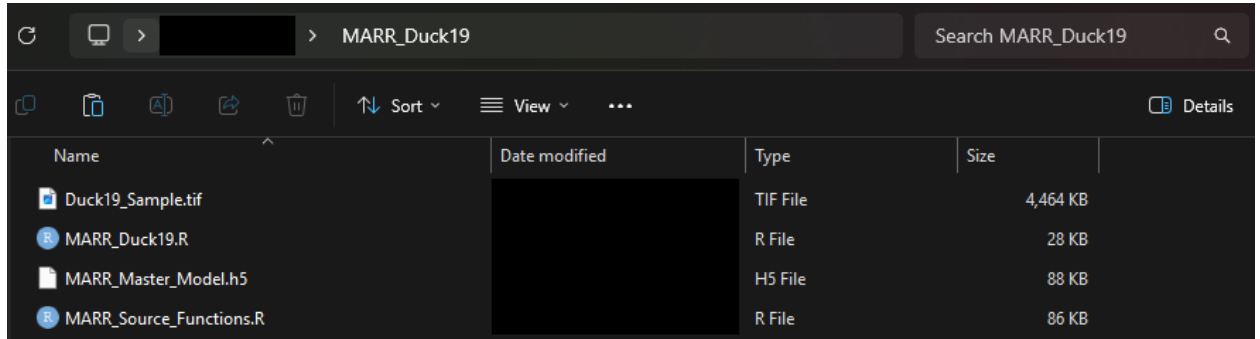


# Example usage of MARR

## Setting up for a Site

1. Prepare a working directory and copy in the 3 master files from the Master folder, rename MARR\_Sample.R based on your preference, siteYR is a good choice. Everything works in the directory based on specific file names so do not change the MARR\_Source\_Functions.R or MARR\_Master\_Model.h5 files. You could also call these files from the Master folder in the MARR\_Sample.R script.



A screenshot of a file explorer window titled 'MARR\_Duck19'. The window shows a list of files with columns for Name, Date modified, Type, and Size. The files listed are Duck19\_Sample.tif (TIF File, 4,464 KB), MARR\_Duck19.R (R File, 28 KB), MARR\_Master\_Model.h5 (H5 File, 88 KB), and MARR\_Source\_Functions.R (R File, 86 KB).

Name	Date modified	Type	Size
Duck19_Sample.tif		TIF File	4,464 KB
MARR_Duck19.R		R File	28 KB
MARR_Master_Model.h5		H5 File	88 KB
MARR_Source_Functions.R		R File	86 KB

2. If keras/tensorflow isn't installed, install them by uncommenting the installation lines, after running the install\_keras() or install\_tensorflow() lines RStudio should be restarted if it is not automatically. Comment these lines after installing the packages, this only has to be run once, so it can remain commented for additional sites. (Tip: ctrl+shift+c will comment/uncomment all highlighted lines)

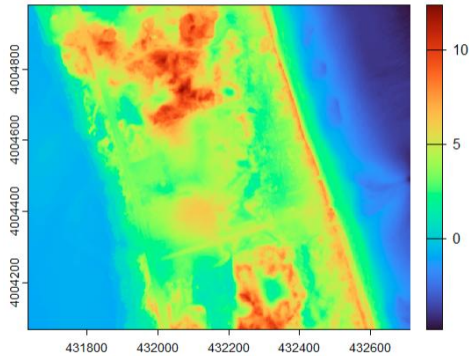
```
# Keras installation -----  
# If keras is not installed run the following lines to install  
# once installed this can be removed or commented out  
#  
install.packages("keras")  
library(keras)  
install_keras()  
  
install.packages("tensorflow")  
library(tensorflow)  
install_tensorflow()  
#  
# -----
```

3. Set up the directory, filenames, and universal values for your respective site. Because the files required are in the working directory defined on line 25, no additional path information is required for loading the data. Only lines 25, 40-43, and 60 need to be changed to account for new data.

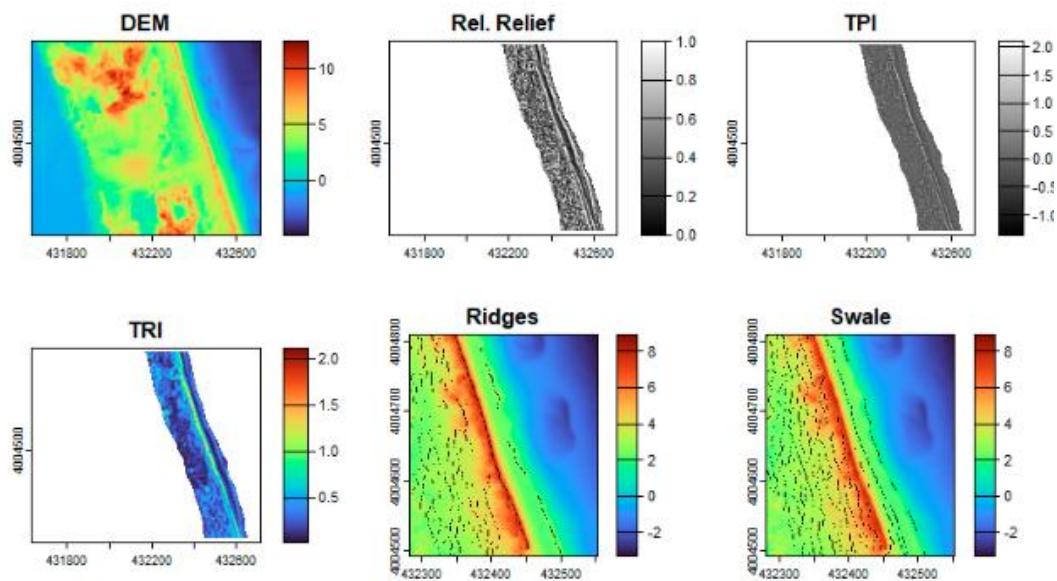
```
24 # Set directory -----
25 dir = "E:\\MARR_Duck19"
26 setwd(dir)
27
28 # Load required packages and Functions and basically everything
29 # if packages not installed (except for keras) they will be installed
30 # through the source file.
31 library(keras)
32 library(tensorflow)
33 suppresswarnings(source("MARR_Source_Functions.R")) # Do not change
34 # -----
35 # Load the master model saved in your working directory -----
36 model <- load_model_hdf5("MARR_Master_Model.h5") # Do not change
37 # -----
38 # =====
39 # Universal name (recommended: siteyr) and other universal variables -----
40 fn.Uni <- "Duck19" # Filename
41 sea.Uni <- "E"      # Direction of sea ("N", "E", "S", or "W")
42 mhw.Uni <- 0.36     # Mean high water for shoreline
43 d2s <- 200          # Distance from shoreline to back of dune mask parameter
44 # -----
45
46 # Create output folders: everything is saved in these folders -----
47 # DO NOT CHANGE THIS
48 nm.Dirs <- c(paste(fn.Uni, "_Preprocess", sep=""),
49             paste(fn.Uni, "_ModelResults_Raw", sep=""),
50             paste(fn.Uni, "_ModelResults_Intermediate", sep=""),
51             paste(fn.Uni, "_ModelResults_Final", sep=""))
52 # Check if folder already exists, if not create it
53 for(d in 1:length(nm.Dirs)){
54   ifelse(!dir.exists(nm.Dirs[d]), dir.create(nm.Dirs[d]),
55         "Folder exists already")
56 }
57 # -----
58 # Load in data -----
59 # DEM
60 M <- raster("Duck19_Sample.tif") # Raster load
61 MT <- rast(M)                    # SpatRaster conversion
```

## Running MARR: Preprocessing

1. Get the scales used (line 145), if the data is large, it is best to keep “perc” small, as well it is useful to run this 2-3 times as it randomly samples the dataset. Shown below Duck has the Atlantic ocean to the east so sea.Uni is “E”.



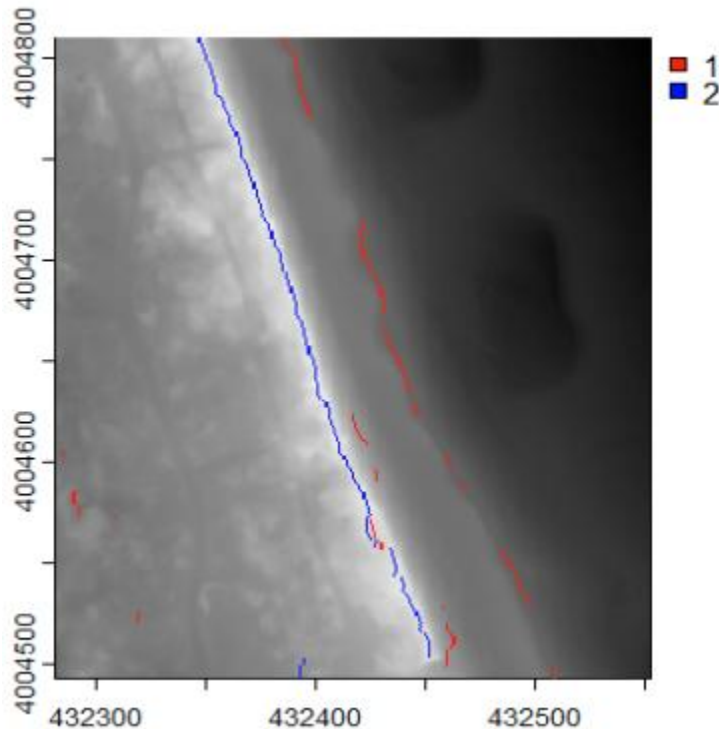
2. Get the variables and processed layers from the DEM (line 166), these are saved to the ‘Preprocessing’ folder within the main directory as backups or for use in separate software. The d2s parameter can be seen in the RR, TPI, and TRI surfaces, only data within d2s from the shoreline are processed to speed up processing time. Create row and column reference matrices (line 190).



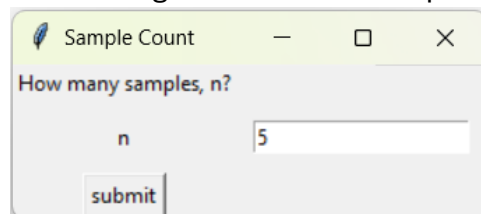
3. Get the ‘features’ of the ridge and swale lines (line205), this can be a slow process if d2s is too large or if the dataset is large.
4. Create data that can be used in the model (line 231).

### Running MARR: Model implementation and training

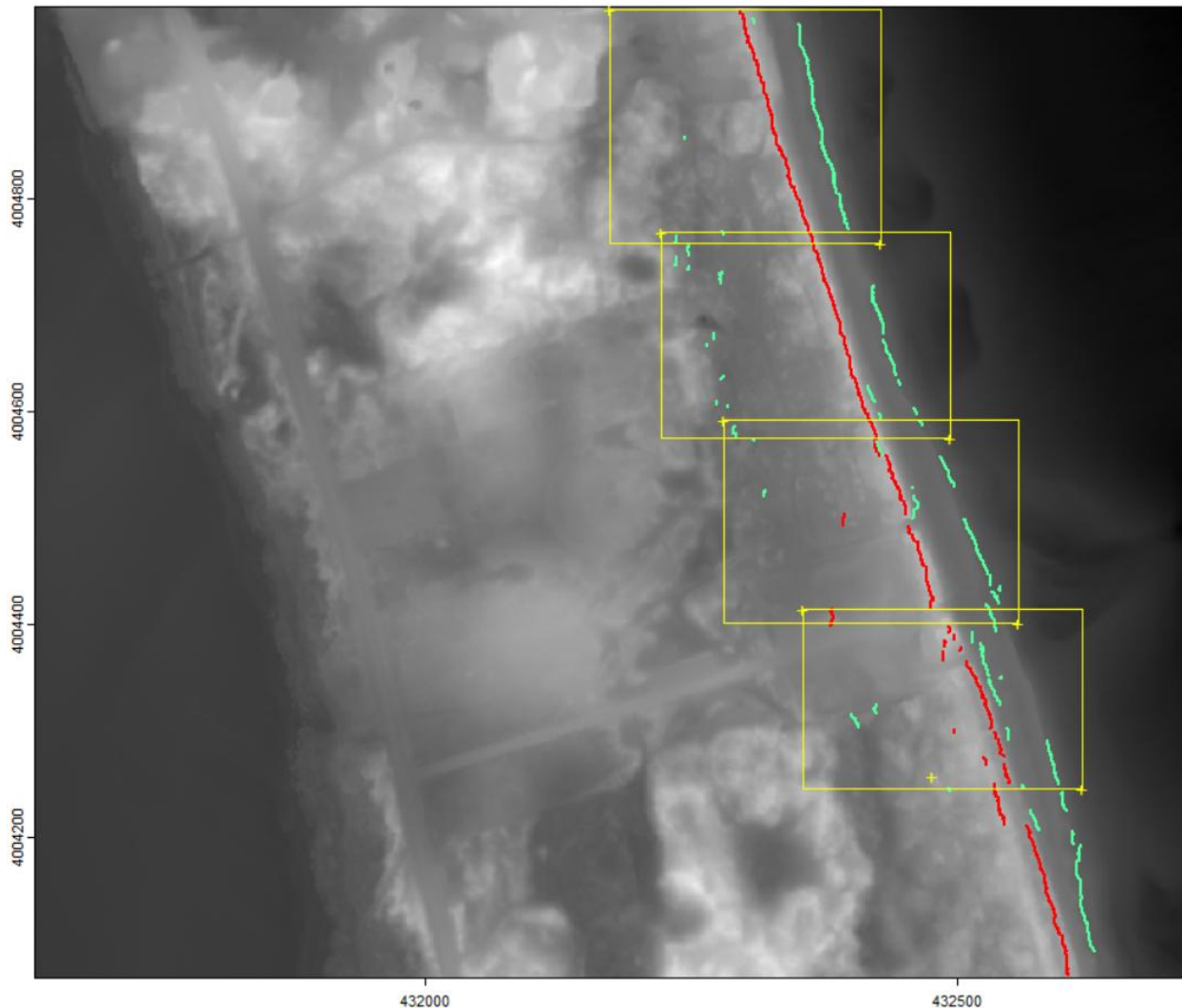
1. Push the data through the model and get preliminary results (line 264).



2. It is highly unlikely that the model returns results that don't require additional training. So, initialize a new dataset that can be used to better train the model on the sites data (line 287), it is not recommended to change the parameters.
3. The easiest way of creating a new training dataset is to select ridge and swale features manually. This requires selecting lines on the map by clicking, which can be difficult to do at the full scale of the DEM. First, create subsamples of the DEM (line 305).
  - a. In the pop-up window set n to be the number of subsamples you wish to make (more subsamples at smaller extents is better). The example dataset is ~1km alongshore so 5 subsamples works well.

A screenshot of a software dialog box titled "Sample Count". The dialog has a standard window frame with minimize, maximize, and close buttons. The main text inside asks "How many samples, n?". Below this text, there is a label "n" followed by a text input field that contains the number "5". At the bottom left of the dialog is a button labeled "submit".

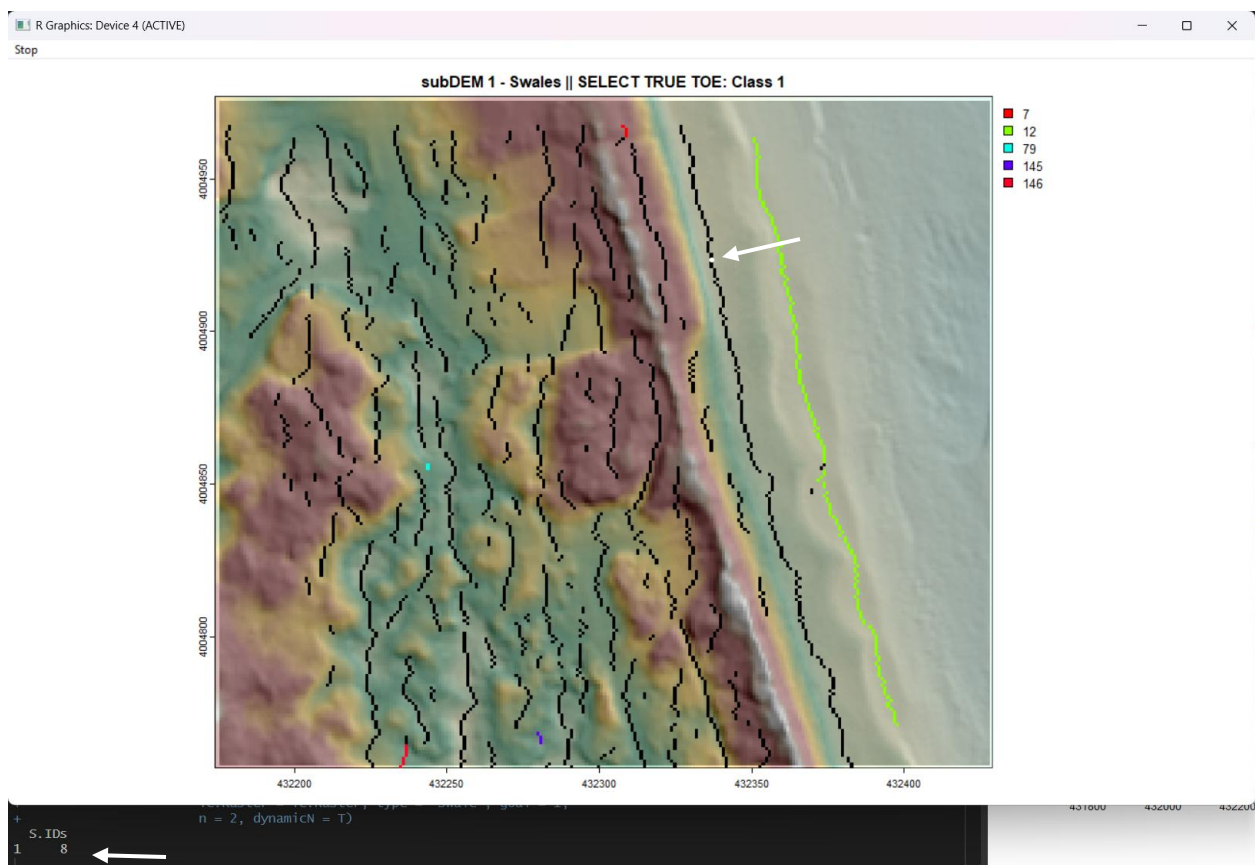
- b. Select the extent of a subsample by clicking the upper left and lower right corners of a rectangle (selected points represented by a +). A small overlap between subsamples is useful. Select subsample space based on the results (red and green points), the model outputs toe, crest, and other classifications so the incorrect locations are useful for the 'other' class.



4. Once the subsamples are selected we can manually select new training samples (lines 343, 347, 351, 355). Pay attention to which variable is being defined, for example, S1 defines swales that represent a toe (class 1), so you want to choose representations of the toe. When the windows open look at the DEM image, black lines represent all features (swale/ridge) available, and colored lines represent model results corresponding to the class (swales = toe, ridges = crest). Each selection made is the ID of the line selected, lines separated by a small (1-2 cell) gap are likely the same ID, however, redundancy is a good idea and you should make multiple selections if in doubt as duplicates are removed. So set your n value to be

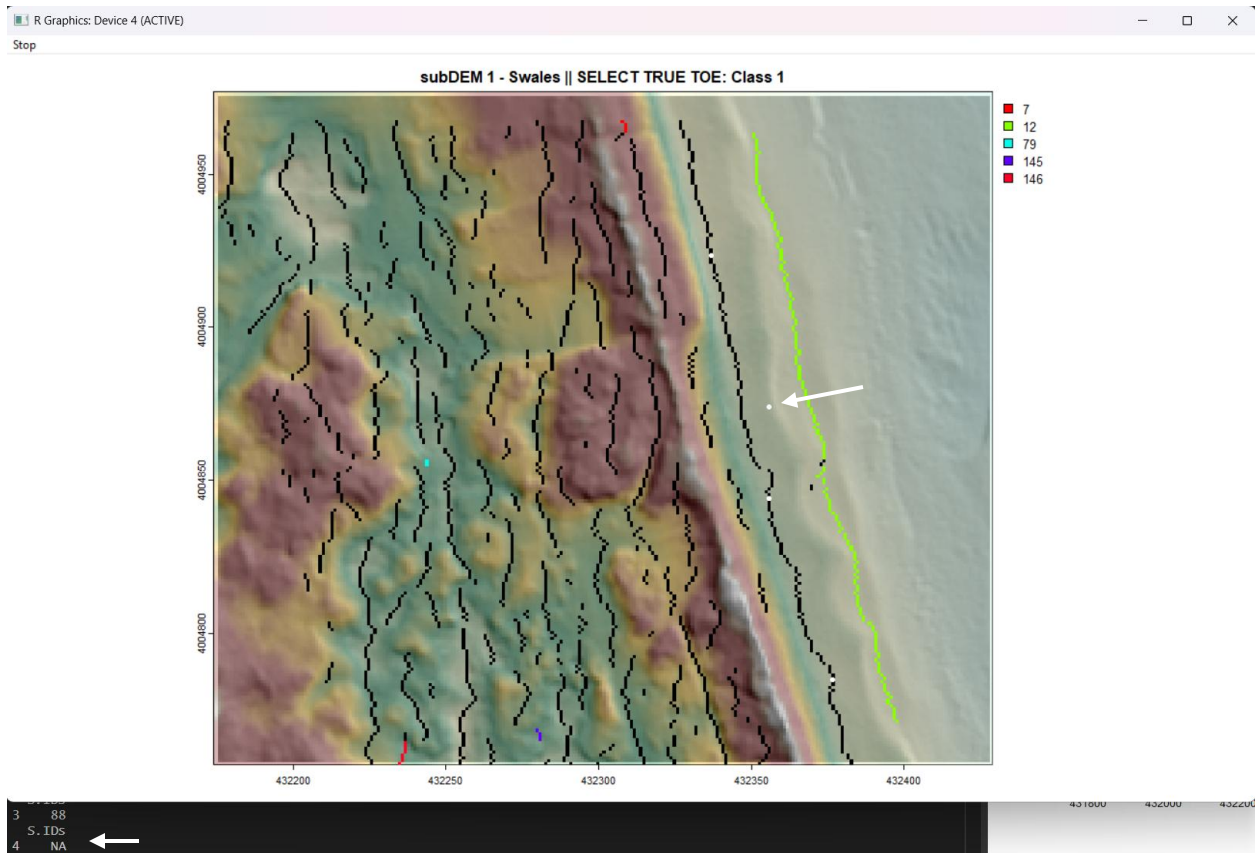
larger than you need. This is also helpful in case you misclick a line and return an NA value. It is also helpful to have the window aligned slightly above the console such that you can see the values selected to know if NA was returned. Also it is important to note that you do not have to select all options, you are building a dataset representative of the features you want the model to learn, so a representative sample is what you want. You can click anywhere on a line.

- a. In this example S1 is being defined so selections should reflect lines that define the toe. If each line at the toe has a unique ID there are ~3-4 choices to make, but it is likely each segment has the same ID, n is set to 5 to be safe. My first selection (represented as the white dot) was on the swale line with ID: 8 (seen in the console below the window).

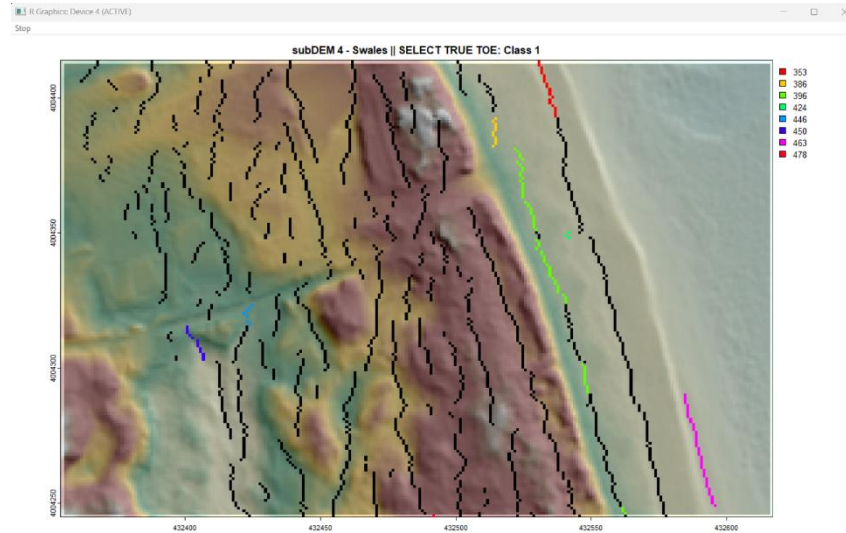




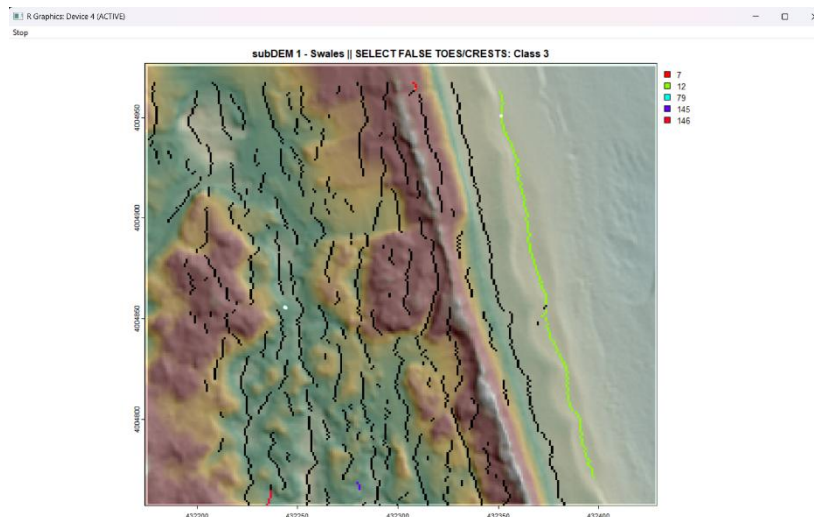
- b. My next 2 selections show that these next line segments all both belong to ID: 88. Therefore, there is nothing to gain from selecting more toe features as all options have been chosen in this subsample. So my final 2 selections (because  $n=5$ ) are on locations with no line to return NA values. Because NAs are removed it is fast to select these locations to exhaust remaining clicks and move to the next subsample.



- c. Do this for all subsamples. In this example (subsample 4) the green line defines a correct classification of a toe, therefore it is useful to select it as a sample. Because we are getting the ID of the line only one selection is needed, you do not have to select the individual green sections.



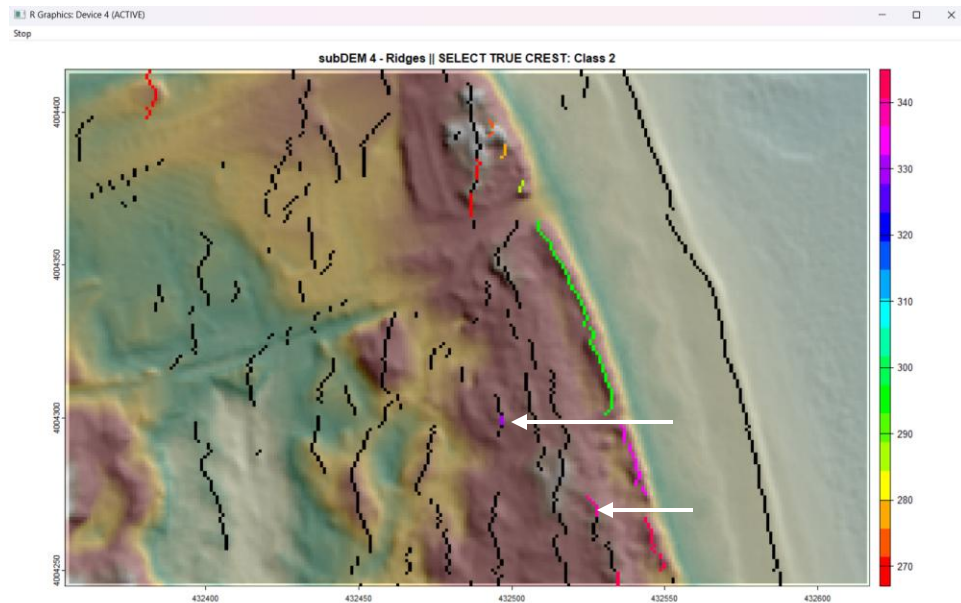
5. Do this for the remaining classes (S3, R2, S3) examples of selection:
  - a. Here we are defining class 3 'other', the dataset is swales so we are defining swales that don't represent a toe. Priority is given to incorrect classifications, in this subsample, 5 lines are classified wrong so we set n=8 to define the 5 misclassifications with extras for misclicks (smaller line sections can be tricky to click) and other samples. But, not all choices to be selected.



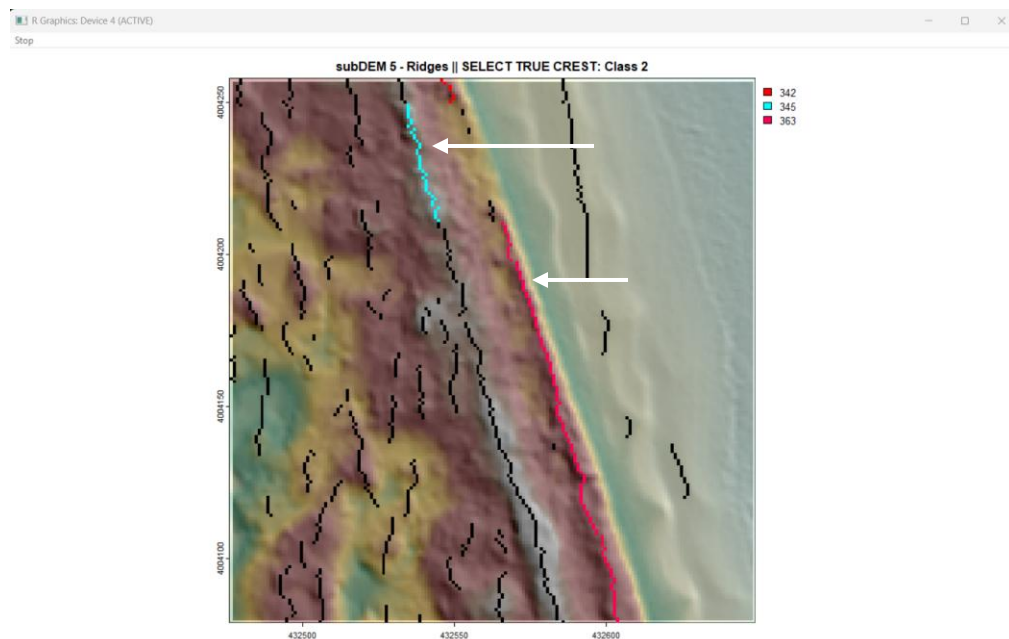
- b. In this example we are selecting locations of the crest, two lines (purple and pink) are classified as crests. They don't align with the primary crest line because no ridge is identified due breaching. Therefore, these could be set



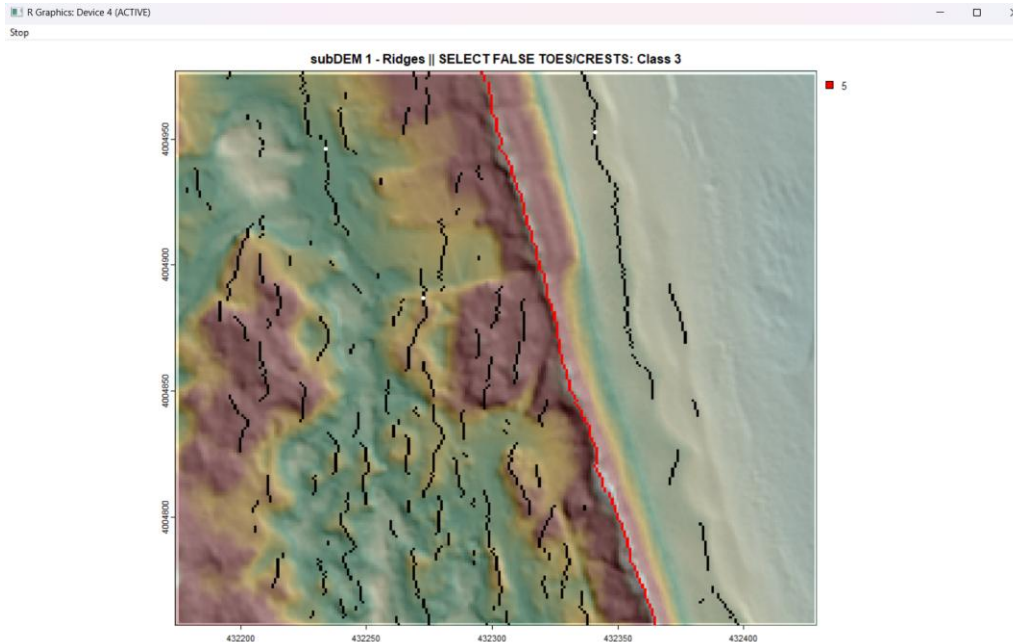
as the crest or they could not, we find that it is simply best to not include these types of patterns in the training dataset.



- c. In the following subsample this same kind of pattern emerges however, here we want to select both lines to encourage both being extracted by the model (the overlap is accounted for in the methodology).



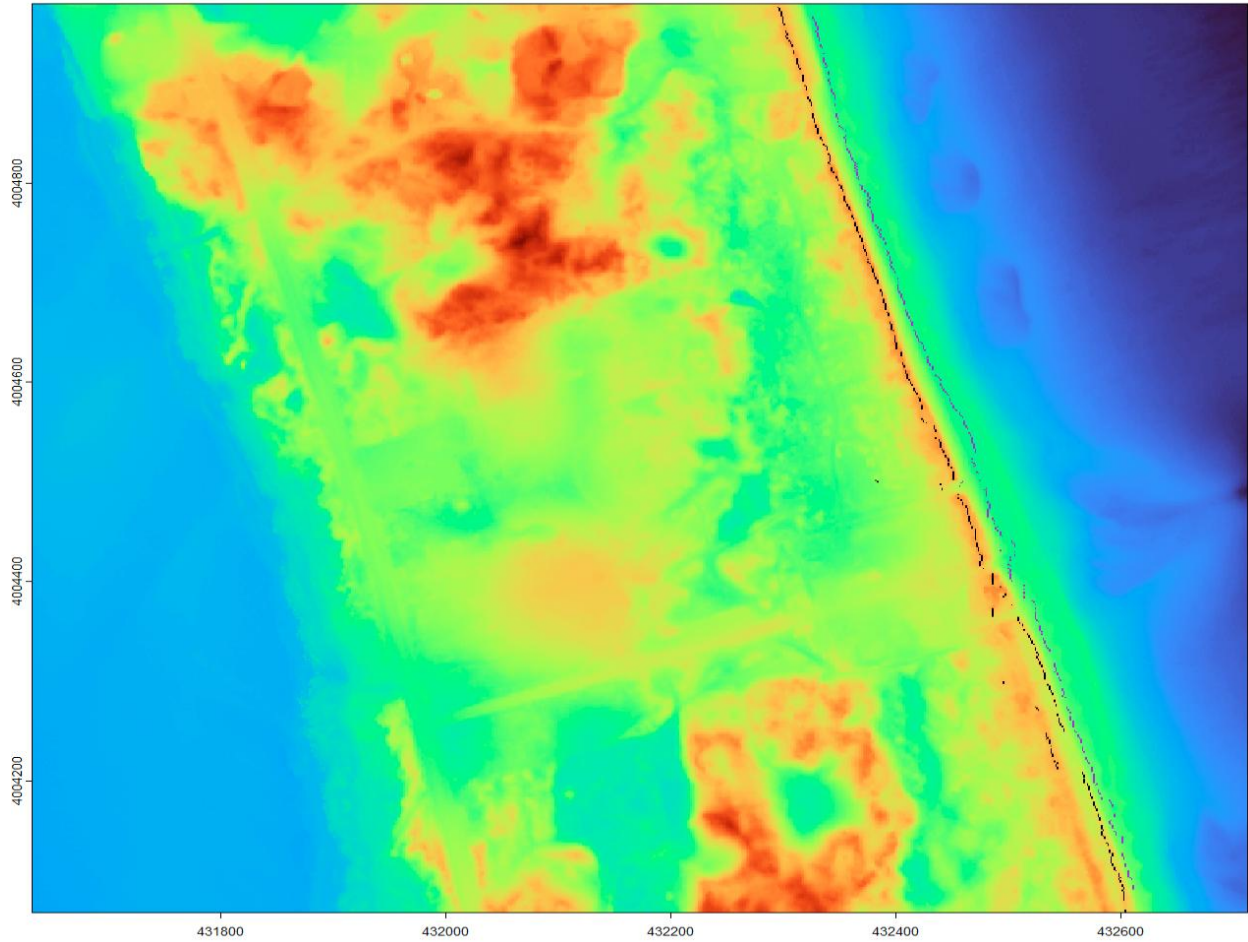
- d. Here we are choosing class 3 for ridges (i.e., ridges that are not the crest) there are no misclassifications, therefore, we set  $n=4$  and select a swath of samples across the site to provide more examples for the model, this is a judgement call but in the first training iteration more data selections is better.



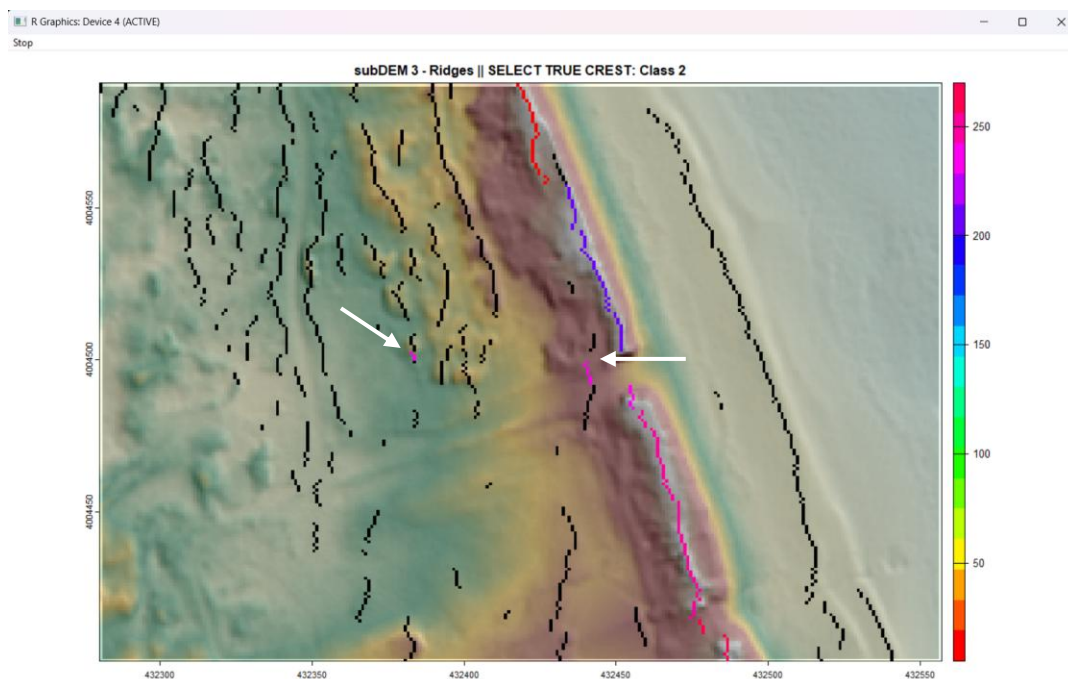
6. After you have created S1, S3, R2, and R3, define their equivalents in lines 360-363 to be used to create the training data (line 376) which is saved in the Intermediate folder (line 383).
7. Train the model using the training dataset (line 388), in our experience there is little gained by adjusting the LL parameter (it defines the minimum length of class 3 examples) unless the site is large, 6 is a reasonable. The model is not sophisticated so no hyperparameters are adjustable and performance is not tracked. This is approximated by the returned contingency we are not worried about overfitting so a perfect return is acceptable. Our model performs well but has a few misclassifications still. The model is saved in the intermediate folder.

	pred		
act	1	2	3
1	10	0	0
2	0	16	2
3	4	1	590

8. Next push the data through the new model and get the results. It is good practice to adjust the iteration value if you wish to track performance. Because this is the first round of training we set iteration=2. Results show that our model performs much better now. It is possible we do not need to adjust the training samples and retrain again, however, we find that it is often worth it to retrain a second time as the selection process is much quicker.



9. In this example it is unnecessary to recreate the subsample DEMs, however, with large datasets you may wish to redo them to better focus on areas of misclassifications which will be identified by green points that are far removed from the correct locations. The process for adjusting the training data is the same as before, except in this iteration you can select fewer samples as your previous selections still exist. Therefore, it is good practice to set  $n=1$  and click an NA position if you do not need to add samples from a subsample. Make sure you run line 394 after running the model to ensure the data being used is current.
- One thing to consider is that in some cases there may be no actual ridge/swale that can be chosen to identify a crest/toe. Shown below the pink line is not the crest, however, there is no actual crest available. Therefore, the model will never identify a crest there so there is no need to fret over training samples for it. We handle these patterns later so simply ignore them. This concept can be applied for any areas that may be insignificant and won't correct with training the model. It is not worthwhile trying to get the model to learn small/specific sections as you gain diminishing returns and the time sink is not beneficial.



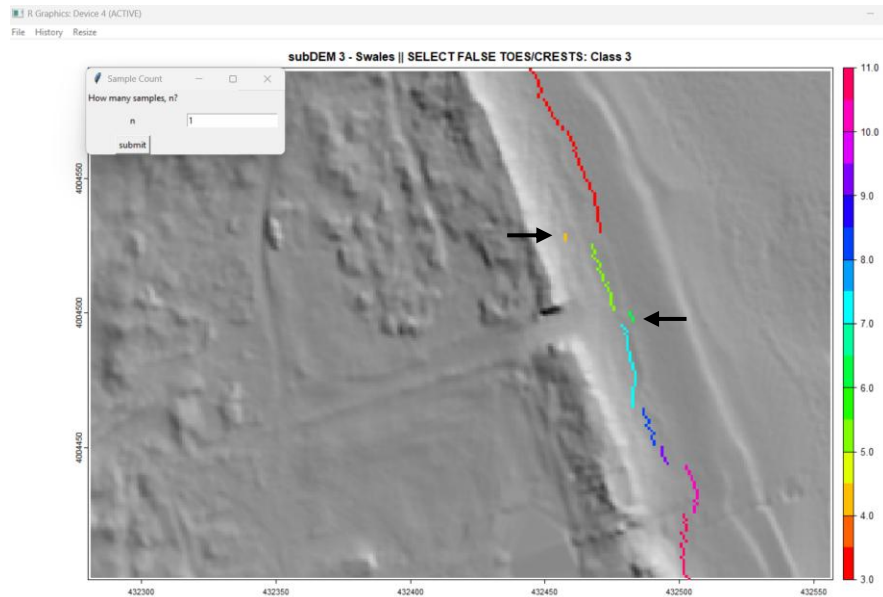
10. Once you have selected your new samples it is important that you run lines 367-370 NOT lines 360-363. This will append your new samples to the originals and remove duplicates. If you run lines 360-363 your original selections will be lost. After that is the same process as before, create the training data (line 376), training the model (line 393), push data through the model (line 393).

### Running MARR: Finalizing results

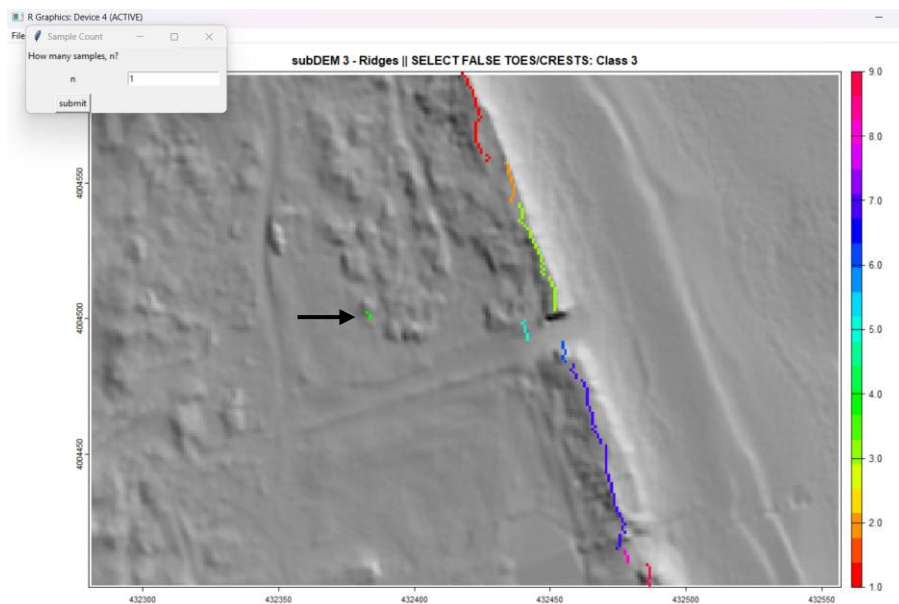
1. After you are happy with the results returned by your model we need to a few extra things to clean up any remaining erroneous classifications. Start by separating the results into the respective toe and crest features (lines 427-447).
2. Next we are going to go through a process similar to selecting training samples. However, this time we select samples that we wish to completely remove. This could be because they are not representative enough but there was no actual option (as seen in 9a above), or maybe there is a lack of continuity in the line, or any other reason. We just want to clean up the results. Because this is likely to be highly localized it is a good idea to recreate your subsample DEMs (line 452). In testing we found that it was best to first go through at a broad extent and repeat the process getting more focused with each iteration. This is highly dependent on the scale of the data and noisiness of the results, some testing/resetting is likely required as you get the hang of it.
3. After you create your DEM subsamples, lines 467 and 471 can be used to select the samples you want to remove. The process is the same as the same as before, you simply click the lines you wish to remove and redundancy is a good choice. Often you won't have anything to remove so set  $n=1$  and click an NA location.



- a. In this example the results are good, however the orange and deep green sections are jumps that cross a small distance. We have found that these are reasonable to remove so we set  $n=4$  and select them.

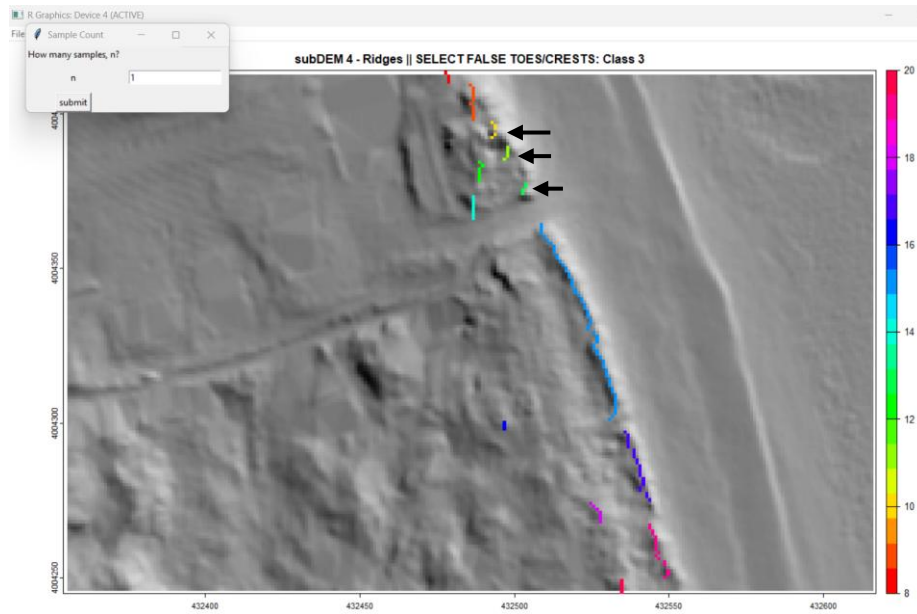


- b. Here we have the instance where no ridge exists to define a crest so we simply remove the classification.

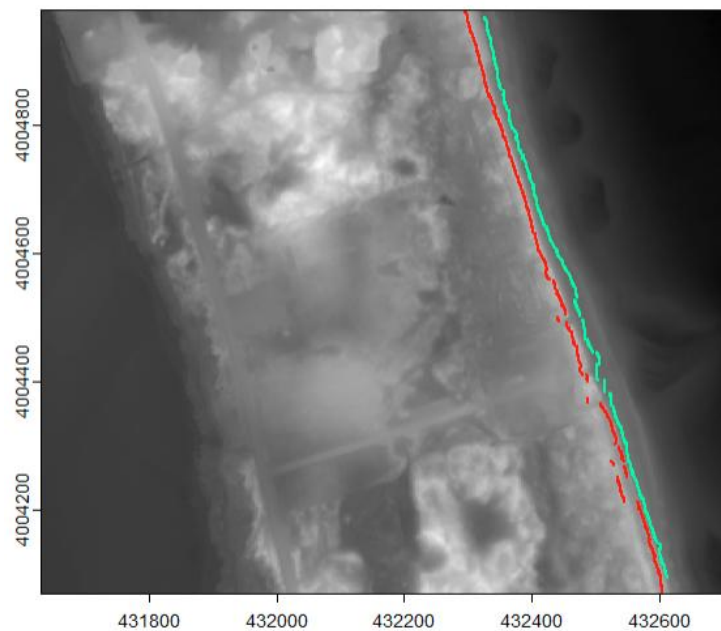




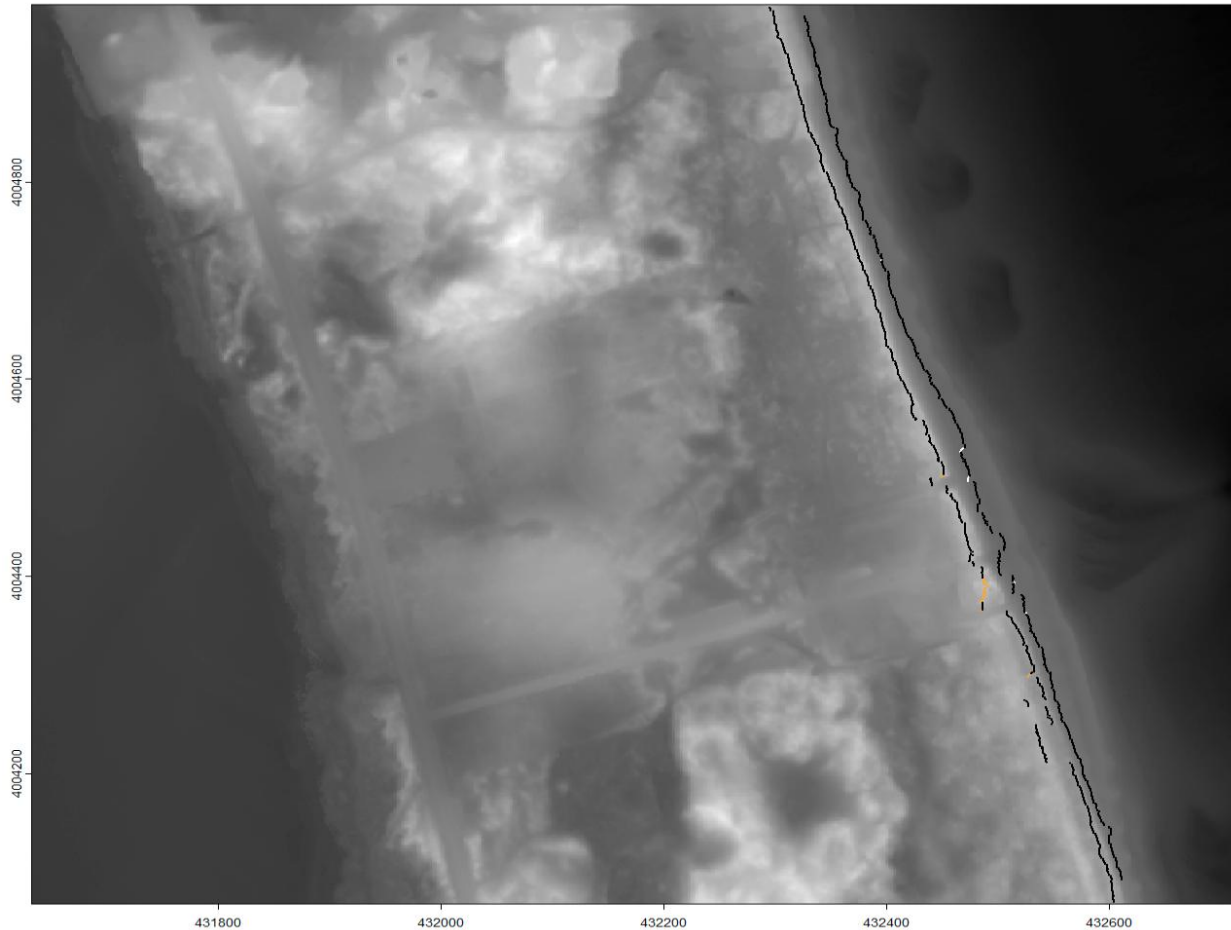
- c. Here we have many small sections that do not completely identify a crest, we find its best to remove these kinds of features as well.



4. After you make your selections remove them from the result using lines 476-480 and then plot them to see if there are any remaining locations to remove. Run the process until you are happy with the removed locations.



5. The areas with missing toe/crest identifications are then filled (line 497). The visualization (lines 508-511) is used to assess the filled locations (white and orange locations are filled, black are the results after removing samples). Testing has shown that R.RNG is best to keep at 5 and S.RNG should be tested for values between 0-5. In our example S.RNG=2.



6. After filling the missing results the last steps are saving the results to the Final folder. Lines 565-581 outputs raster surfaces for all results (TC), crest results, and toe results. Lines 585-598 outputs point shapefiles for all results, crest results, and toe results. Line 605 plots random profiles to a PDF with toe and crest locations included. Lines 608-622 save the data as .CSVs. Finally, line 626, plots a PDF of the alongshore elevation of the toe and crest, as well as the planar positions of both.