# Vignette: Creating a GridEZ Sampling Frame by Breaking Runs Into Pieces

*Caitlin B. Clary, Ph.D.*

*Biostat Global Consulting*

*7 June 2024*

This vignette illustrates the process of generating a gridded population sampling frame for a country by stitching together *multiple* sets of GridEZ output. This process of generating a sampling frame in pieces is sometimes required when memory limitations prevent users from creating the entire sampling frame at once with a single call to GridEZ.

The process outlined in this vignette has steps in R and in [QGIS](https://qgis.org/en/site/).

## Input Files

The GridEZ algorithm requires three data inputs:

* **Boundaries file**: shapefile that indicates boundaries within which PSUs should be created
* **Population file**: raster that provides population counts
* **Settlement file**: raster that provides settlement type information

For this example we will create a sampling frame for Eswatini (formerly Swaziland) using:

* Eswatini level2 boundaries geoJSON from [GADM](https://gadm.org/data.html)
* Gridded population from [GHS-POP R2023A](https://ghsl.jrc.ec.europa.eu/ghs_pop2023.php) E2030
* Gridded settlements from [GHS-SMOD R2023A](https://ghsl.jrc.ec.europa.eu/ghs_smod2023.php) E2030

When creating a national sampling frame on personal computers it is often necessary to run the GridEZ algorithm multiple times, creating sampling frames for different sections of the country—unless the country is small, personal computers usually don’t have enough memory to be able to generate a sampling frame for the entire country at once.

Eswatini is a very small country, so on most computers it would **not** be necessary to run GridEZ multiple times to generate a national sampling frame. Eswatini is used in this vignette as a simple illustration with output that is generated relatively quickly.

## Planning GridEZ Runs

A map of different colored states

Description automatically generatedThe Eswatini shapefile has four admin1 units (regions) and 55 admin2 units (tinkhundla). For this example, we will split Eswatini by region (GID\_1) and run GridEZ once for each region.

## Buffering Boundary Inputs

In early testing an issue emerged when combining output from multiple gridEZ runs, where the PSUs from different runs would be misaligned (see the white gaps in the screenshot below):

A colorful map of the united states

Description automatically generated

To avoid this alignment issue when combining files from multiple GridEZ runs, we first add a *buffer* to the outer boundary for each component GridEZ run, and later *crop* the GridEZ output using the original boundaries.

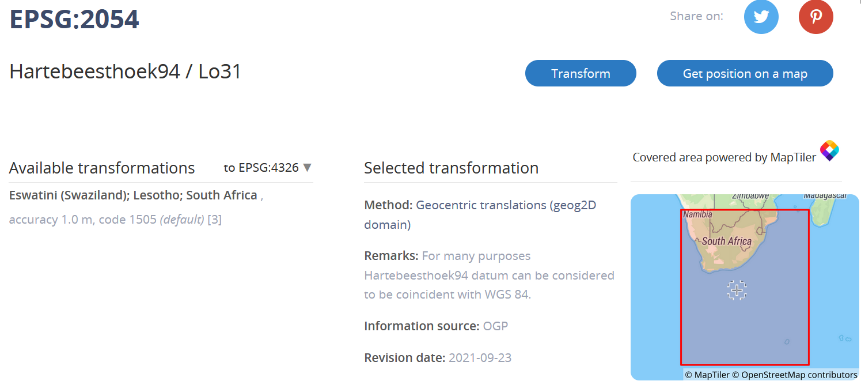
### QGIS: reproject country shapefile and divide into chunks

We will do a separate GridEZ run for each of the chunks that we create and buffer.

When creating a buffer around each chunk, we will want to do operations in meters and kilometers, so the first step is to **reproject** this shapefile using a projection with units of meters, rather than units of degrees. To do this, find an appropriate *projected* coordinate reference system (CRS) for this part of the world.

1. Go to epsg.io
2. Search for the country name
3. A screenshot of a computer

   Description automatically generatedIn the sidebar, select ‘Projected’ to narrow down to projected coordinate reference systems
4. You’ll get a variety of results, and often there will be multiple projections that would be appropriate to use. Browse the results and look for a CRS with a covered area that covers all or most of your country of interest.

For this Eswatini example we will use EPSG:2054. Keep note of the CRS you select.

Next we will **reproject** the shapefile layer in QGIS.

1. Open your shapefile, either by dragging and dropping the file into the QGIS interface, or by adding a vector layer (click Layer -> Add Layer -> Add Vector Layer, then click the … icon by the ‘Vector Dataset(s)’ input to select your shapefile).
2. Open the processing toolbox

A screenshot of a computer

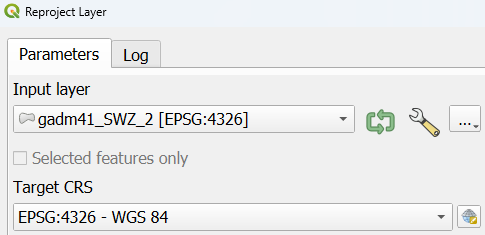
Description automatically generated

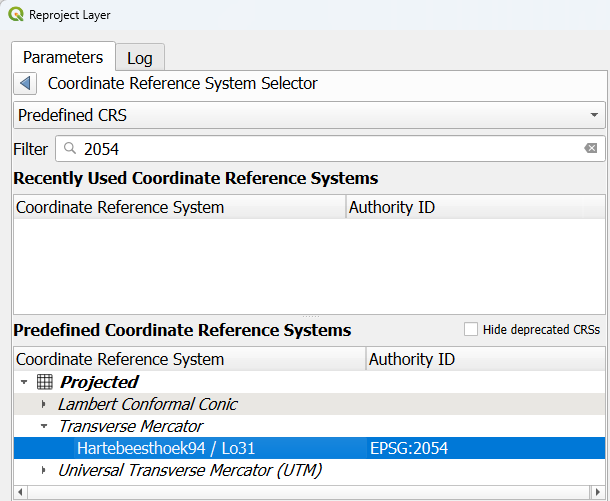
1. Search for ‘reproject’ in the processing toolbox and double click ‘Reproject layer’

A screenshot of a toolbox

Description automatically generated

1. Make sure your input layer is the shapefile you just added. Click the globe icon in the Target CRS input to browse for the CRS you want to reproject to.



Search for the CRS you identified using epsg.io, and select that CRS.

A screenshot of a computer

Description automatically generated

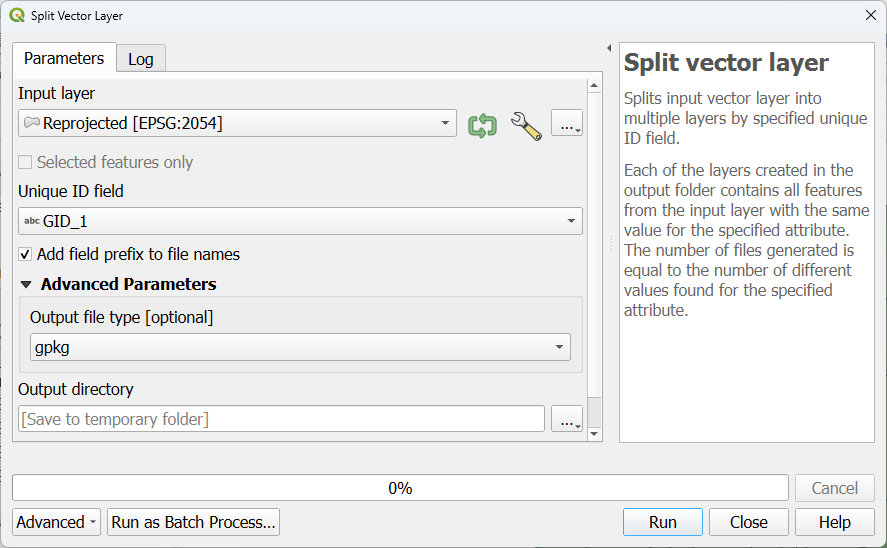
Then click ‘Run’. A layer called ‘Reprojected’ should now be in your ‘Layers’ sidebar in QGIS.

1. Next, we will **split** the reprojected shapefile into chunks for each gridEZ run. From the menu at the top of the QGIS window, select Vector -> Data Management Tools -> Split Vector Layer

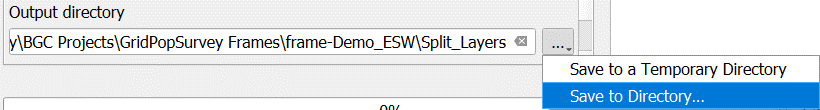
A screenshot of a computer

Description automatically generated

1. Make sure your reprojected layer is selected as the input layer. In this example, we are splitting the shapefile into regions, denoted by the GID\_1 variable, so select GID\_1 as the Unique ID field.



In the ‘Output directory’ input, select a folder where the split shapefile layers will be saved. We will use these layers for later steps.



1. Click ‘Run’. After QGIS has finished this operation, check the directory you specified in the previous step to confirm that the shapefile layers were saved there.



### Create buffered files.

In this section, we will (1) **buffer** each file; (2) use the **difference** tool to isolate the exterior buffer; (3) **merge** the exterior buffer and the original file; (4) **dissolve** the exterior buffer with the original polygons.

* First, the initial **buffer** step. From the processing toolbox, search for buffer and find the buffer operation under the ‘Vector geometry’ heading. Right click Buffer, then click ‘Execute as Batch Process’—batching will allow us to buffer all of the split files that we created in the previous section.

A screenshot of a computer

Description automatically generatedA screenshot of a toolbox

Description automatically generated

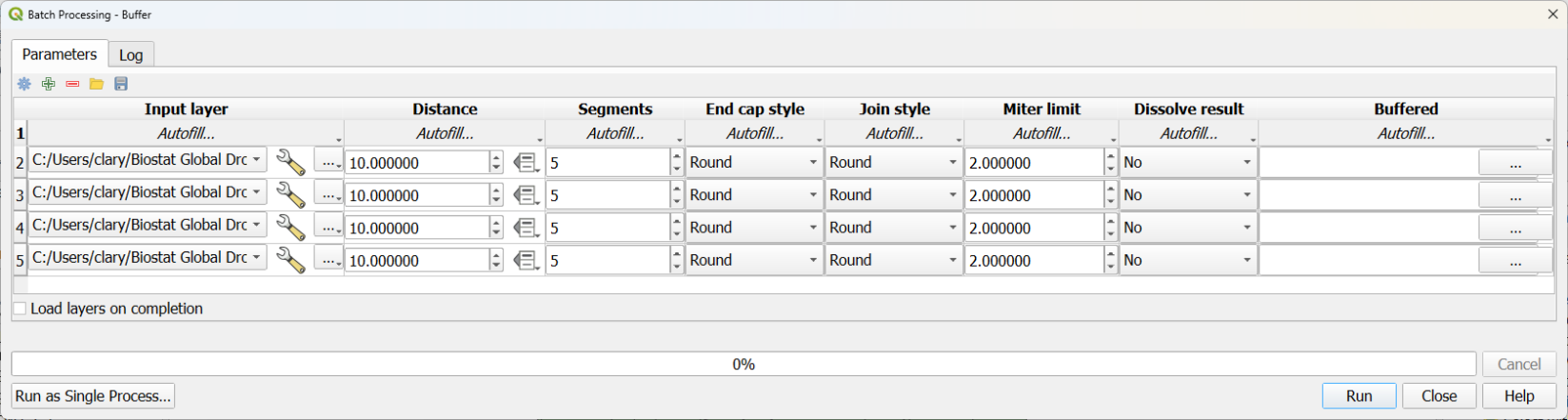
In the batch processing window, we want to add all the split files in the ‘Input layer’ section. Click ‘Autofill’ under ‘Input layer’ and then click ‘Select Files’. Then, select the four shapefile layers created in step (a).

A screenshot of a computer

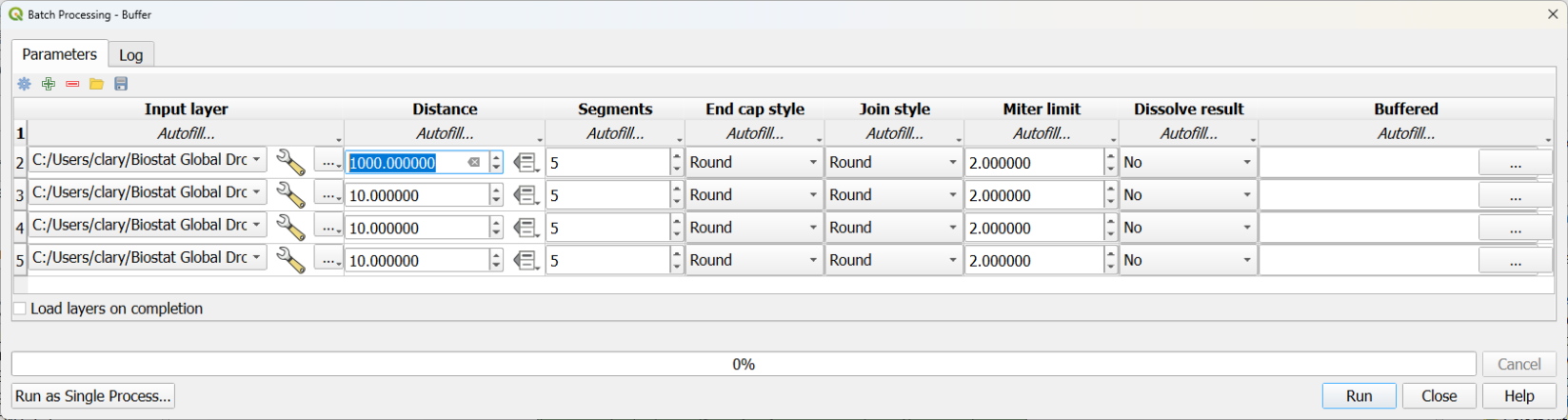
Description automatically generated

When we split our shapefile initially, all the files were named according to their value of the GID\_1 variable. We want to be sure to maintain this naming convention when generating new files in the remaining steps in this section.

Once you’ve selected all your files and clicked ‘Open’ you should see them listed in the ‘Input layer’ column in the Batch Processing window.

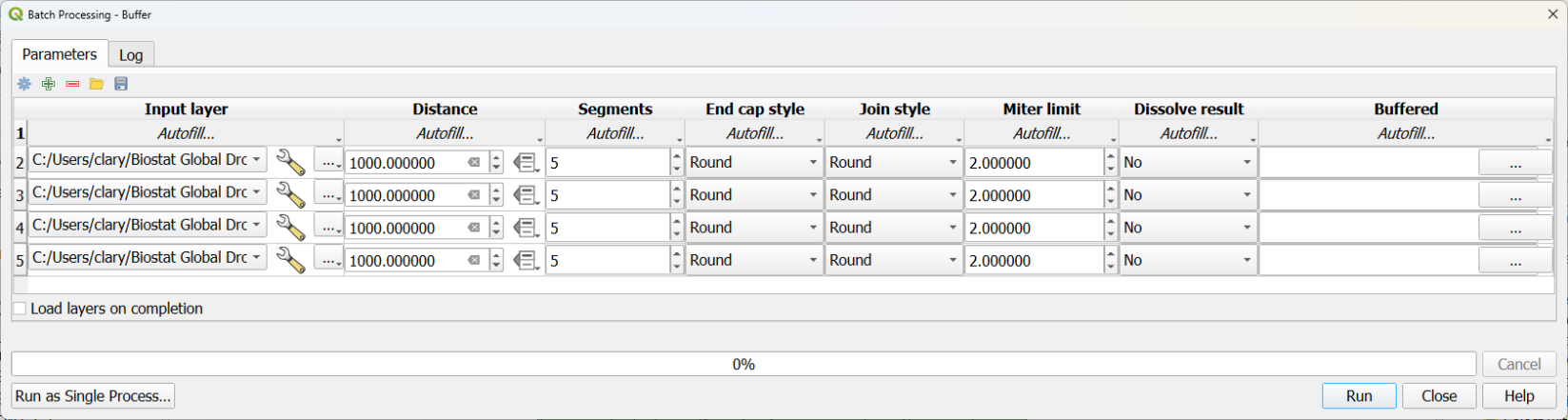


Next we’ll change the Distance parameter to create a buffer of 1 kilometer (1000 meters). Change the distance parameter in the *first* row to 1000.

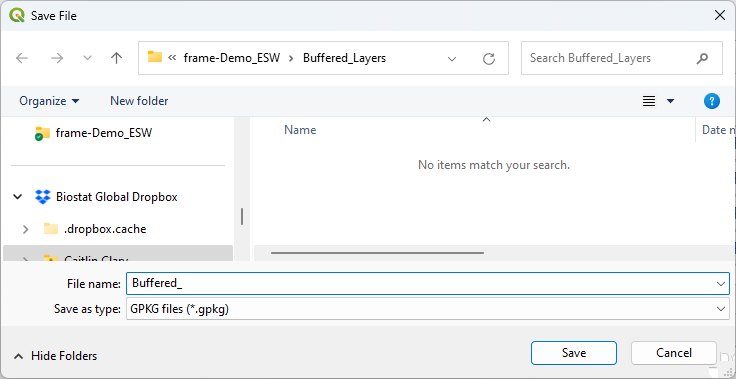


A screenshot of a computer

Description automatically generatedThen click the autofill button in the Distance column and click ‘Fill Down’. This will populate the value ‘1000’ for all of the remaining rows.



The last column in the batch processing window, called ‘Buffered’, is where output file names are specified. In the ‘Buffered’ column click the … button in the first row. This will open a file browser. Navigate to the folder where you want the buffered files to be saved, and then in the ‘File name’ section enter what you want the *beginning* of your file names to be. In this example, all my output file names will start with ‘Buffered\_’.

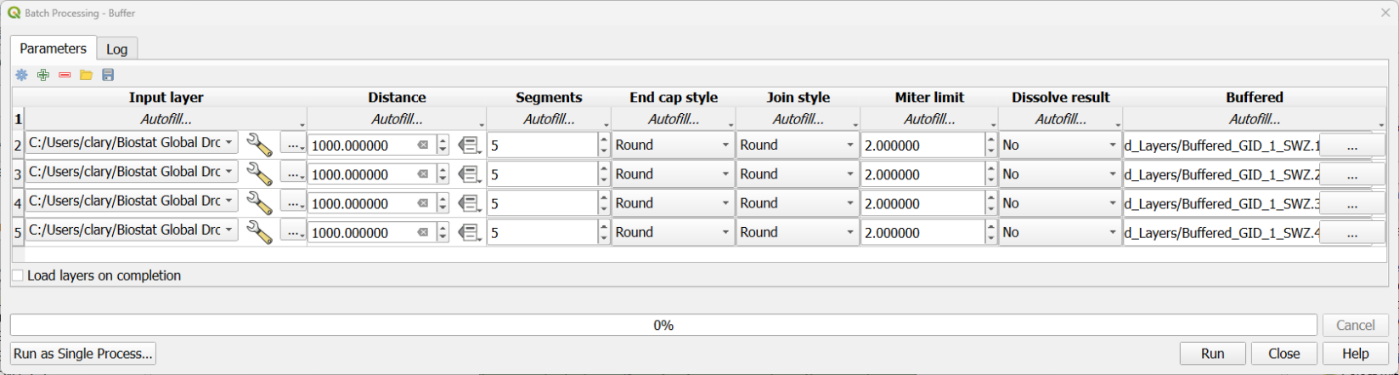


Click ‘Save’. An ‘Autofill settings’ pop-up will appear. In this pop-up set ‘Autofill mode’ to ‘Fill with parameter values’. Then set ‘Parameter to use’ to ‘Input layer’. Click ‘OK’.

A screenshot of a computer

Description automatically generated

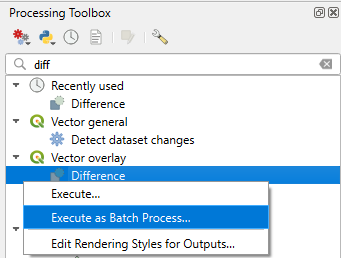
Now the ‘Buffered’ column will be populated with file paths and file names that correspond to the input files selected.



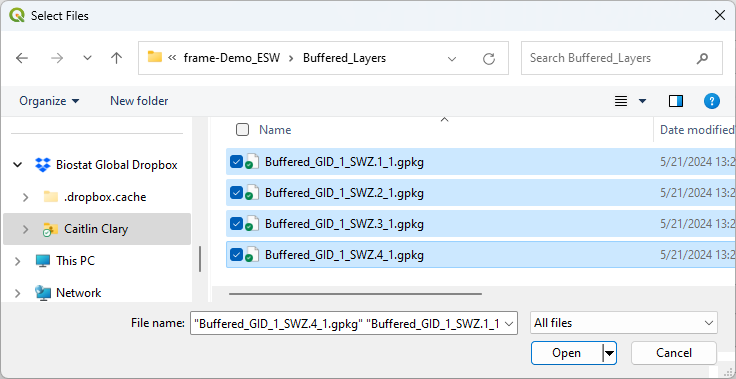
A white rectangle with black text

Description automatically generated

Once these output file paths are created, click ‘Run’ to create the buffered files.

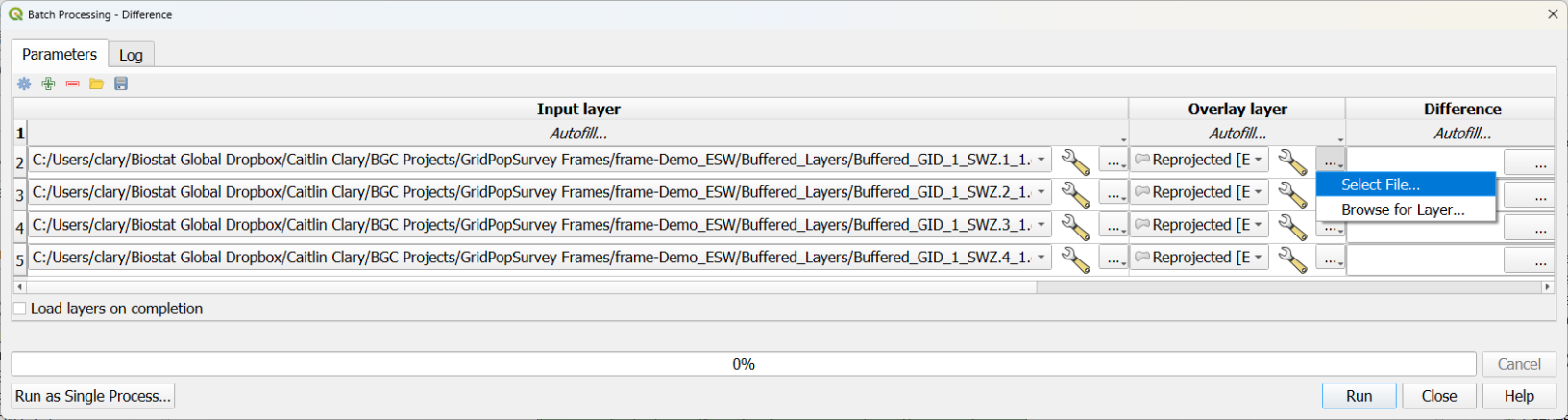
* The buffering steps above add a buffer to *all* boundaries, internal and external. We only want to keep the *external* buffer, so our next step is to use the **Difference** tool to isolate the buffered area outside the original shapefile chunk. In the Processing Toolbox, search for ‘difference’, right click the Difference operation under the ‘Vector overlay’ heading, and select ‘Execute as Batch Process’.

Then, in the batch processing window, select all the buffered files created in the previous step:

A screenshot of a computer

Description automatically generated

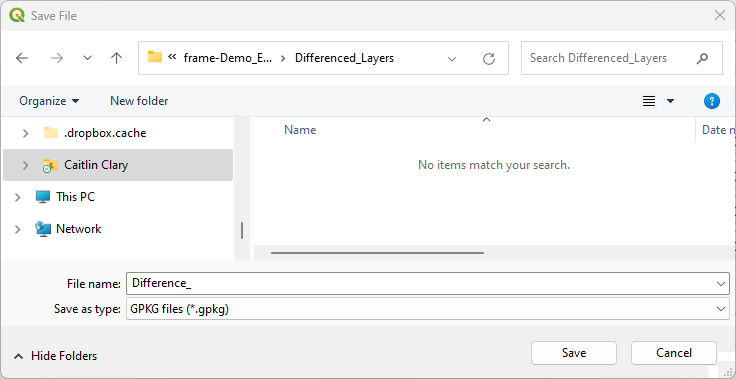
For the Overlay layer, select the *original* non-buffered file appropriate for **each** buffered file, e.g. for the first file here (Buffered\_GID\_1\_SWZ.1\_1) we’ll select the regular GID\_1\_SWZ.1\_1.gpkg file:



A screenshot of a computer

Description automatically generated

Once you’ve selected and double checked all the overlay layers, the column called ‘Difference’ is again where you’ll specify file names for your outputs. Click the … icon by the first row, navigate to the folder where you want to save the differenced files, and enter the beginning of a file name, e.g. ‘Difference\_’. Then click ‘Save’.

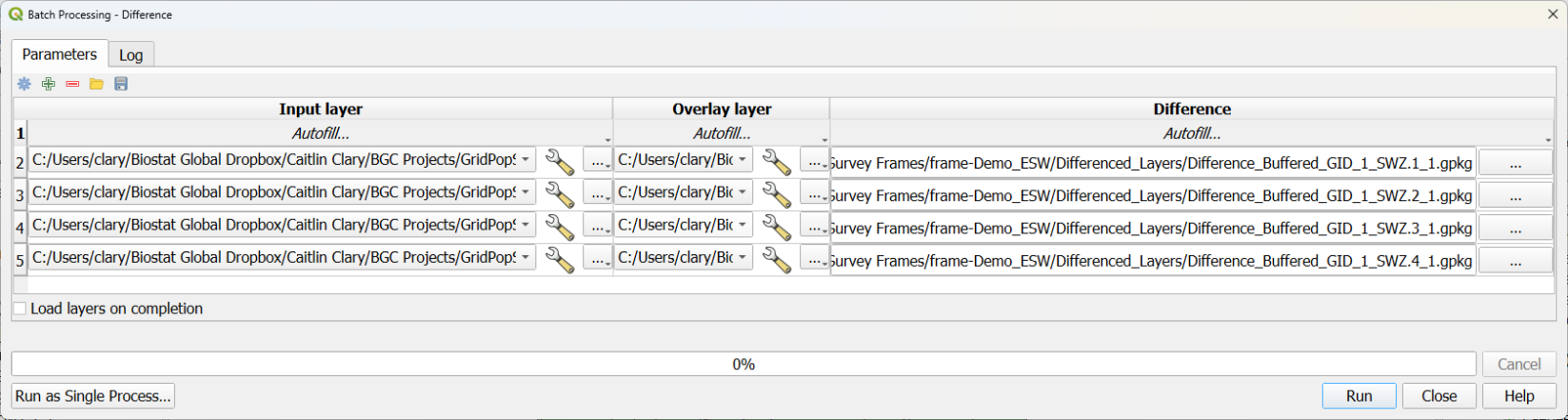


In the ‘Autofill settings pop-up window, again select ‘Fill with parameter values’ and select ‘Input layer’ as the parameter to use.

A screenshot of a computer

Description automatically generated

Then click ‘OK’. The ‘Difference’ column should now be filled with file paths/file names that contain the input file name.



Click Run to create the differenced files. These files consist of buffer areas around the *exterior* of each of the regions, like the yellow rings in the images below:

A blue and yellow outline map

Description automatically generatedA blue and yellow outline map

Description automatically generatedA blue and yellow outline map

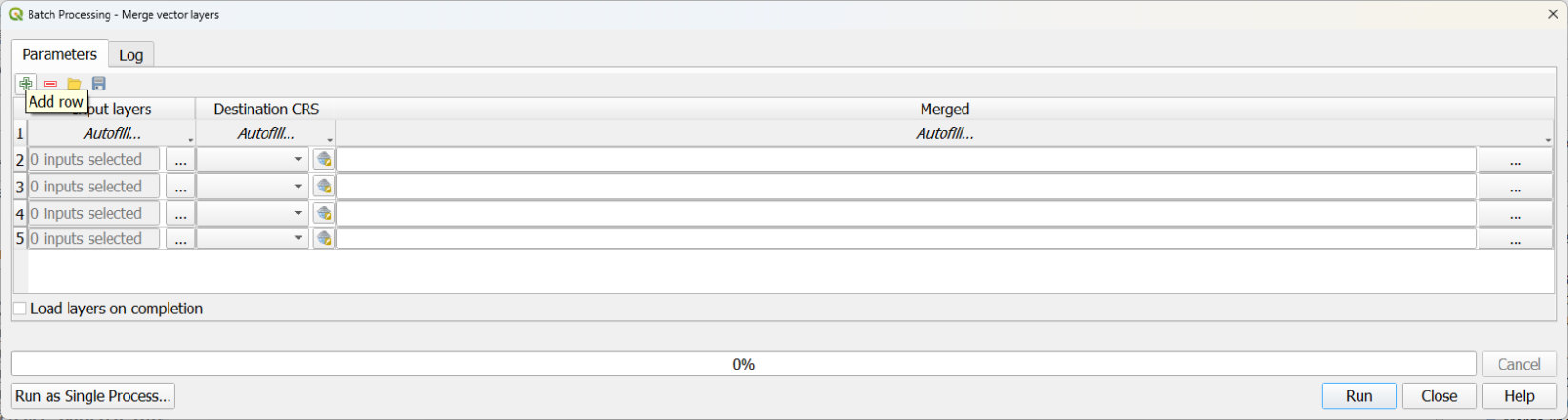
Description automatically generatedA blue and yellow outline map

Description automatically generated

* A screenshot of a computer

  Description automatically generatedThe next step is to **merge** each original non-buffered file with the buffer areas just created. Find ‘Merge vector layers’ in the processing toolbox, right click, and select ‘Execute as Batch Process’.

This will open the batch processing window. Our process for adding files here is a bit different than in previous steps. First, use the **+** icon to add rows—we want one row for each of our regions, so for this example ensure there are four rows for input.

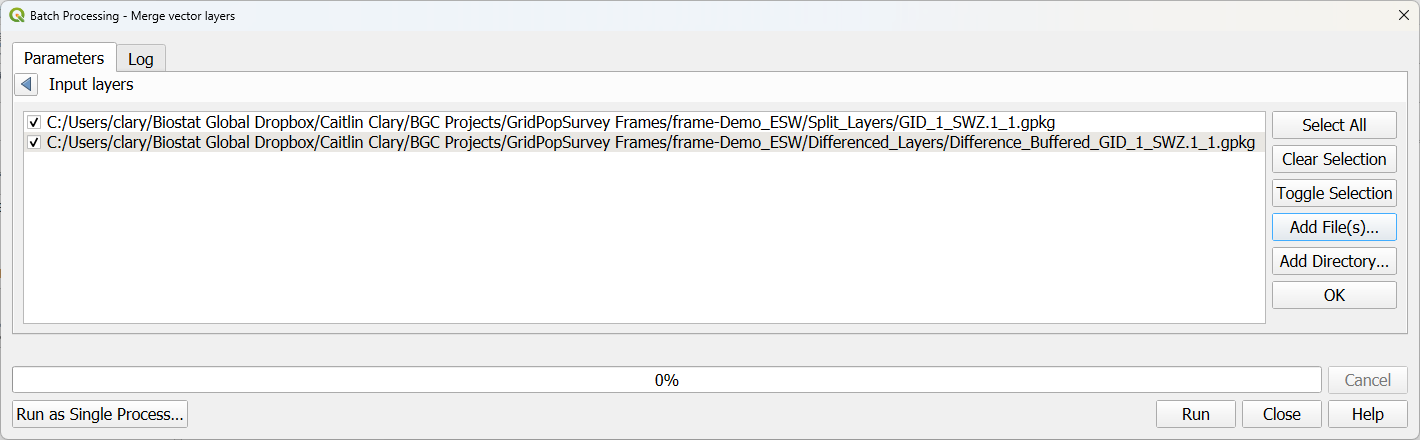


A screenshot of a computer

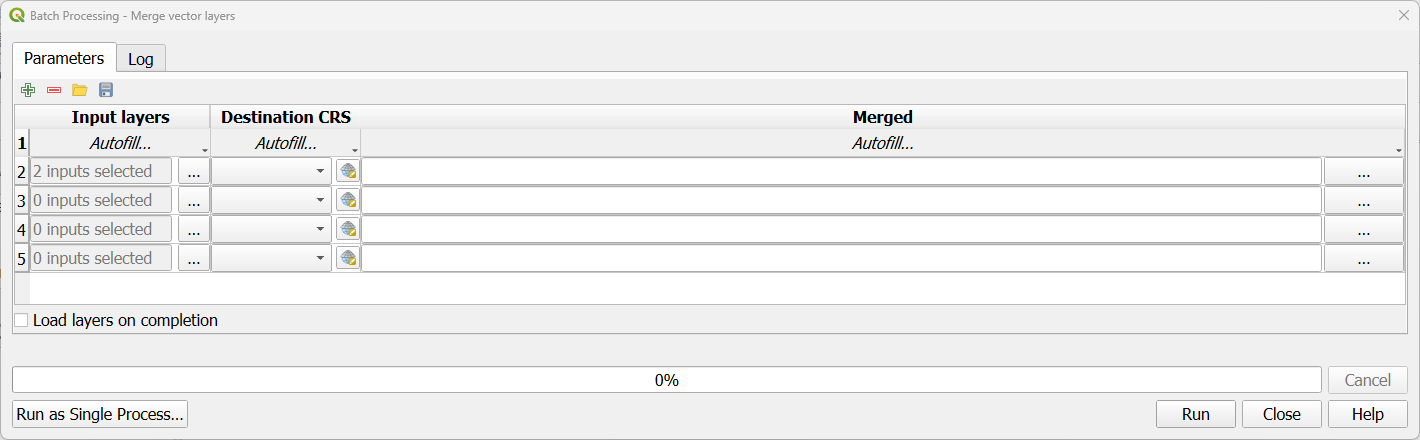
Description automatically generatedIn the first row, click the … icon under ‘Input layers’.

Then, in the window that appears, click ‘Add File(s)’ from the menu on the right. Select the *original* split file and the *difference* file for GID\_1\_SWZ.1\_1.

Make sure both of these files are selected, then click **OK** (do *not* click Run yet—we still need to select the inputs for the other three regions).



Now the first row should have two input layers selected:



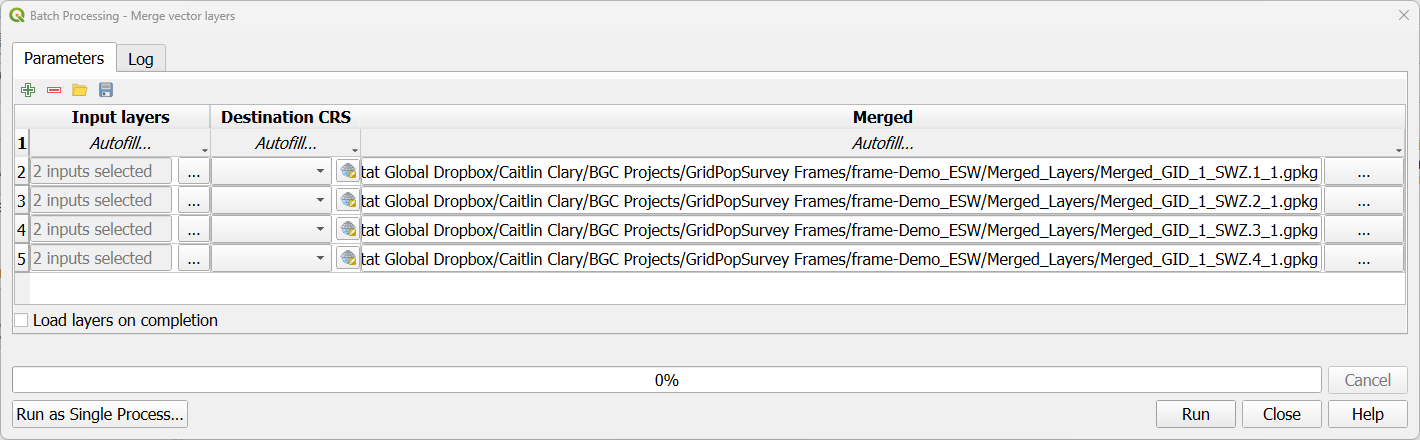
Repeat this process for the remaining rows: for each region, select the original split file and the difference file.

A screenshot of a computer

Description automatically generated

Once inputs have been selected for each row, create file names in the ‘Merged’ column. Typically I manually specify file paths at this stage, as the autofill process used to create file names in previous steps does not work when there are multiple input layers.

In this example, we selected the GID\_1\_SWZ.1\_1 files in the first row, so to keep organized it’s helpful for the file name for that row to contain ‘GID\_1\_SWZ.1\_1), and for the file name for the next row to contain ‘GID\_1\_SWZ.2\_1’, and so on. Because this process is manual, make sure to carefully double check your output file names.



Then click Run. This produces output where the buffer is merged with the original polygons.

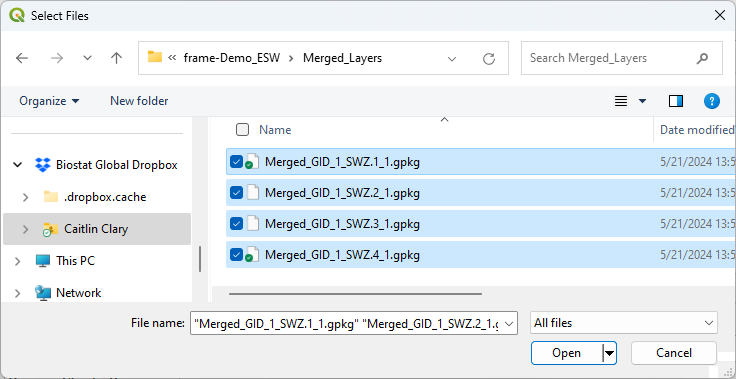
* A screenshot of a computer

  Description automatically generatedWe want the buffer to become part of the polygons it touches, so we’ll now take our final step of **dissolving**. Find ‘Dissolve’ under the ‘Vector geometry’ section in the processing toolbox, right click, and select ‘Execute as Batch Process’.

Select all the merged files from the previous step:

A screenshot of a computer

Description automatically generated



We want to dissolve by our *most detailed* admin boundary, which in this case is GID\_2. In the first row, click the … icon for Dissolve field(s) and select the variable to dissolve by.

A screenshot of a computer

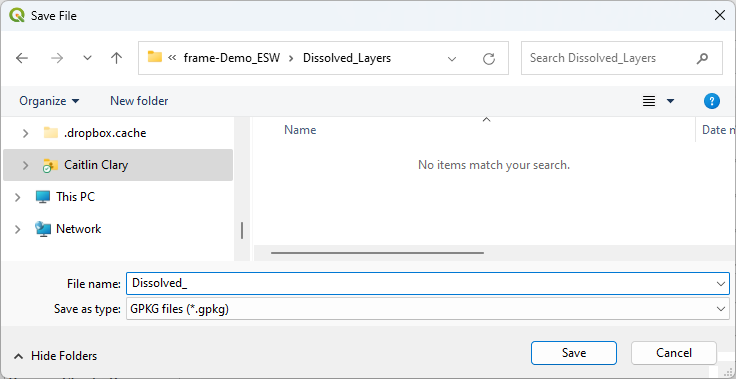
Description automatically generated

Click OK. Then click Autofill and Fill Down

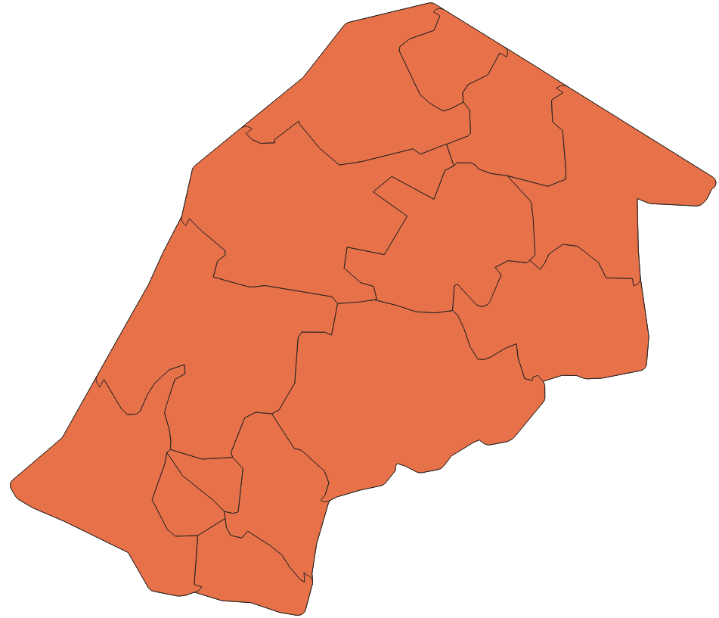
A screenshot of a computer

Description automatically generated

Finally in the ‘Dissolved’ column specify file names for your output. As with earlier steps, start by selecting a file location and the beginning of a file name:



Then use the Autofill settings window to use the input layer as a parameter for the output file names:

A screenshot of a computer

Description automatically generated

Finally, click Run.

This final step produces layers with an exterior buffer, but with *internal* boundaries maintained.

## Move to R to run GridEZ

Download the GridEZ function (*gridEZ\_fn\_public\_release\_v3.R*) from GitHub.[[1]](#footnote-1)

Download *Demo – create sampling frame with multiple gridEZ runs.R* from GitHub and save this script in the working folder where you’ve saved the shapefiles from previous steps.

This example uses:

* Eswatini buffered region shapefiles created in the previous step
* Gridded population from [GHS-POP R2023A](https://ghsl.jrc.ec.europa.eu/ghs_pop2023.php) E2030
* Gridded settlements from [GHS-SMOD R2023A](https://ghsl.jrc.ec.europa.eu/ghs_smod2023.php) E2030

Ensure that you have the gridded population dataset and the gridded settlement dataset downloaded on your computer.

### Modify file paths in the R script

In the R script downloaded from GitHub, update the file paths (highlighted below) to point to the appropriate folders and files on your computer.

# Prevent the sp package from calling rgdal

options("sp\_evolution\_status" = 2)

# Load packages

library(sf)

library(sp)

library(raster)

library(parallel)

library(igraph)

library(tidyverse)

# Setup ----

# GridEZ function

source("Q:/GridPopSurvey Frames/2.CODE-gridEZ-master/gridEZ\_fn\_public\_release\_v3.R")

# Output folder

output\_dir <- "Q:/GridPopSurvey Frames/frame-Demo\_ESW/Output/"

# Buffered shapefile folder

buffer\_dir <- "Q:/GridPopSurvey Frames/frame-Demo\_ESW/Dissolved\_Layers/"

# Gridded population raster file path

gridded\_pop\_path <- "Q:/GridPopSurvey Frames/3.DATA-global-GHSL/GHS\_POP\_E2030\_GLOBE\_R2023A\_54009\_100\_V1\_0/GHS\_POP\_E2030\_GLOBE\_R2023A\_54009\_100\_V1\_0.tif"

# Gridded settlement raster file path

gridded\_sett\_path <- "Q:/GridPopSurvey Frames/3.DATA-global-GHSL/GHS\_SMOD\_E2030\_GLOBE\_R2023A\_54009\_1000\_V1\_0/GHS\_SMOD\_E2030\_GLOBE\_R2023A\_54009\_1000\_V1\_0.tif"

if (!dir.exists(output\_dir)){dir.create(output\_dir)}

# Specify number of cores to use during parallel processing and type of parallel processing

ncores <- detectCores() - 1

par\_type <- "PSOCK" # parallel processing type - "PSOCK" for windows

# Projections

proj <- "+proj=longlat +ellps=WGS84 +datum=WGS84 +no\_defs"

proj.moll <- "+proj=moll +lon\_0=0 +x\_0=0 +y\_0=0 +ellps=WGS84 +units=m +no\_defs"

### Load data and set up for loop.

Create a data frame to structure the for loop for the individual GridEZ calls. In this example, we are splitting Eswatini by region (**GID\_1**) and creating a GridEZ call for each region.

The data frame will have one column called loop\_id, containing the unique values of GID\_1 from the Eswatini shapefile:

# Load population data

pop\_full <- raster(gridded\_pop\_path)

# Load settlement layer

settlement\_full <- raster(gridded\_sett\_path)

# Data frame to define gridEZ loops

loop\_var <- "GID\_1"

loop\_df <- data.frame(

loop\_id = c("SWZ.1\_1", "SWZ.2\_1", "SWZ.3\_1", "SWZ.4\_1")

)

In the for loop section of the script, ensure the file paths for your buffered shapefile inputs are correct (the file names you’ll use in this script will depend on the file names you created when saving the buffered files)—see the highlighted sections below.

Note that this script is set up to allow GridEZ to be run on multiple computers without creating file conflicts, so each loop checks for whether files for this loop have already been created.

The remainder of each loop processes the population and settlement rasters for the loop, creates a raster version of the boundaries file, and runs GridEZ. Any errors with any of the gridEZ calls are captured and stored in the object called ‘errs’.

errs <- NULL # track any failed loops

for(i in 1:nrow(loop\_df)){

# Set run ID

loop\_runid <- paste0("\_", loop\_var, "\_", loop\_df$loop\_id[i])

if (

!file.exists(paste0(output\_dir, "EZ\_raster\_master", loop\_runid, ".tif")) &

!dir.exists(paste0(output\_dir, "temp\_folder", loop\_runid))

){

boundaries <- sf::st\_read(

paste0(

buffer\_dir, "Dissolved\_Merged\_GID\_1\_", loop\_df$loop\_id[i], ".gpkg")) %>%

sf::st\_transform(., proj) %>%

filter(!sf::st\_is\_empty(.)) %>%

sf::as\_Spatial()

boundaries@data$fid <- 1:nrow(boundaries@data)

boundaries\_moll <- sf::st\_read(

paste0(

buffer\_dir, "Dissolved\_Merged\_GID\_1\_", loop\_df$loop\_id[i], ".gpkg")) %>%

sf::st\_transform(., proj.moll) %>%

filter(!sf::st\_is\_empty(.)) %>%

sf::as\_Spatial()

# Crop population data to boundaries extent

pop <- crop(pop\_full, extent(boundaries\_moll))

pop <- mask(pop, boundaries\_moll)

# Crop settlement layer to boundaries extent, transform to same grid as pop

settlement <- crop(settlement\_full, extent(boundaries\_moll))

settlement <- mask(settlement, boundaries\_moll)

settlement <- resample(settlement, pop, method = 'ngb')

pop\_gridez <- projectRaster(pop, crs = proj)

settlement\_gridez <- projectRaster(settlement, crs = proj, method = "ngb")

# Rasterize boundaries to the same grid as population

boundaries\_gridez <- rasterize(boundaries, pop\_gridez, field = "fid")

tryCatch({

gridEZ(

population\_raster = pop\_gridez,

settlement\_raster = settlement\_gridez,

strata\_raster = boundaries\_gridez,

exclude\_unsettled = FALSE,

using\_ghs\_smod\_pop2015 = TRUE,

predefined\_EZ\_size = TRUE,

EZ\_target\_size = "medium",

output\_path = output\_dir,

run\_ID = loop\_runid)

},

error = function(x){

errs <- c(errs, loop\_df$loop\_id[i])

assign("errs", errs, envir = globalenv())

})

} else {next}

}

## Post-processing

Once GridEZ has finished running all the loops, we’re ready to combine the output and generate a sampling frame.

### Clip files in R

Clip the GridEZ output files using the original chunk boundaries. See the following section of the demo script. Be sure to modify the highlighted sections to use appropriate file paths and file names for your computer. Note that the CRS used when defining my\_crs below is the same CRS selected in step 3(a), EPSG:2054.

# Folder with the original split shapefile layers

boundaries\_dir <- "Q:/GridPopSurvey Frames/frame-Demo\_ESW/Split\_Layers/"

# Data frame to define gridEZ loops

loop\_var <- "GID\_1"

loop\_df <- data.frame(

loop\_id = c("SWZ.1\_1", "SWZ.2\_1", "SWZ.3\_1", "SWZ.4\_1")

)

my\_crs <- 2054

for(i in 1:nrow(loop\_df)){

loop\_runid <- paste0(loop\_var, "\_", loop\_df$loop\_id[i])

# Read raster from gridEZ

loop\_ez <- terra::rast(paste0(output\_dir, "EZ\_raster\_master\_", loop\_runid, ".tif")) %>%

terra::as.polygons() %>%

sf::st\_as\_sf() %>%

sf::st\_transform(., my\_crs)

names(loop\_ez) <- c("gridEZ\_ID\_old", "geometry")

loop\_boundary <- sf::st\_read(paste0(boundaries\_dir, loop\_runid, ".gpkg")) %>%

summarize()

loop\_ez\_crop <- sf::st\_intersection(loop\_ez, loop\_boundary)

orig\_nrow <- nrow(loop\_ez\_crop)

# If any point/line artifacts hiding in a geometrycollection, extract just

# the polygons and reattach to main data frame

if (any(st\_geometry\_type(loop\_ez\_crop) %in% "GEOMETRYCOLLECTION")){

extract\_collection <- loop\_ez\_crop[sf::st\_geometry\_type(loop\_ez\_crop) %in% "GEOMETRYCOLLECTION",]

loop\_ez\_crop <- loop\_ez\_crop[!sf::st\_geometry\_type(loop\_ez\_crop) %in% "GEOMETRYCOLLECTION",]

extract\_poly <- sf::st\_collection\_extract(extract\_collection, "POLYGON")

loop\_ez\_crop <- rbind(loop\_ez\_crop, extract\_poly)

}

loop\_ez\_crop <- sf::st\_cast(loop\_ez\_crop, "MULTIPOLYGON") %>%

mutate(gridEZ\_ID = row\_number(),

gridEZ\_ID = paste0(loop\_df$loop\_id[i], "\_", gridEZ\_ID))

sf::write\_sf(loop\_ez\_crop, paste0(output\_dir, "R\_cropped\_", loop\_runid, ".gpkg"),

append = FALSE)

}

### Calculate area and population in QGIS and handle any PSU fragments from clipping

The final steps will be in QGIS:

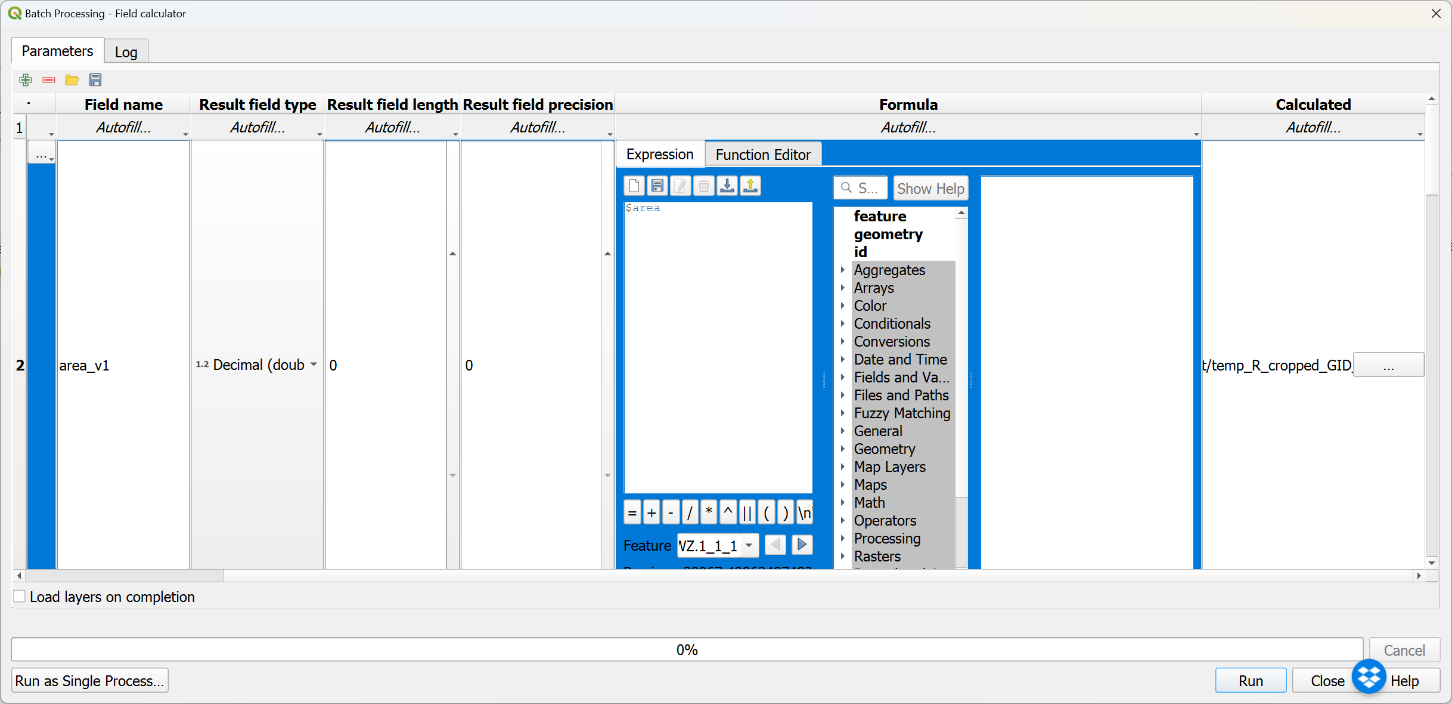
1. Calculate the area of each PSU. From the Processing Toolbox, search for ‘Field calculator’, right click, and click ‘Execute as batch process’. In the batch processing window:

* *Input layer:*add **all the clipped layers created using the script in the previous step**

A screenshot of a computer

Description automatically generated

* *Field name*: enter a name for the area variable to create (e.g. area\_v1), then click ‘Autofill’ and ‘Fill Down’
* *Formula*: in the Expression window, enter $area to calculate the area for each polygon. Then click ‘Autofill’ and ‘Fill Down’.
* *Calculated*: click the three dots icon, add a file name prefix e.g. ‘temp\_’, click OK; when prompted select autofill mode = ‘Fill with parameter values’ and click OK. Then, click Run.



1. In QGIS, calculate population in each PSU:

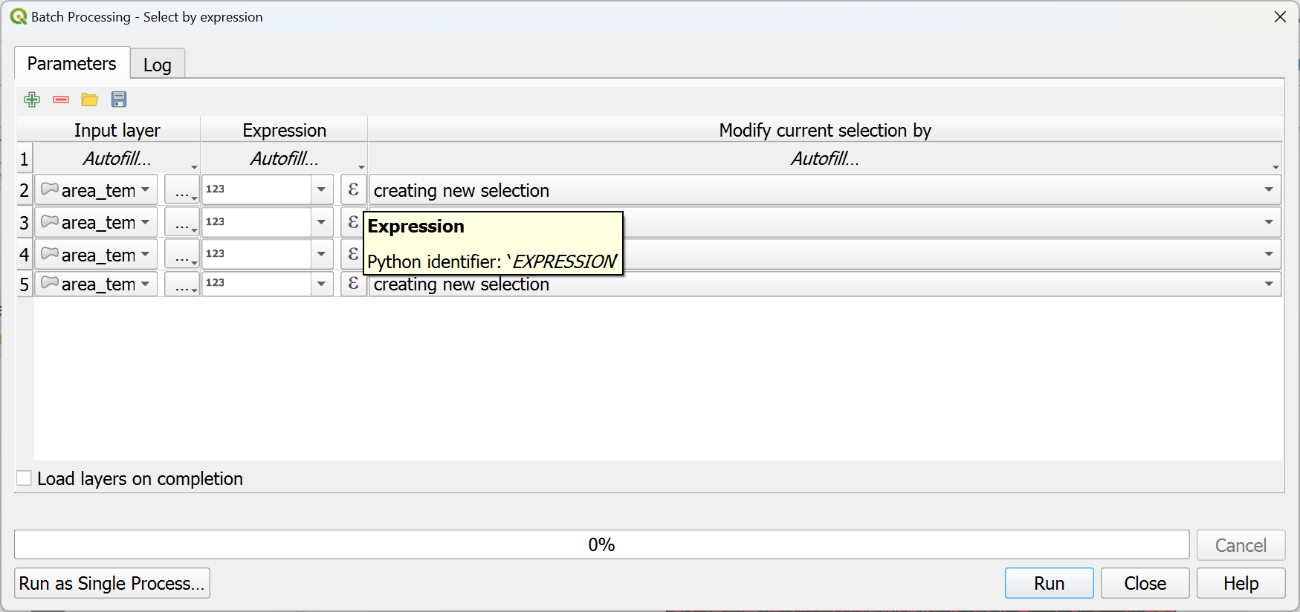
* Load the population raster layer (Layer -> Add Layer -> Add Raster Layer; select the population raster you used in the R script)
* From the Processing Toolbox, find ‘Zonal statistics’, right click, and select ‘Execute as batch process’. In the batch processing window:
  + *Input layer:* Add the layers with area calculations from the previous step
  + *Raster layer:* select the population raster you loaded into QGIS (click Autofill -> Fill Down)
  + *Output column prefix*: prefix for your calculated columns, e.g. ‘v1\_pop\_’ (click Autofill -> Fill Down)
  + *Statistics to calculate*: click the … button and select Sum (click Autofill -> Fill Down)
  + *Zonal Statistics*: click the three dots icon, add a file name prefix e.g. ‘area\_’, click Save; when prompted select autofill mode = ‘Fill with parameter values’ and click OK.
  + Click ‘Load layers on completion’, then click Run.

A screenshot of a computer

Description automatically generated

1. Select PSUs with small area and small population (this will help to remove some small fragments that got created when we clipped the gridEZ output using the original boundaries).
   * From the Processing Toolbox, find ‘Select by expression’, right click, and select ‘Execute as batch process’

* *Input layer*: add the layers with population calculations from the previous step
* *Expression*: in the first row, click the epsilon (Ɛ) button



* In the window that pops up enter: "v1\_pop\_sum" < 50 AND "area\_v1" < 1000000 in the Expression text box, then click OK.

A screenshot of a computer

Description automatically generated

* In the Expression column, click Autofill -> Fill down to populate this expression for the remaining rows. Then click Run.

A screenshot of a computer

Description automatically generated

Polygons matching the expression criteria will be highlighted in yellow in the QGIS interface.

A map of different colored squares

Description automatically generated

1. The next step is to merge the highlighted polygons with a neighbor.

From the Processing Toolbox, find ‘Eliminate selected polygons’, right click, and execute as batch process.

* *Input layer*: select the layers from the previous step.
* *Merge selection with…*: in the first row, select ‘Largest Common Boundary’, then click Autofill -> Fill Down
* *Eliminated*: click the three dots icon, navigate to the folder where these layers should be saved, and enter a file name prefix e.g. ‘elim\_’, and click Save. Then in the Autofill settings prompt, set Autofill mode to ‘Fill with parameter values’ and select ‘Input layer’ as the parameter to use.

A screenshot of a computer

Description automatically generated

* Click ‘load layers on completion’ and run. The output layers will be loaded into the ‘Layers’ sidebar in QGIS.

1. Merge the layers created in the previous step into a single sampling frame shapefile for the country. From the processing toolbox, find ‘Merge vector layers’ and double click (we do *not* need to run as a batch process in this step).

A screenshot of a computer

Description automatically generated

Select input layers by clicking the three dots icon next to the ‘Input layers’ selector. Under ‘Input layers’ select all the layers created in the previous step.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedThen click Run. The merged layer will be loaded into the Layers sidebar in QGIS, with the name ‘Merged’.

1. With the merged shapefile, recalculate area and population

* Find ‘Field calculator’ in the processing toolbox and double click. Ensure the input layer is the Merged layer from the previous step. Under ‘Field name’ specify a variable name (e.g. area\_v2), and enter $area in the Expression field, then click Run. The output layer will be called ‘Calculated’ in the Layers sidebar.

A screenshot of a computer

Description automatically generated

* Find ‘Zonal statistics’ in the processing toolbox and double click. Set the Input layer to be the ‘Calculated’ layer from the previous step, set the raster layer to be the population raster, specify an output column prefix (e.g. v2\_pop\_), select ‘Sum’ under ‘Statistics to calculate’ and then click ‘Run’. The output layer will be called ‘Zonal Statistics’ in the Layers sidebar.

A screenshot of a computer

Description automatically generated

* *Save* the layer created in this step to your working folder.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Congratulations! You’ve just exported the combined gridded population sampling frame.

1. <https://github.com/caitlinbclary/gridEZ> [↑](#footnote-ref-1)