,count := as.numeric(count)]

  #calculate percent of total area in raster
  high_arctic_bathy_bydepth[,percent_area := count/sum(count)] #assuming that all cells
 are the same size


  high_arctic_bathy_bydepth[,area := percent_area*high_arctic_raster_area] #calculate ar
ea by percent area * total area



(basic_raster_plot <- ggplot(data = high_arctic_bathy_bydepth, aes(x = depth, y = 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)))
```



Hide

```
NA
```

This time, to reduce bias by grid cell size, I will also assign specific sizes to each grid cell

Hide

```
cell_size_high_arctic # cell sizes
```

```
   [1] 1.386723 1.386723 1.386723 1.386723 1.386723 1.386723 1.386723 1.386723 1.386723
1.386723
  [11] 1.386723 1.386723 1.394665 1.394665 1.394665 1.394665 1.394665 1.394665 1.394665
1.394665
  [21] 1.394665 1.394665 1.394665 1.394665 1.394665 1.394665 1.394665 1.394665 1.394665
1.394665
  [31] 1.394665 1.394665 1.394665 1.394665 1.394665 1.394665 1.394665 1.394665 1.394665
1.394665
  [41] 1.394665 1.394665 1.394665 1.394665 1.394665 1.394665 1.394665 1.394665 1.394665
1.394665
  [51] 1.394665 1.394665 1.394665 1.394665 1.394665 1.394665 1.394665 1.394665 1.394665
1.394665
  [61] 1.394665 1.394665 1.394665 1.394665 1.394665 1.394665 1.394665 1.394665 1.394665
1.394665
  [71] 1.394665 1.394665 1.394665 1.394665 1.394665 1.394665 1.394665 1.394665 1.394665
1.394665
  [81] 1.394665 1.394665 1.394665 1.394665 1.402606 1.402606 1.402606 1.402606 1.402606
1.402606
  [91] 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606
1.402606
 [101] 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606
1.402606
 [111] 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606
1.402606
 [121] 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606
1.402606
 [131] 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606
1.402606
 [141] 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606
1.402606
 [151] 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606
1.402606
 [161] 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606
1.402606
 [171] 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606
1.402606
 [181] 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606
1.402606
 [191] 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606 1.402606
1.402606
 [201] 1.402606 1.402606 1.402606 1.402606 1.410547 1.410547 1.410547 1.410547 1.410547
1.410547
 [211] 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547
1.410547
 [221] 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547
1.410547
 [231] 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547
1.410547
 [241] 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547
1.410547
 [251] 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547
1.410547
 [261] 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547
```

```
1.410547
 [271] 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547
1.410547
 [281] 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547
1.410547
 [291] 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547
1.410547
 [301] 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547
1.410547
 [311] 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547
1.410547
 [321] 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547
1.410547
 [331] 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547
1.410547
 [341] 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547
1.410547
 [351] 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547
1.410547
 [361] 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547
1.410547
 [371] 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547
1.410547
 [381] 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547
1.410547
 [391] 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.410547 1.418487
1.418487
 [401] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [411] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [421] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [431] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [441] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [451] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [461] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [471] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [481] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [491] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [501] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [511] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [521] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [531] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
```

```
1.418487
 [541] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [551] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [561] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [571] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [581] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [591] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [601] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [611] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [621] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [631] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [641] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [651] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [661] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [671] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [681] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [691] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [701] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [711] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [721] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [731] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [741] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [751] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [761] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [771] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [781] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [791] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [801] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
```

```
1.418487
 [811] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [821] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [831] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.418487
 [841] 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487 1.418487
1.426427
 [851] 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427
1.426427
 [861] 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427
1.426427
 [871] 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427
1.426427
 [881] 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427
1.426427
 [891] 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427
1.426427
 [901] 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427
1.426427
 [911] 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427
1.426427
 [921] 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427
1.426427
 [931] 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427
1.426427
 [941] 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427
1.426427
 [951] 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427
1.426427
 [961] 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427
1.426427
 [971] 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427
1.426427
 [981] 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427
1.426427
 [991] 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427 1.426427
1.426427
 [ reached getOption("max.print") -- omitted 200762 entries ]
```

Hide

```
high_arctic_bathy_depth_list # cell values
```

```
  [1] -262 -262 -262 -263 -263 -264 -265 -266 -265 -264 -265 -265 -259 -256 -252 -249 -
247 -243
 [19] -241 -239 -236 -234 -232 -229 -228 -227 -225 -226 -225 -225 -225 -224 -224 -223 -
222 -222
 [37] -222 -222 -221 -221 -221 -221 -221 -222 -222 -223 -223 -223 -224 -225 -226 -226 -
227 -227
 [55] -228 -230 -231 -233 -234 -235 -236 -238 -239 -241 -242 -243 -246 -248 -250 -252 -
255 -257
 [73] -259 -261 -263 -265 -267 -269 -271 -273 -275 -277 -279 -281 -291 -290 -289 -287 -
285 -283
 [91] -281 -278 -275 -272 -268 -266 -263 -260 -257 -254 -251 -248 -245 -242 -239 -237 -
234 -231
[109] -228 -226 -222 -219 -217 -215 -213 -211 -209 -208 -206 -205 -202 -201 -200 -199 -
198 -198
[127] -198 -198 -199 -200 -200 -199 -199 -200 -199 -199 -198 -199 -199 -199 -200 -200 -
200 -200
[145] -201 -201 -202 -202 -203 -203 -204 -205 -206 -206 -207 -208 -210 -211 -211 -212 -
214 -215
[163] -217 -218 -220 -222 -224 -227 -229 -231 -233 -235 -237 -240 -243 -245 -248 -250 -
252 -255
[181] -257 -260 -263 -264 -267 -270 -273 -275 -277 -279 -281 -284 -286 -287 -288 -290 -
291 -292
[199] -293 -294 -295 -296 -297 -298 -353 -352 -353 -353 -353 -352 -351 -351 -351 -352 -
352 -353
[217] -355 -357 -359 -360 -361 -364 -367 -369 -371 -373 -375 -377 -289 -289 -288 -287 -
286 -286
[235] -286 -286 -284 -282 -280 -280 -278 -276 -275 -273 -271 -269 -268 -266 -264 -262 -
260 -258
[253] -255 -254 -252 -251 -248 -246 -244 -241 -239 -236 -232 -230 -228 -225 -222 -220 -
217 -215
[271] -213 -211 -209 -206 -205 -204 -202 -199 -198 -196 -194 -193 -192 -191 -190 -190 -
190 -189
[289] -188 -187 -187 -187 -187 -186 -186 -186 -186 -186 -187 -189 -188 -188 -189 -190 -
191 -191
[307] -191 -192 -193 -193 -193 -193 -194 -194 -194 -195 -195 -196 -196 -196 -196 -197 -
198 -198
[325] -198 -198 -198 -199 -199 -199 -199 -201 -203 -203 -204 -206 -206 -208 -210 -212 -
213 -215
[343] -217 -219 -221 -222 -224 -226 -228 -231 -233 -235 -237 -239 -241 -244 -246 -248 -
250 -251
[361] -253 -256 -258 -259 -262 -263 -265 -267 -269 -271 -272 -273 -275 -277 -278 -279 -
282 -283
[379] -284 -285 -286 -287 -288 -290 -291 -292 -294 -297 -300 -301 -302 -302 -304 -305 -
306 -308
[397] -310 -312 -334 -332 -331 -330 -328 -327 -327 -328 -328 -328 -329 -329 -328 -329 -
330 -330
[415] -332 -333 -333 -334 -336 -336 -337 -337 -338 -339 -340 -340 -341 -342 -343 -345 -
344 -345
[433] -346 -347 -348 -348 -349 -351 -350 -351 -349 -348 -348 -347 -345 -345 -344 -343 -
342 -341
[451] -340 -339 -337 -336 -334 -333 -333 -333 -332 -332 -331 -331 -330 -328 -326 -324 -
322 -320
[469] -318 -315 -312 -310 -310 -310 -309 -310 -312 -312 -313 -314 -316 -317 -319 -320 -
```

```
321 -323
 [487] -324 -326 -328 -330 -333 -336 -338 -341 -344 -347 -350 -353 -356 -358 -361 -364 -
366 -368
 [505] -370 -371 -372 -372 -373 -374 -373 -374 -375 -376 -377 -376 -375 -375 -374 -373 -
373 -372
 [523] -372 -371 -370 -369 -367 -364 -362 -360 -358 -354 -352 -350 -347 -345 -343 -340 -
337 -335
 [541] -332 -330 -329 -328 -327 -325 -325 -325 -326 -325 -324 -325 -326 -327 -327 -328 -
331 -333
 [559] -335 -336 -337 -339 -340 -340 -341 -342 -344 -345 -346 -347 -347 -347 -346 -345 -
345 -344
 [577] -343 -342 -341 -342 -342 -342 -341 -342 -342 -343 -343 -344 -346 -348 -349 -352 -
355 -357
 [595] -360 -363 -364 -367 -370 -371 -373 -375 -378 -379 -380 -380 -380 -381 -383 -383 -
384 -384
 [613] -385 -385 -385 -385 -386 -385 -385 -386 -387 -388 -388 -388 -388 -389 -288 -287 -
285 -284
 [631] -284 -283 -283 -283 -283 -282 -282 -280 -280 -280 -279 -278 -277 -276 -275 -274 -
273 -272
 [649] -272 -270 -269 -268 -266 -265 -264 -263 -261 -260 -259 -258 -256 -253 -253 -252 -
250 -247
 [667] -244 -242 -240 -237 -235 -233 -231 -229 -227 -224 -220 -218 -216 -214 -213 -211 -
210 -209
 [685] -208 -205 -203 -202 -200 -199 -197 -196 -194 -192 -191 -191 -189 -189 -188 -187 -
186 -184
 [703] -184 -183 -182 -183 -182 -181 -181 -180 -181 -180 -180 -180 -180 -180 -180 -180 -
180 -180
 [721] -180 -180 -180 -181 -181 -182 -182 -182 -183 -183 -183 -185 -185 -185 -185 -186 -
187 -187
 [739] -187 -187 -188 -189 -189 -189 -189 -190 -191 -191 -191 -192 -192 -192 -194 -194 -
194 -194
 [757] -196 -196 -196 -197 -198 -199 -200 -200 -201 -203 -203 -204 -205 -206 -207 -208 -
209 -210
 [775] -210 -211 -213 -215 -217 -218 -220 -221 -222 -223 -225 -226 -228 -229 -230 -231 -
232 -233
 [793] -235 -235 -236 -237 -238 -239 -240 -241 -242 -244 -245 -246 -247 -249 -251 -252 -
254 -257
 [811] -258 -259 -262 -263 -264 -265 -265 -266 -267 -267 -267 -268 -269 -271 -271 -271 -
271 -272
 [829] -273 -274 -275 -277 -279 -281 -283 -284 -285 -285 -287 -289 -292 -295 -297 -300 -
301 -301
 [847] -302 -304 -306 -342 -340 -338 -335 -333 -331 -328 -325 -323 -322 -321 -320 -319 -
318 -316
 [865] -315 -315 -316 -317 -317 -318 -319 -320 -320 -320 -321 -322 -321 -321 -322 -323 -
323 -323
 [883] -323 -324 -324 -324 -324 -325 -325 -326 -326 -327 -326 -325 -326 -326 -326 -326 -
326 -327
 [901] -328 -327 -327 -327 -328 -328 -328 -328 -328 -328 -328 -328 -328 -328 -328 -328 -
328 -329
 [919] -329 -329 -328 -328 -328 -328 -328 -328 -327 -327 -325 -325 -325 -323 -322 -322 -
321 -320
 [937] -319 -318 -317 -315 -314 -314 -314 -314 -313 -313 -312 -311 -310 -309 -308 -307 -
306 -304
 [955] -304 -303 -301 -301 -303 -303 -304 -305 -304 -304 -305 -305 -304 -305 -306 -305 -
```

```
305 -305
 [973] -305 -304 -305 -306 -307 -308 -309 -310 -311 -313 -315 -317 -319 -320 -322 -324 -
325 -327
 [991] -330 -332 -333 -335 -336 -338 -340 -342 -342 -343
 [ reached getOption("max.print") -- omitted 200762 entries ]
```

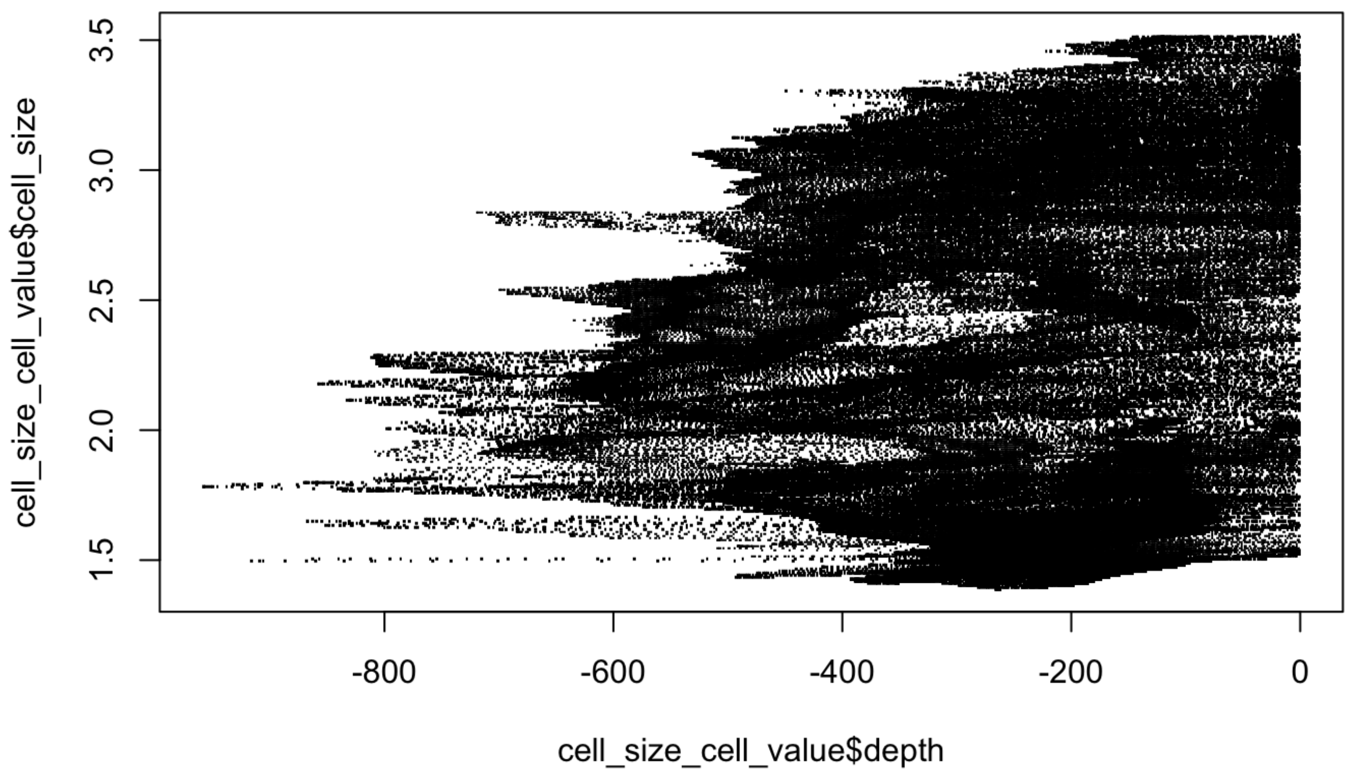
Hide

```
cell_size_cell_value <- data.table(cell_size = cell_size_high_arctic, depth = high_arcti
c_bathy_depth_list )

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

```
[1] 0.09480613
```

Hide

```
plot(cell_size_cell_value$depth, cell_size_cell_value$cell_size, pch = ".") #more likely
to have smaller cells deeper which could lead us to conclude that we have less habitat a
t depth than we thought, let's check
```



Sum area across cells with same depth

Hide

```
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))
```

And lastly, the 3rd method to convert to polygon for equal area projection and then produce frequency table

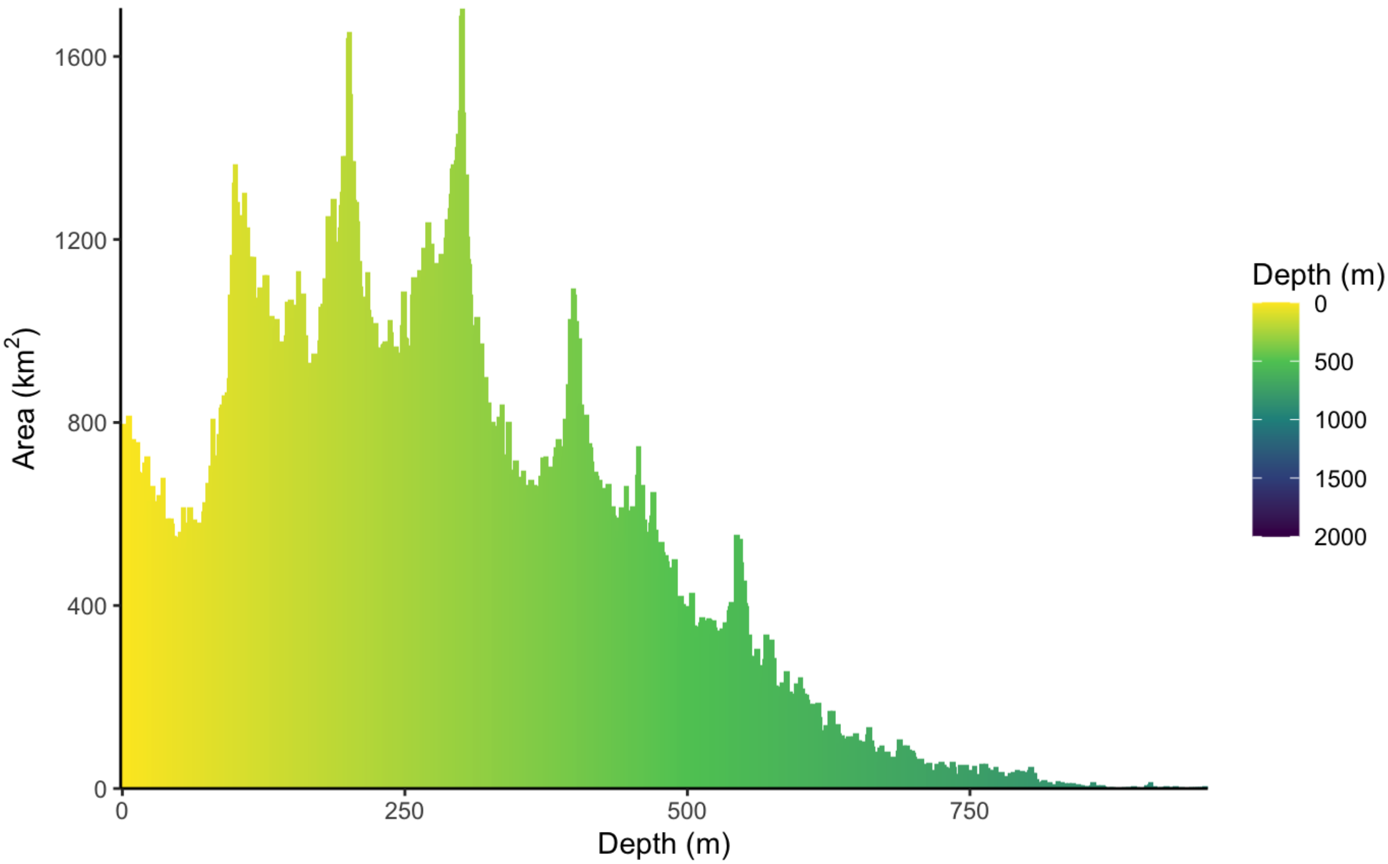
Hide

```
#POLYGON MAY BE THE WAY TO GO BECAUSE CELL SIZE VARIES SO MUCH

  #raster back to polygon
  LME_high_arctic.sp <- rasterToPolygons(LME_high_arctic_mask, dissolve = T) #dissolve m
eans all same values are one polygon feature

  #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 ar
ea in m^2 of each depth within polygon

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

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 = dep
th)) +
    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 three graphs visually

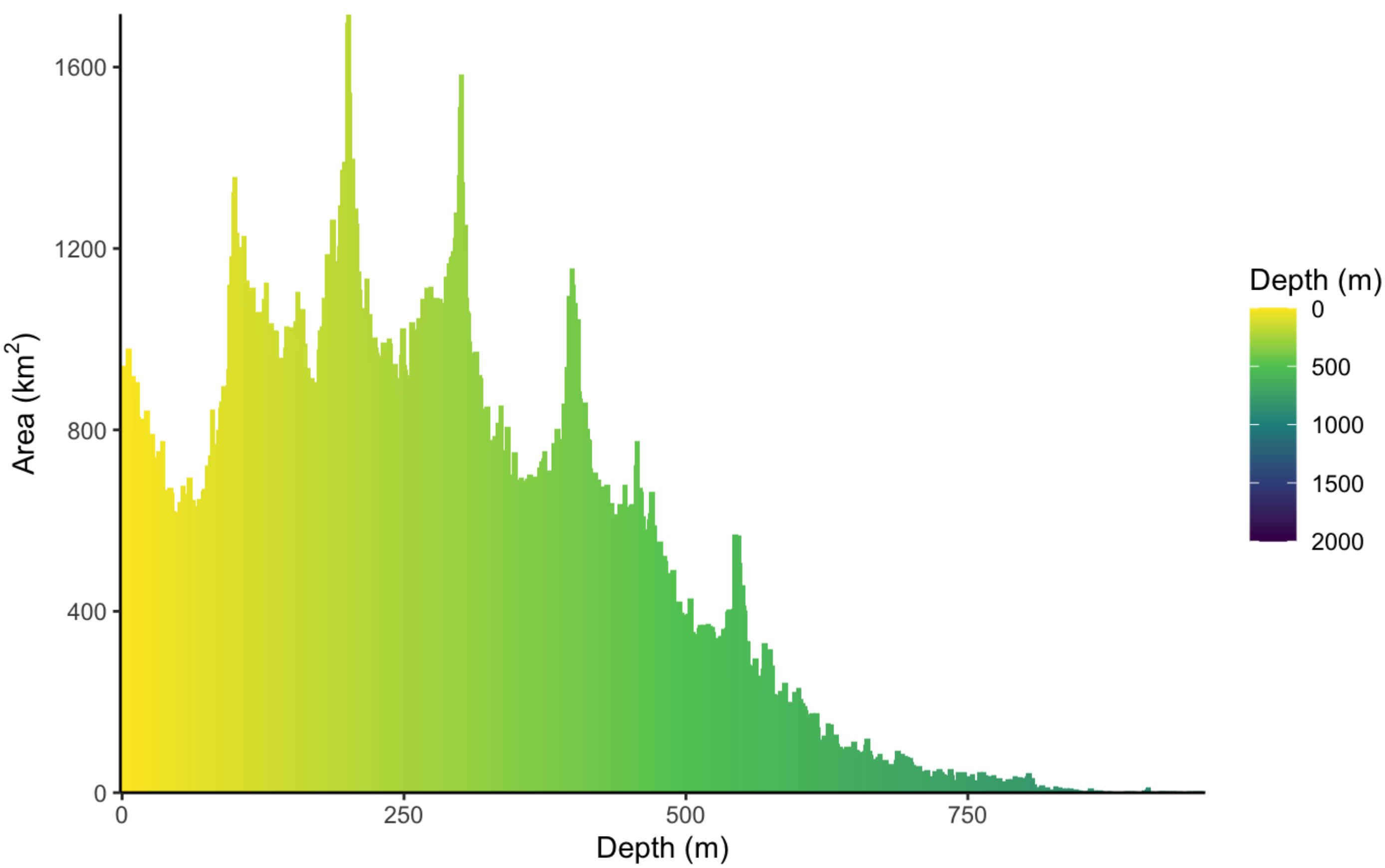
Hide

```
basic_raster_plot
```



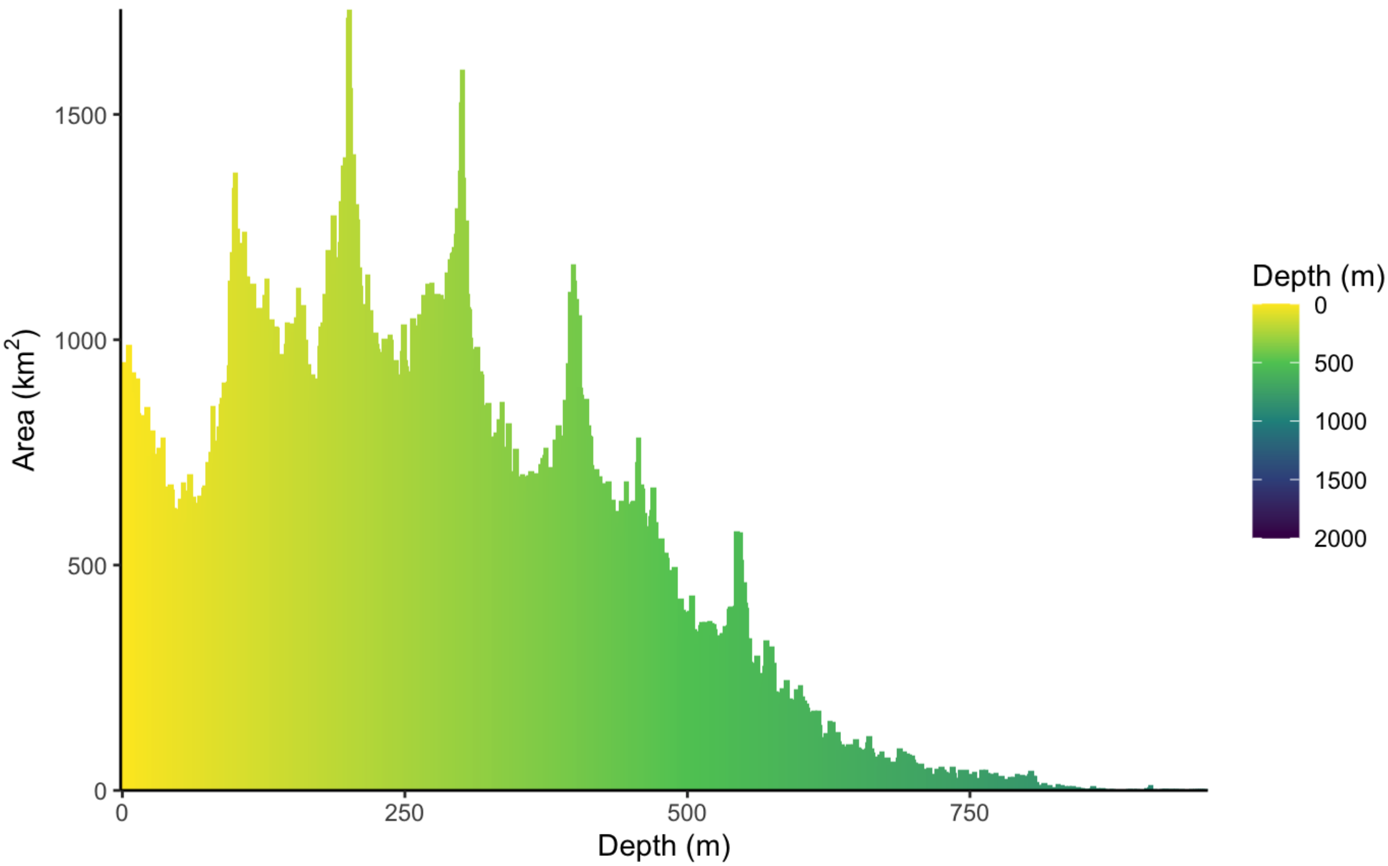
Hide

```
raster_areabycell_plot
```



Hide

```
polygon_plot
```



Visually, the shape does not change very much, even when we allow for the bias to persist across because of variable cell sizes (basic_raster_plot)

But, what about the statistics, and final classification?

First, for raster without a bias correction

Hide

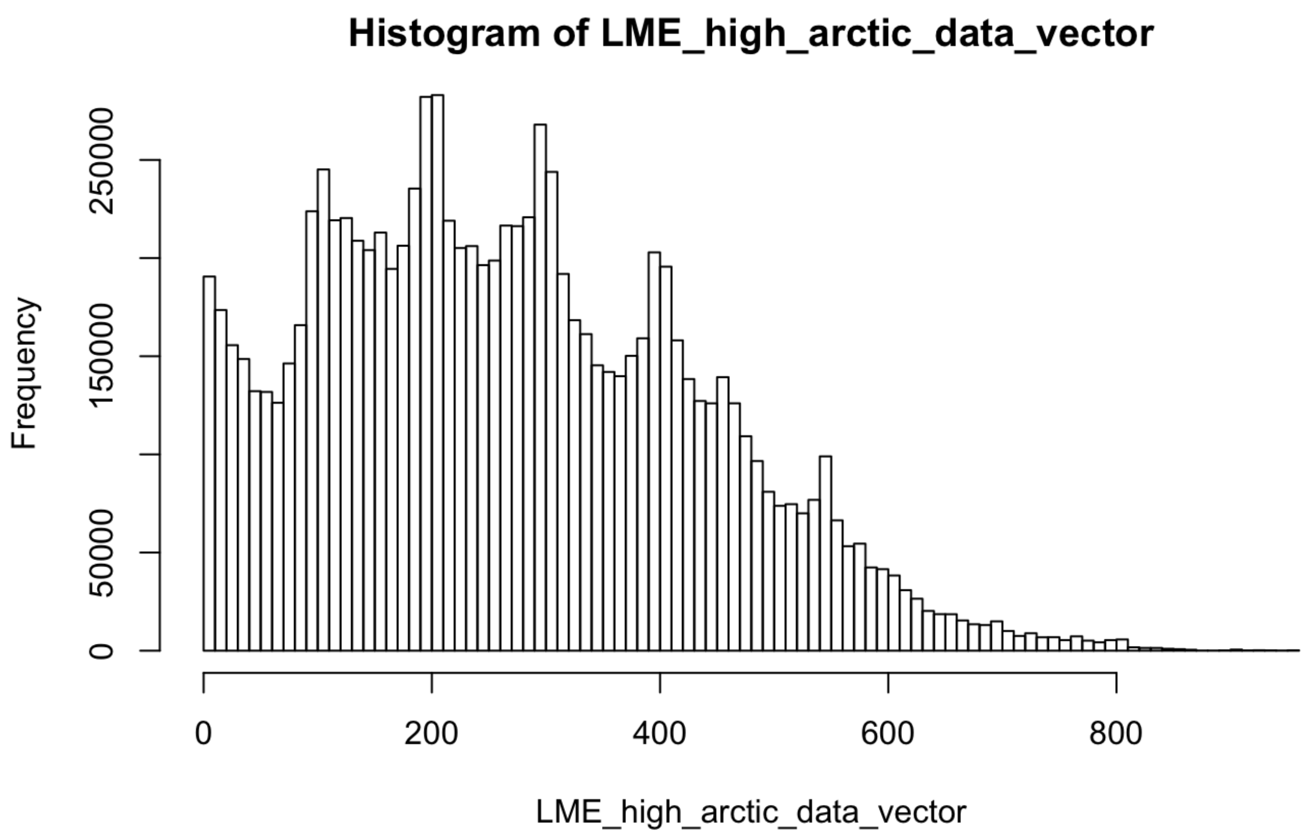
```
diptest_high_arctic_raster <- dip.test(high_arctic_bathy_depth_list_pos, simulate.p.valu
e = TRUE, B = 2000)
save(diptest_high_arctic_raster, file = "diptest_high_arctic_raster.RData")
#if already created
#load("diptest_high_arctic_raster.RData")

p.value_high_arctic_raster <- diptest_high_arctic_raster$p.value
skew_high_arctic_raster <- skewness(high_arctic_bathy_depth_list_pos, na.rm = T)
kurtosis_high_arctic_raster <- kurtosis(high_arctic_bathy_depth_list_pos, na.rm = T)
mean_high_arctic_raster <- mean(high_arctic_bathy_depth_list_pos)
max_depth_high_arctic_raster <- max(high_arctic_bathy_depth_list_pos)
median_depth_high_arctic_raster <- median(high_arctic_bathy_depth_list_pos)
```

Next, 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.

Hide

```
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(L
ME_high_arctic_data_vector))

}

#plotting
hist(LME_high_arctic_data_vector, 100)
```



Perform calculations on unbiased raster.

Hide

```
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 = TRU
E, 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)
  kurtosis_unbiased_raster <- kurt.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)
```

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

Hide

```
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(LME_high_arctic_data_vector_polygon))

}
```

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

Hide

```
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 = TRU
E, B = 2000)
  p.value_dip.test_polygon <- dip.test_polygon$p.value
  skew_dip.test_polygon <- skew.freqtab(freq_table_polygon.ft)
  kurtosis_polygon <- kurt.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)
```

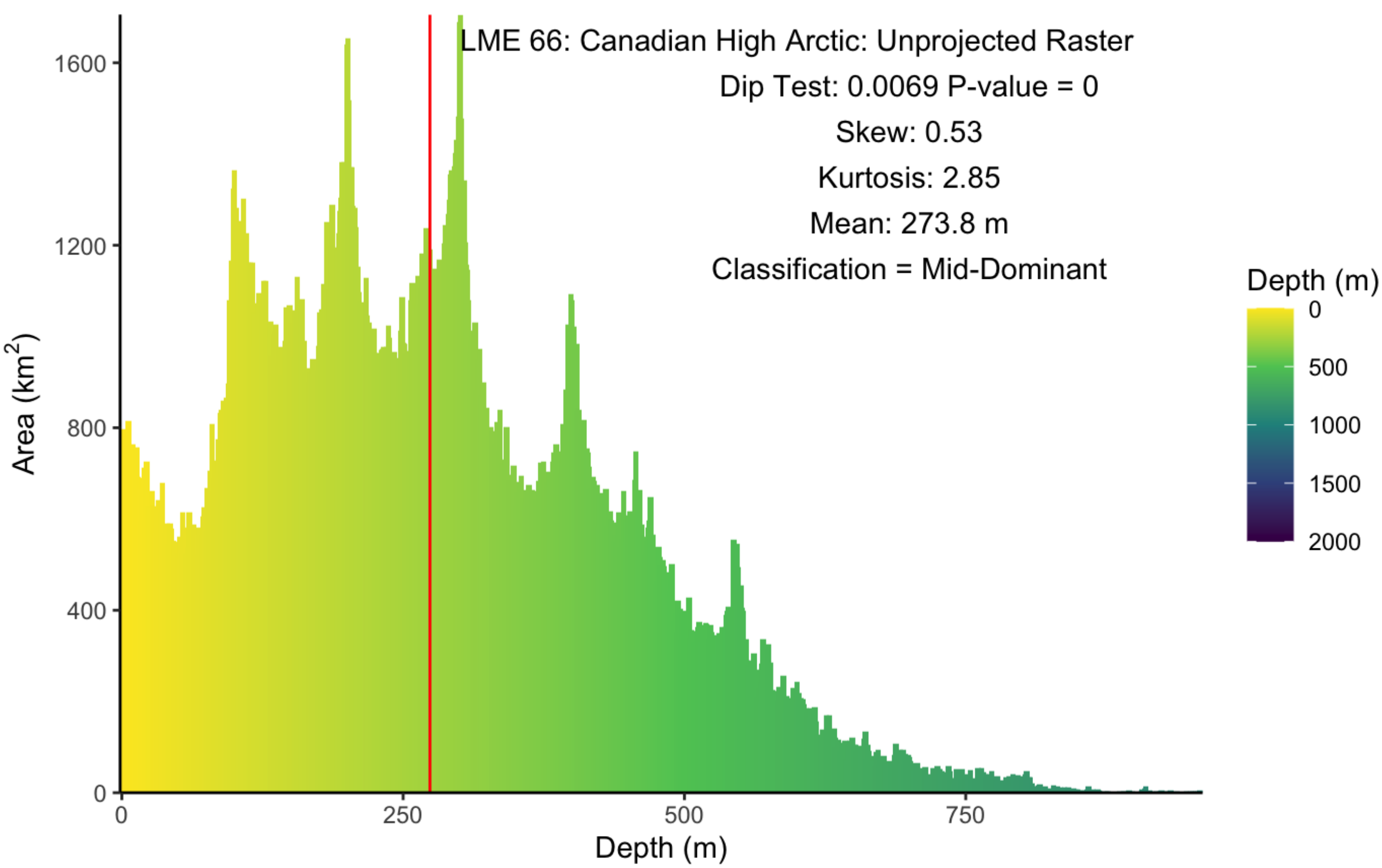
Re-plot three methods with all the statistics, and classification based on:

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

We are not currently using Kurtosis, but I'm still including to show the value doesn't change much as you change how we calculate area.

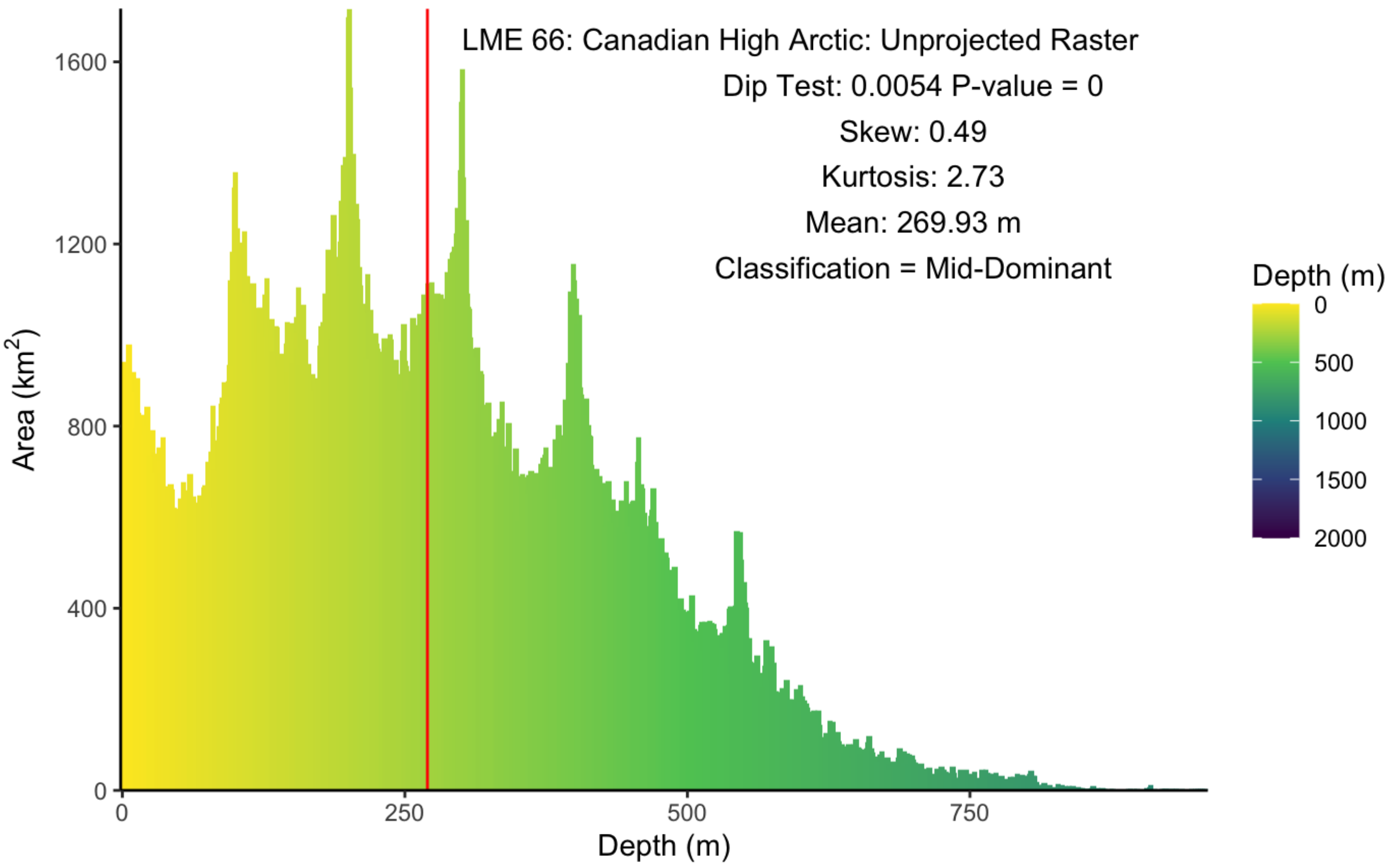
Hide

```
#first, plot for basic raster
basic_raster_plot +
annotate("text", x = 600, y = 1650, label = "LME 66: Canadian High Arctic: Unprojected R
aster") +
  annotate("text", x = 700, y = 1550, label = paste0("Dip Test: ", signif(diptest_high_a
rctic_raster$statistic,2), " P-value = ", signif(p.value_high_arctic_raster, 2))) +
  annotate("text", x = 700, y = 1450, label = paste0("Skew: ", round(skew_high_arctic_ra
ster,2))) +
  annotate("text", x = 700, y = 1350, label = paste0("Kurtosis: ",round(kurtosis_high_ar
ctic_raster,2))) +
  annotate("text", x = 700, y = 1250, label = paste0("Mean: ",round(mean_high_arctic_ras
ter,2), " m")) +
  annotate("text", x = 700, y = 1150, label = "Classification = Mid-Dominant") +
  geom_vline(xintercept = mean_high_arctic_raster, color = "red")
```



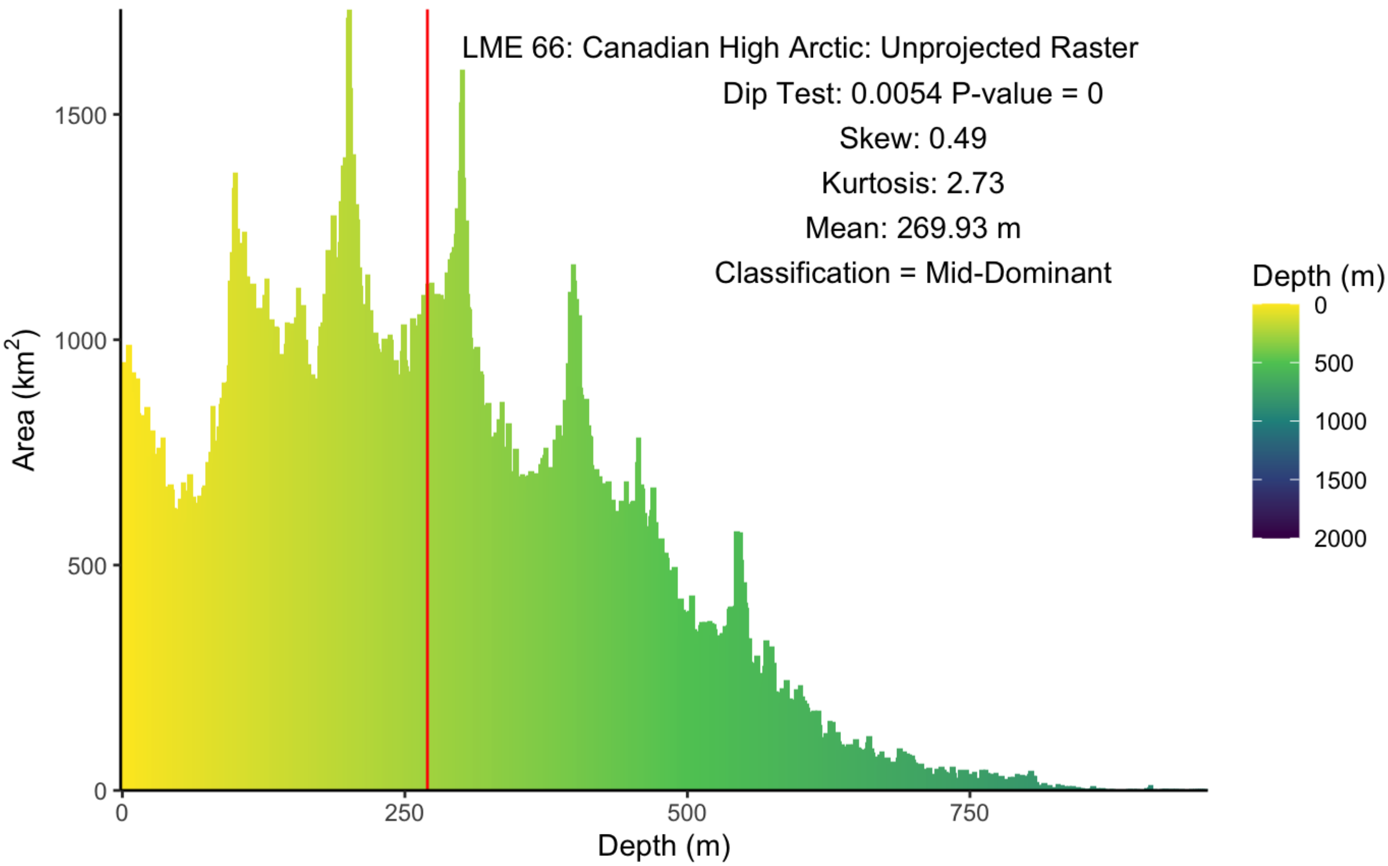
Hide

```
#second, plot for unbiased raster
raster_areabycell_plot +
annotate("text", x = 600, y = 1650, label = "LME 66: Canadian High Arctic: Unprojected R
aster") +
  annotate("text", x = 700, y = 1550, label = paste0("Dip Test: ", signif(dip.test_unbia
sed_raster$statistic,2), " P-value = ", signif(p.value_dip.test_unbiased_raster, 2))) +
  annotate("text", x = 700, y = 1450, label = paste0("Skew: ", round(skew_dip.test_unbia
sed_raster,2))) +
  annotate("text", x = 700, y = 1350, label = paste0("Kurtosis: ",round(kurtosis_unbiase
d_raster,2))) +
  annotate("text", x = 700, y = 1250, label = paste0("Mean: ",round(mean_unbiased_raste
r,2), " m")) +
  annotate("text", x = 700, y = 1150, label = "Classification = Mid-Dominant") +
  geom_vline(xintercept = mean_unbiased_raster, color = "red")
```



Hide

```
#third, for projected polygon
polygon_plot +
annotate("text", x = 600, y = 1650, label = "LME 66: Canadian High Arctic: Unprojected R
aster") +
  annotate("text", x = 700, y = 1550, label = paste0("Dip Test: ", signif(dip.test_polyg
on$statistic,2), " P-value = ", signif(p.value_dip.test_polygon, 2))) +
  annotate("text", x = 700, y = 1450, label = paste0("Skew: ", round(skew_dip.test_polyg
on,2))) +
  annotate("text", x = 700, y = 1350, label = paste0("Kurtosis: ",round(kurtosis_polygo
n,2))) +
  annotate("text", x = 700, y = 1250, label = paste0("Mean: ",round(mean_polygon,2), "
 m")) +
  annotate("text", x = 700, y = 1150, label = "Classification = Mid-Dominant") +
  geom_vline(xintercept = mean_polygon, color = "red")
```



Hide

```
NA
NA
```

Note that there is no change in classification across the three methods.