



WP3 : MONITORING AND MITIGATION OF COASTAL EROSION

Technical Note

Spatialization of a coastal vulnerability index characterizing the role of coastal ecosystems (mangrove, seagrass beds, coral reefs) in the protection against coastal erosion in the Caribbean by remote sensing

Introduction

Although island coasts are naturally subject to erosion, the current context of climate change and human activities are among the sources of accentuation of this phenomenon. In particular, the increase in the frequency and/or intensity of climatic disturbances (cyclones, swells, rise in sea level, etc.) is responsible for an amplification of this erosion which has a strong impact on the Caribbean coasts and threatens both natural ecosystems, and activities such as tourism. In addition to anthropogenic infrastructure, coastal ecosystems naturally have a role to play in protecting and mitigating the phenomenon of coastal erosion. This is particularly the case for mangroves, sea grass beds and coral reefs. Part of the work carried out by IRD/UMR Espace-Dev in collaboration with Telescop in the WP3 of the CARIBCOAST project consisted in taking advantage of the availability of satellite images and cartographic products from Earth observation to spatialize a coastal vulnerability index (EBCVI for Ecosystem-Based Coastal Vulnerability Index) characterizing the role of these coastal ecosystems in protecting against coastal erosion in the Caribbean.

A vulnerability index aims to simplify a number of complex and interacting parameters, represented by various types of data, into a form that is more easily understood and therefore more useful as a management tool. Based on the literature, we chose to gather variables that define :

- geomorphological characteristics and coastal ecosystems
- the degree to which the coast is exposed to coastal forcing and energy
- information on the socio-economic elements for which coastal erosion represents a risk

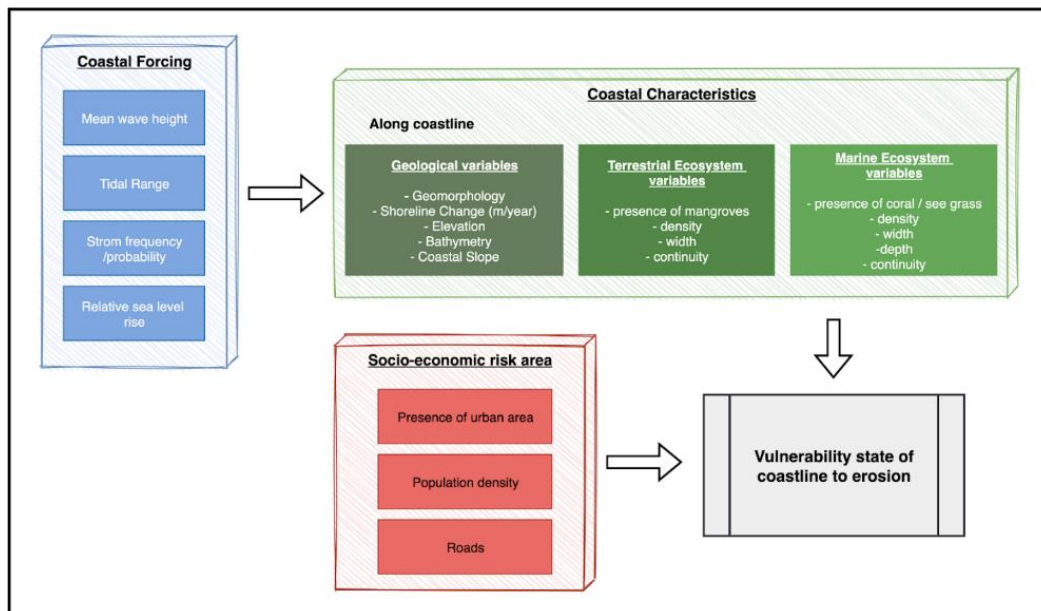


Figure 1 : General approach used to produce the EBCVI in the project

Figure 1 above summarizes the general approach that will be used to produce EBCVI at the project pilot sites (Figure 2).



Figure 2 : Pilot sites of the CARIBCOAST project, including Cul-de-Sac (Guadeloupe).

Input data for the calculation of the EBCVI

For each of the pilot sites of the project, the creation of this spatialized index is based on the use as input data of satellite imagery, digital terrain models and external databases. In particular, the EBCVI calculation method is based on (Figure 3):

- mapping of urban areas and mangroves at 10m spatial resolution from Sentinel 2, produced as part of this project by IRD/UMR Espace-Dev in collaboration with Telescop,
- seagrass and coral reef mapping from the Allen Coral Atlas (<https://allencoralatlas.org/>), and obtained from Planetscope images (<https://earth.esa.int/eogateway/mission/planetscope>),
- the mangrove height map provided by the 2019 Global Forest Canopy Height product (<https://glad.umd.edu/dataset/gedi>) from the GEDI sensor.

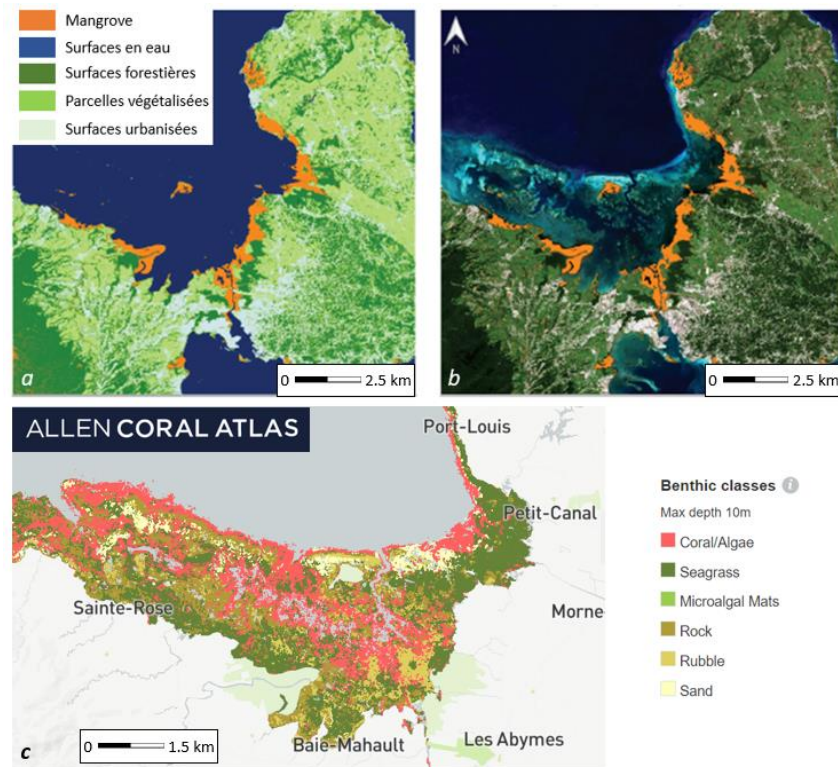


Figure 3: Example of the input data used in the Cul-de-Sac area (Guadeloupe) for the production of the basic variables of the EBCVI. (a) Land cover, including urban class, (b) mangrove distribution and (c) coral and seagrass distribution provided by the Allen Coral Atlas benthic map.

In addition to data on coastal ecosystems, the EBCVI calculation requires input data on the morphology of the area studied, as shown in Figure 4.

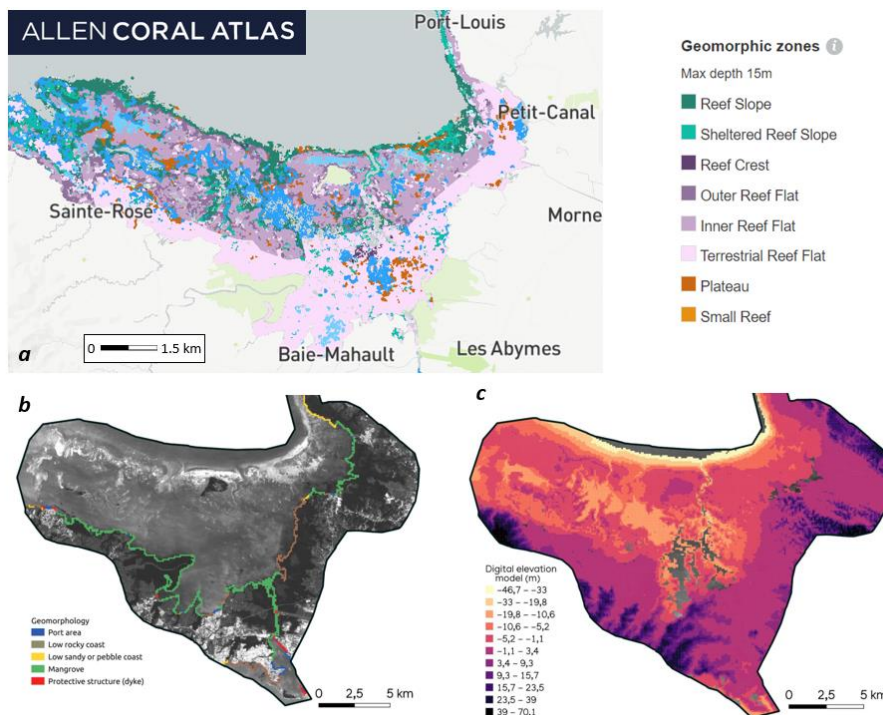


Figure 4: Geomorphological data used as input for the EBCVI calculation: (a) morphological map of the coral substrate and seagrass beds (Allen Coral Atlas), (b) geomorphology of the coastline (BRGM Guadeloupe) and (c) Digital Model of Terrain from the BD Litto 3D.

Methodology for the production of the EBCVI

From a methodological point of view, the distribution maps of the ecosystems and the data on the morphology of the study area mentioned above (figures 3 and 4) make it possible to calculate and spatialize a set of landscape metrics describing the configuration of the coastal landscape (density of ecosystems, fragmentation, height, width, etc.). As shown in Figure 5, these metrics provide a set of descriptive variables that are weighted. These weighted variables are finally associated in a linear combination to calculate an EBCVI per pilot site.

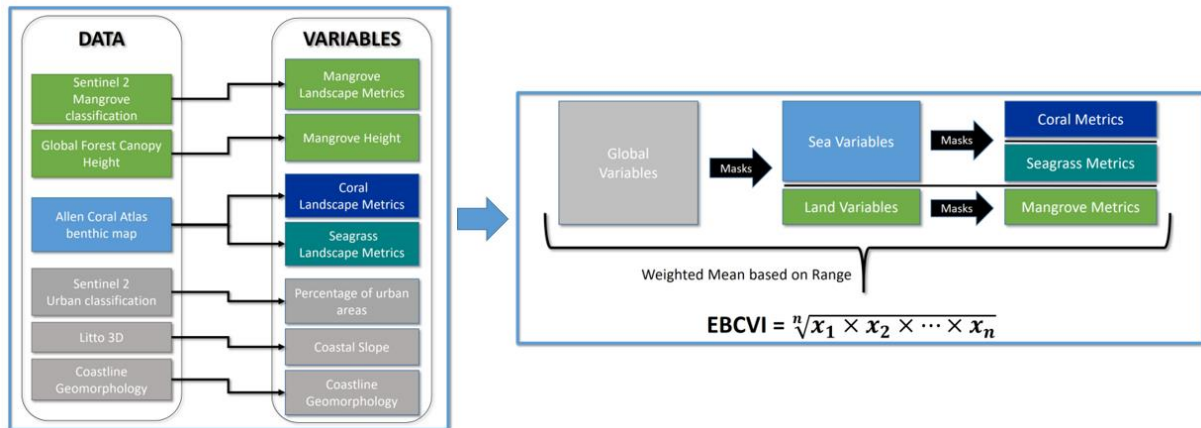


Figure 5: Simplified methodological diagram of EBCVI production at CARIBCOAST pilot sites

For each ecosystem, four landscape metrics were calculated using the Pylandstats python package (<https://pypi.org/project/pylandstats/>, Bosch et al., 2019), on a given surface grid, here each cell of the grid measuring 100x100 m:

- the proportion of the landscape (area covered by the ecosystem on a given reference surface)
- the patch density (number of patches of a given ecosystem on a reference surface)
- Fractal dimension (a measure of shape complexity across a wide range of patch sizes)
- Landscape Form Index (a measure of form complexity)

With the variables from geomorphology, these landscape metrics are the inputs for the EBCVI calculation (figures 7 to 9). All variables are averaged over a grid with 100x100 m cells. Then, to obtain the value of the EBCVI brought back to the pixels of the coastline, the values of the variables in the cells surrounding each coastal pixel are averaged with a weight varying with the distance from the coastline, in a circle of 5 km from diameter centered on each pixel of the sideline (figure 6). Finally, each mean value is assigned a vulnerability rank (from low to high) which is used to calculate the EBCVI.

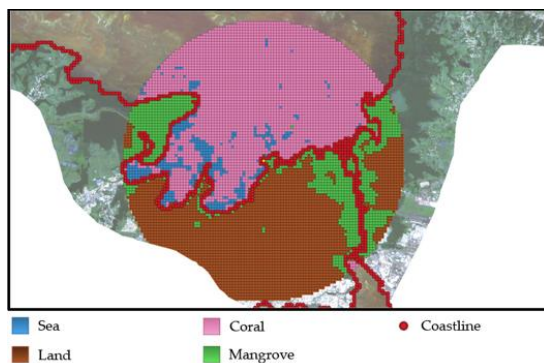


Figure 6: Average and weighting of the descriptive variables of the coastal ecosystems according to the distance from the coastline in a circle of 5 km in diameter centered on the coastal pixels.

The production of this spatialized coastal vulnerability index is semi-automated and is based on the use of an algorithm developed in Python and QGIS, the detailed steps of which are presented in Figure 7 (see appendix for details of the commands used).

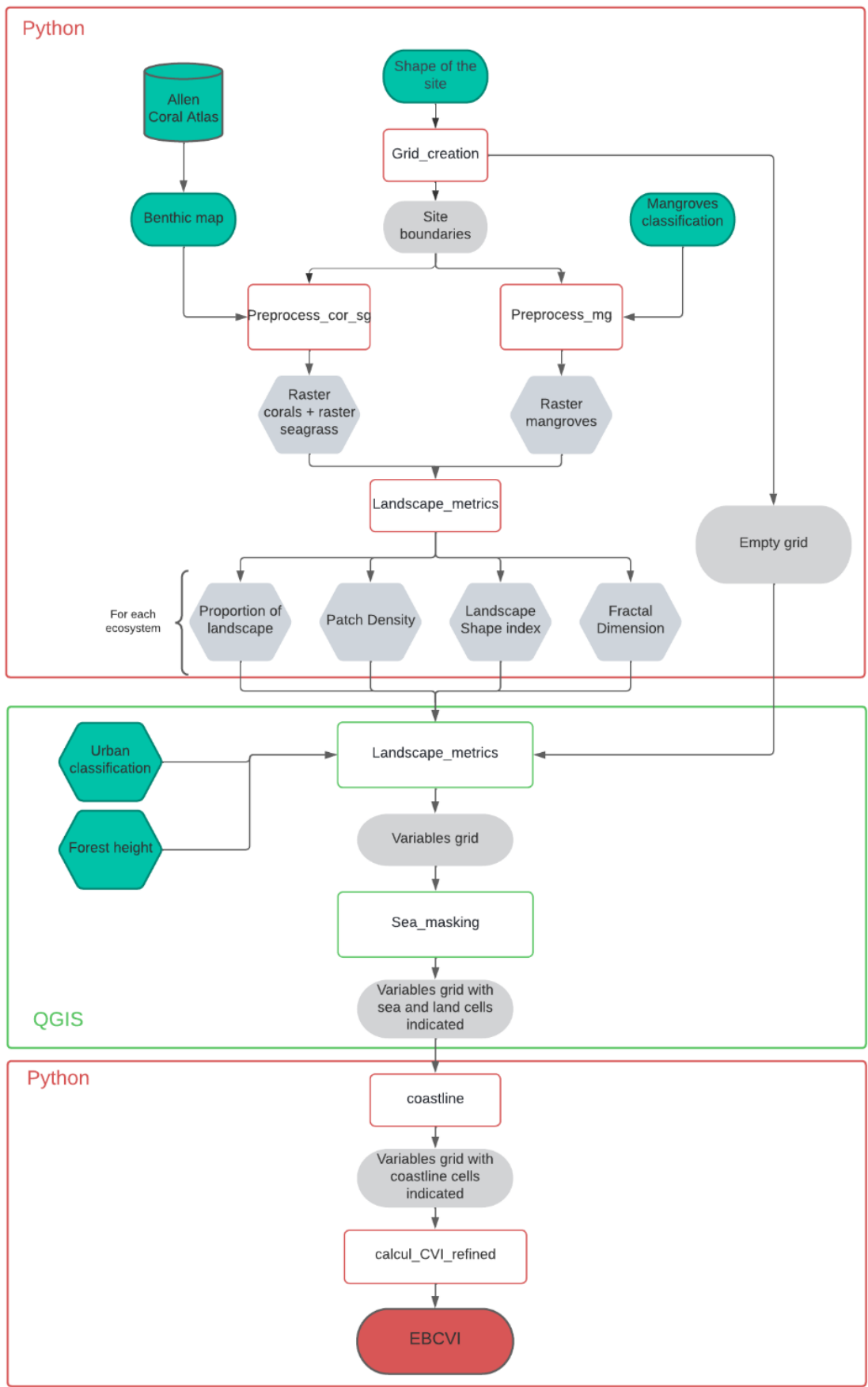


Figure 7: Detail of the steps of the EBCVI production method, based on an algorithm developed in Python and QGIS.

Results

The use of products from satellite imagery as well as descriptive variables of the configuration of the landscape allow the spatialization of the coastal vulnerability index characterizing the role played by coastal ecosystems in protection against erosion.

Figures 8, 9 and 10 present the main variables produced during the initial stages of calculating the EBCVI.

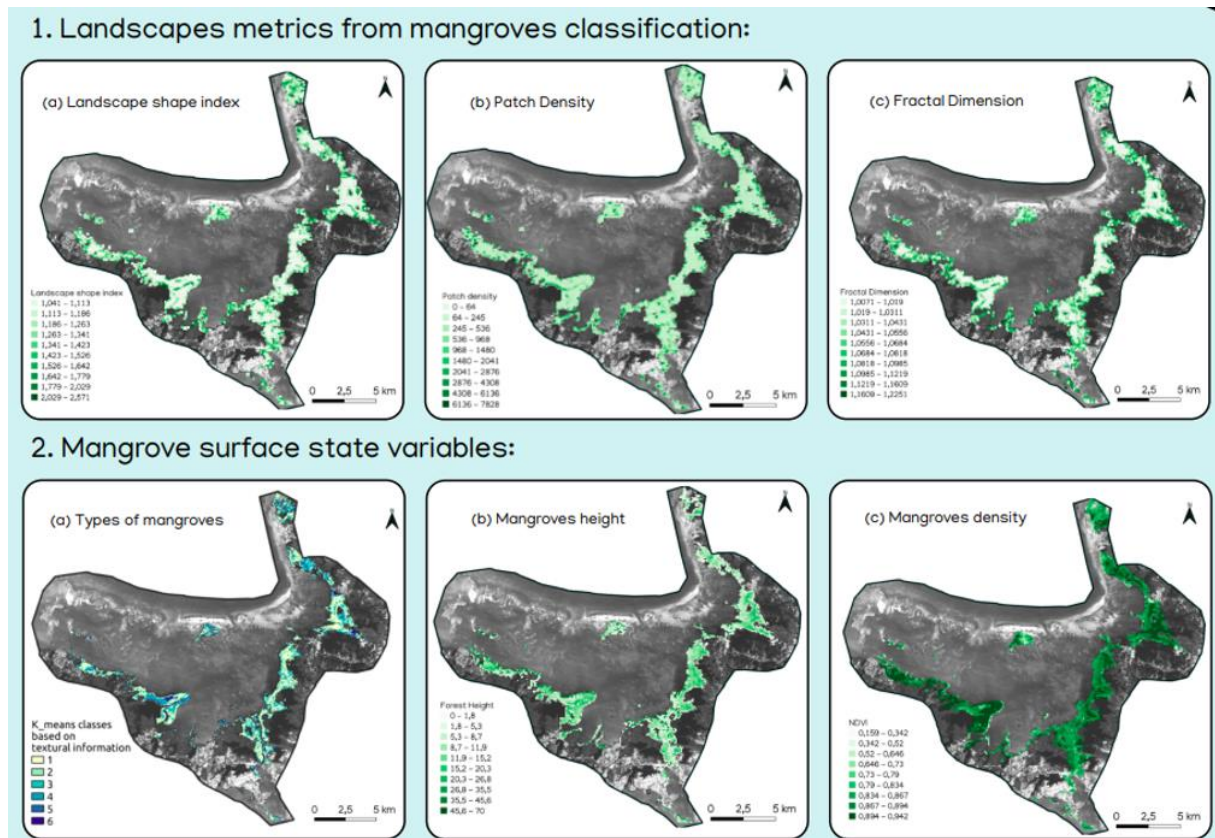


Figure 8: The different variables from the landscape metrics describing the configuration of the mangrove used for the production of EBCVI in Cul-de-Sac (Guadeloupe).

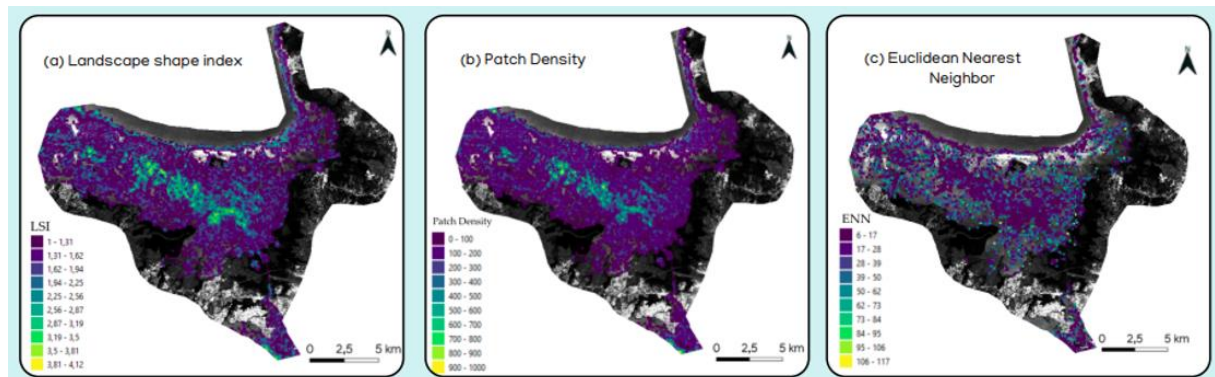


Figure 9: The different variables from landscape metrics describing the configuration of seagrass beds and corals used for the production of EBCVI at Cul-de-Sac (Guadeloupe).

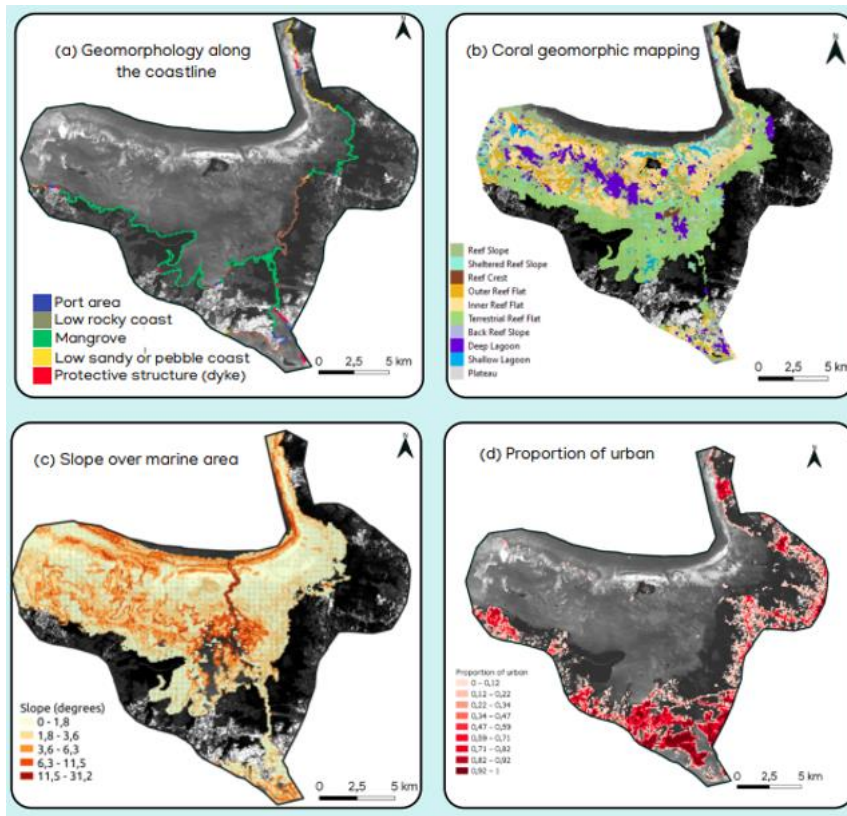


Figure 10: Other variables describing the configuration of the study area for EBCVI production at Cul-de-Sac (Guadeloupe).

This index is calculated according to the linear combination presented in FIG. 5, then normalized. The minimum values correspond to low coastal vulnerability (high protection of the coast by coastal ecosystems) and the maximum values to high coastal vulnerability (low protection of the coast by coastal ecosystems). Figure 11 shows the EBCVI in the Cul-de-Sac area (Guadeloupe).

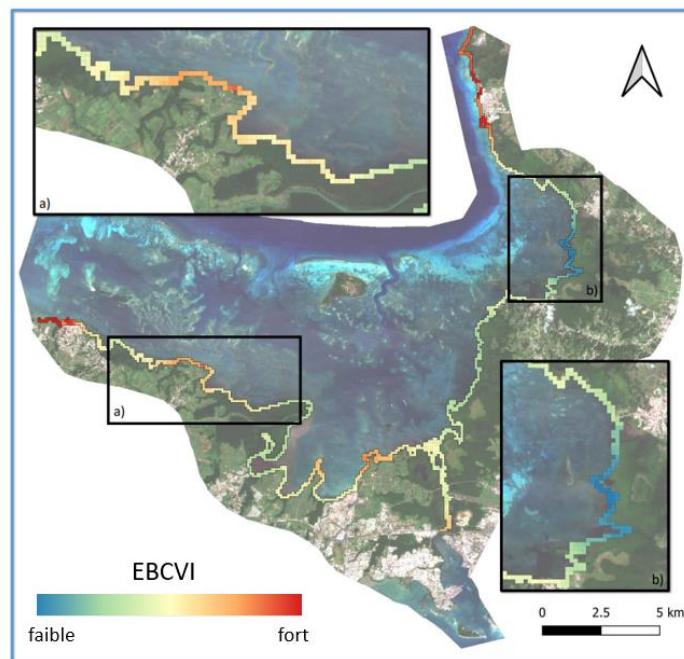


Figure 11: Result of the calculation of the EBCVI, the coastal vulnerability index characterizing the role of coastal ecosystems in protection against erosion, at Cul-de-Sac (Guadeloupe).

The EBCVI maps on all the pilot sites of the project are presented below (figures 12 to 18), and can be viewed on the digital portal of the CARIBCOAST project.

Guadeloupe

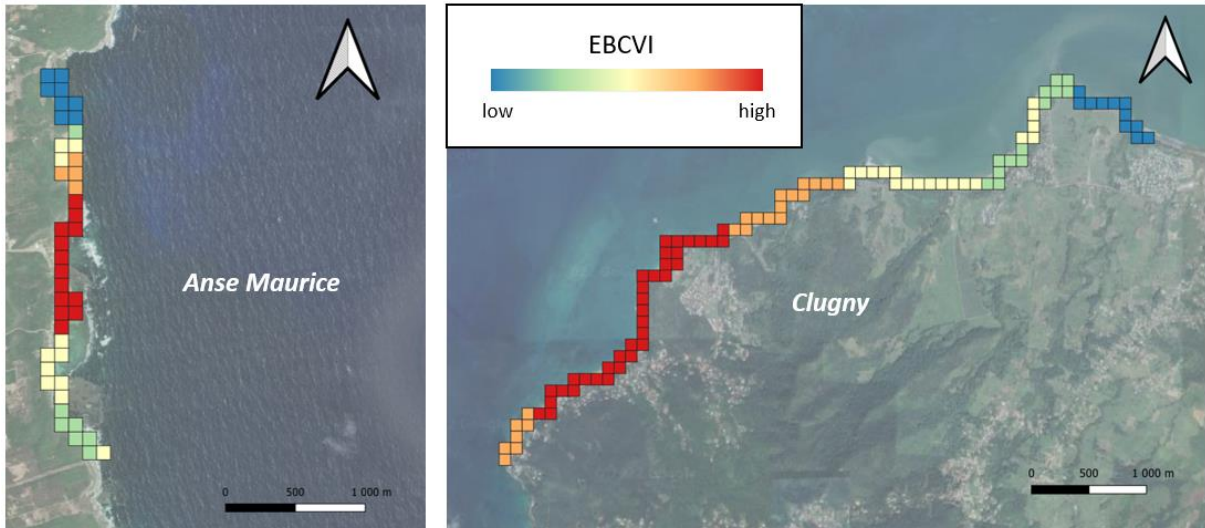


Figure 12: EBCVI on the pilot sites of Anse Maurice and Clugny (Guadeloupe). See Figure 10 for the Cul de Sac site.

Jamaïque

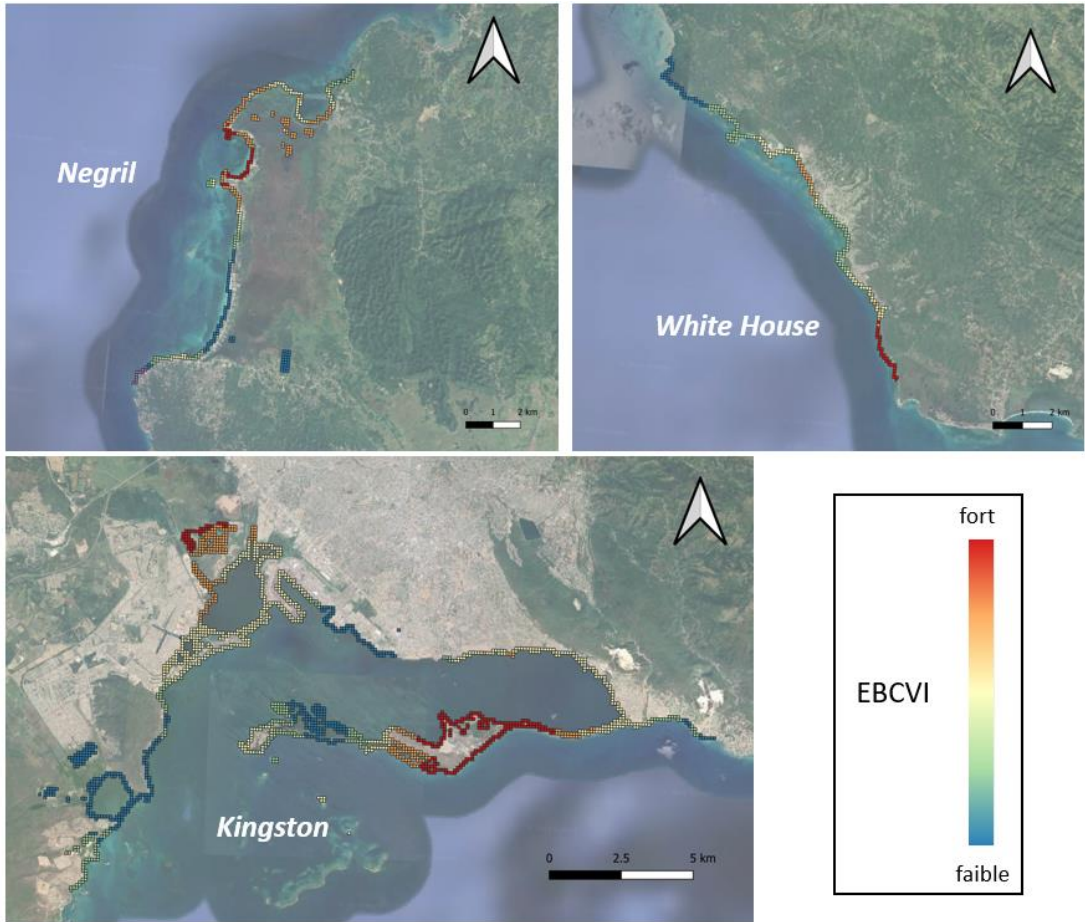


Figure 13: EBCVI at pilot sites in Jamaica, Kingston, White House and Negril.

Martinique

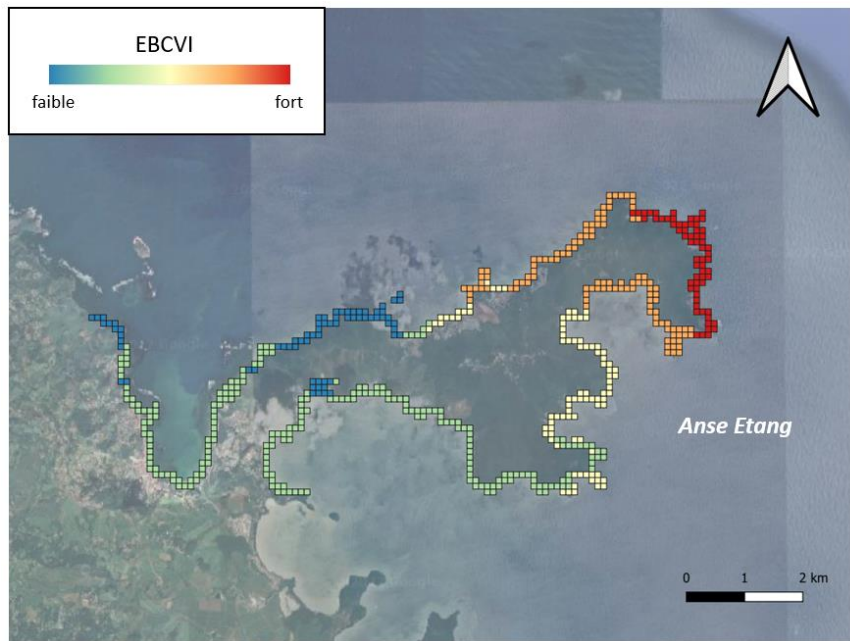


Figure 14: EBCVI on the Anse-Etang pilot site (Martinique).

Porto-Rico

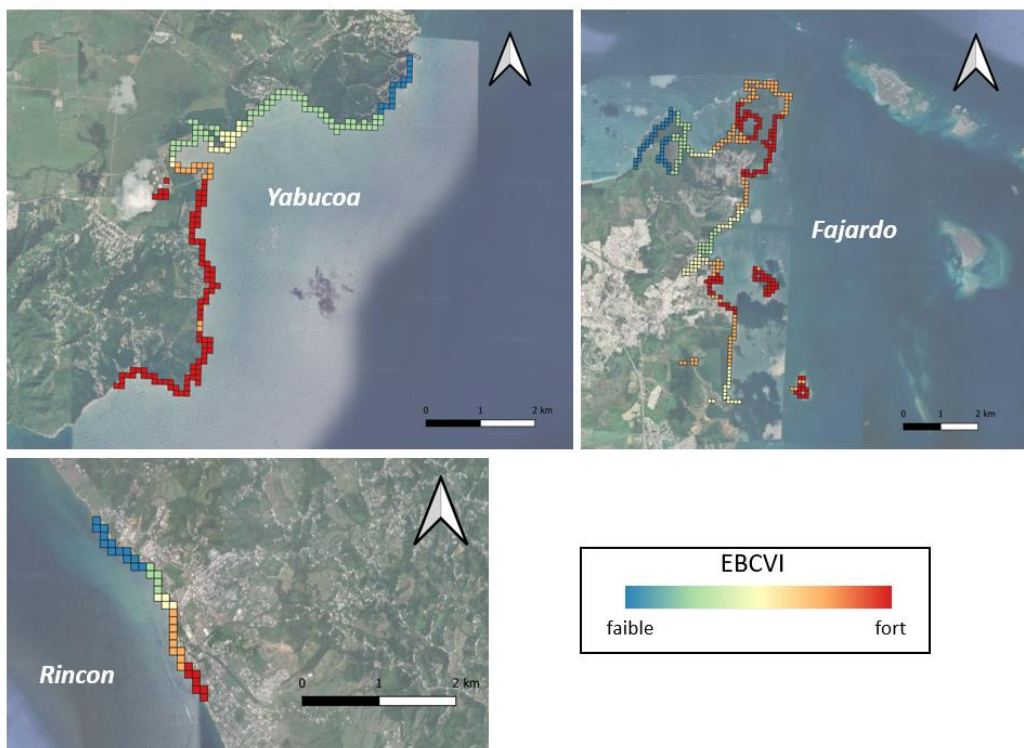


Figure 15: EBCVI in Puerto Rico at the Yabucoa, Rincon and Fajardo sites.

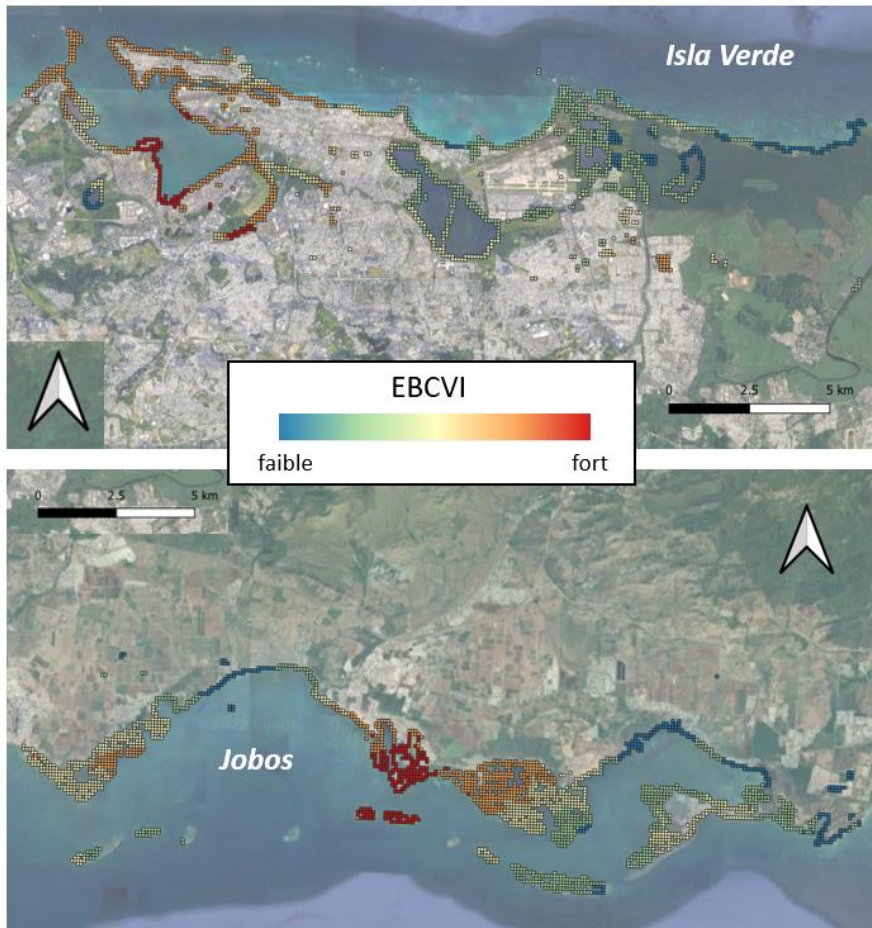


Figure 16: EBCVI in Puerto Rico at the Isla Verde and Jobos sites.

Trinidad et Tobago

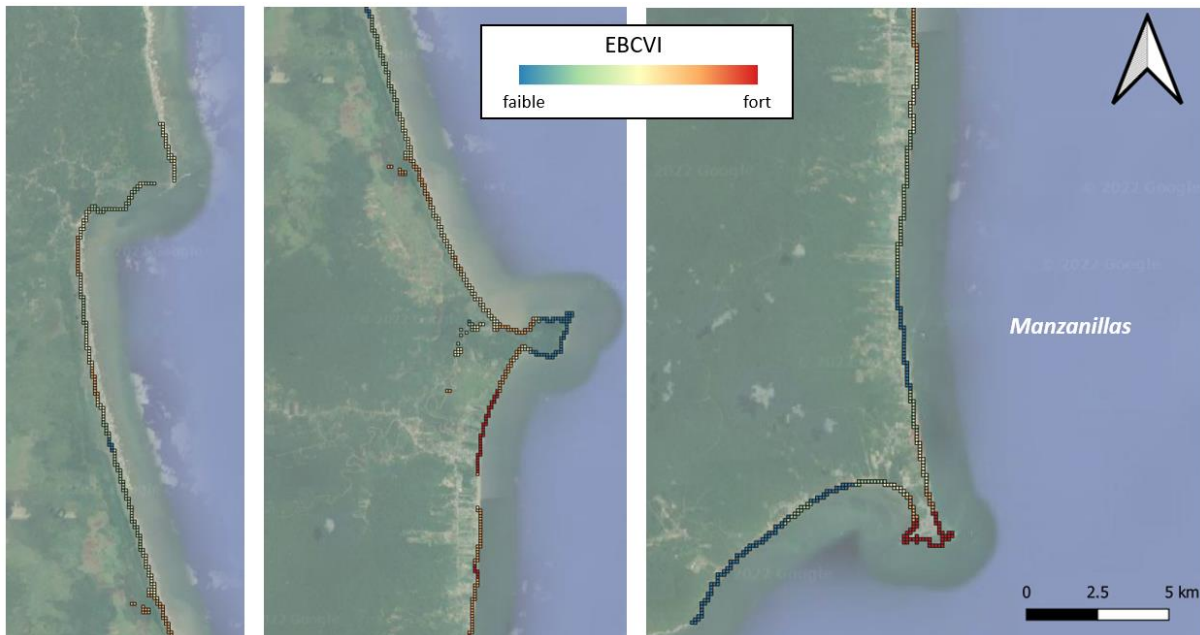


Figure 17: EBCVI in Trinidad and Tobago at the Manzanillas sites.

Saint-Martin

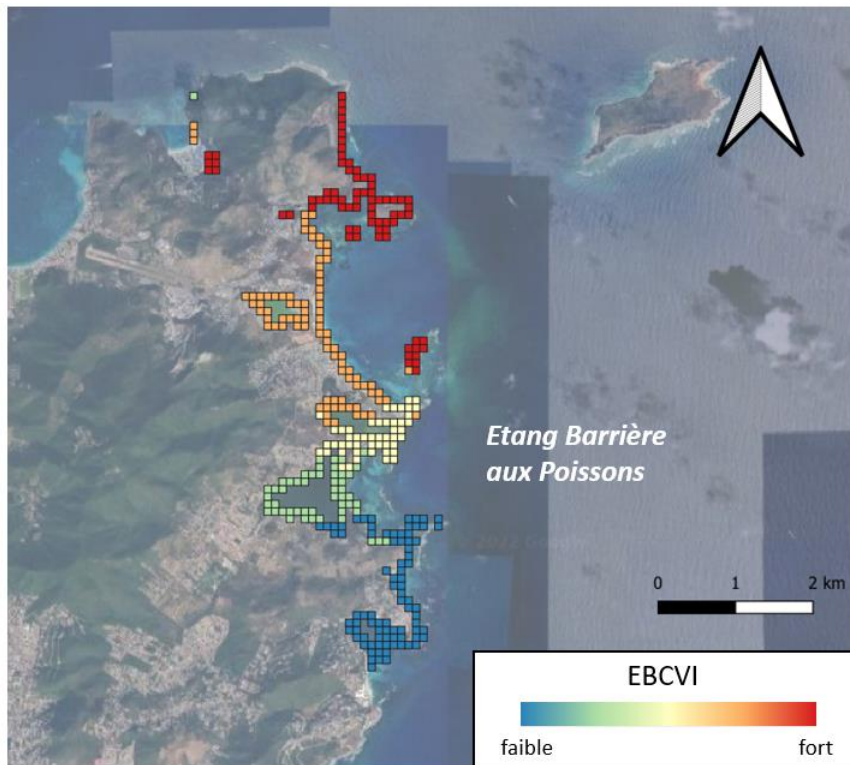


Figure 18: EBCVI in Saint-Martin at the Etang Barrière aux Poissons site.

References

Bosch M (2019) PyLandStats: An open-source Pythonic library to compute landscape metrics. PLoS ONE 14(12): e0225734. <https://doi.org/10.1371/journal.pone.0225734>

Appendix

The commands used in Python to produce the EBCVI following the steps of figure 7 are presented below

Pre-processing of products from Earth observation data (mangroves)

```
import geopandas as gpd
from geocube.api.core import make_geocube
#https://github.com/corteva/geocube
#Emplacement des vecteurs indiquant l'emplacement de mangroves et l'emprise
du site
classif_mg = gpd.read_file("C:/Users/catry/Documents/CaribCoast
Nathan/Global/Classif_mangrove/"

"Guadeloupe/S2_L2A_20190211_Guadeloupe_Extent_Mangrove_Finale.shp")
site = gpd.read_file("C:/Users/catry/Documents/CaribCoast
Nathan/Global/Etendues_sites/Guadeloupe/Emprise_culdesac.shp")
#Emplacement des fichiers vecteur et raster contenant la classif
preprocessed
#On obtient un raster de la classification avec une valeur 1 pour la classe
mangrove et 2 pour le reste
output_vect = "C:/Users/catry/Documents/CaribCoast
Nathan/Global/Classif_mangrove/Guadeloupe/Preprocessed/culdesac_mg_vect.shp
"
output_raster = "C:/Users/catry/Documents/CaribCoast
Nathan/Global/Classif_mangrove/Guadeloupe/Preprocessed/culdesac_mg_raster.ti
f"
```

```

#SCR (projeté) du site
crs = "EPSG:32620"
epsg = 32620

classif_mg = classif_mg.to_crs(epsg=epsg)

#Test de validité (problème avec la validité de la fonction "Repare les
géométries" de QGIS, pas toujours valide pour python)
a = classif_mg.is_valid
b = site.is_valid

# Création de deux Geoseries d'un seul MultiPolygon (c'est ce que renvoie
la fonction) pour les zones mg/pas mg
# ATTENTION, les deux shp doivent être dans le même SCR
inter = site.intersection(classif_mg) #Intersection donc présence de
mangrove
diff = site.difference(classif_mg) #Différence donc absence de mangrove

#Création d'une GeoDataframe à partir de ces Geoseries
data = {'mangrove' : [1, 2], 'geometry' : [inter[0], diff[0]]} #Respecter
l'ordre 1/2 présence/absence
# [0] après les GeoSeries pour accéder au géométries seulement
mg_vect = gpd.GeoDataFrame(data, crs=crs)

#Enregistrement en Shapefile
mg_vect.to_file(output_vect, driver='ESRI Shapefile')

#Création d'un géocube (= rasterisation)
#Bien mettre dans measurements l'attribut tel qu'il est labelisé dans la
dataframe
#Résolution en mètres avec un moins (indique le sens de graduation) pour le
premier terme
cube = make_geocube(mg_vect, measurements=['mangrove'], resolution=(-3,3))
# Pour accéder aux différents attributs (couches) du raster : cube.attribut

#Exportation en tif
cube.mangrove.rio.to_raster(output_raster)

```

Pre-processing of products from Earth observation data (seagrasses and corals)

```

import geopandas as gpd
from geocube.api.core import make_geocube
#https://github.com/corteva/geocube
#Similaire à preprocess_mg mais avec la carte benthique de l'ACA en entrée
classif_benthic = gpd.read_file("C:/Users/catry/Documents/CaribCoast "
"Nathan/Global/ACA/Martinique/AnseEtang/benthic.geojson")
site = gpd.read_file("C:/Users/catry/Documents/CaribCoast
Nathan/Global/Etendues_sites/Martinique/Emprise_anseetang.shp")
epsg = 32620
crs = "EPSG:32620"

output_cor_vect = "C:/Users/catry/Documents/CaribCoast
Nathan/Global/ACA/Martinique/AnseEtang/anseetang_cor_vect.shp"
output_cor_raster = "C:/Users/catry/Documents/CaribCoast
Nathan/Global/ACA/Martinique/AnseEtang/anseetang_cor_raster.tif"

output_sg_vect = "C:/Users/catry/Documents/CaribCoast
Nathan/Global/ACA/Martinique/AnseEtang/anseetang_sg_vect.shp"

```

```

output_sg_raster = "C:/Users/catry/Documents/CaribCoast
Nathan/Global/ACA/Martinique/AnseEtang/anseetang_sg_raster.tif"

coral = classif_benthic.loc[classif_benthic['class'] == 'Coral/Algae']
seagrass = classif_benthic.loc[classif_benthic['class'] == 'Seagrass']

#Changement de SCR car les classifs bethiques sont en 4326, mais nous on a
besoin d'un SCR porjeté (choisir le bon en fonction du site)
coral = coral.to_crs(epsg=epsg)
seagrass = seagrass.to_crs(epsg=epsg)

#Test de validité (problème avec la validité de la fonction "Repare les
géométries" de QGIS, pas toujours valide pour python)
a = coral.is_valid
b = seagrass.is_valid
c = site.is_valid

#Pour l'inter/diff, on utilise unary_union de coral qui nous donne qu'une
seule geometrie union de toutes celles de la dataframe
#pour qu'il fasse l'intersection de cette géométrie avec chaque polygone de
coral/seagrass
#sinon la fonction cherche à faire une intersection "terme à terme" et
plante car coral a plusieurs polygone
#et kingston_site en a un seul
inter_cor = site.intersection(coral.unary_union)
inter_sg = site.intersection(seagrass.unary_union)

diff_cor = site.difference(coral.unary_union)
diff_sg = site.difference(seagrass.unary_union)

#Transformation GeoDataFrame
data_cor = {'coral' : [1,2], 'geometry' : [inter_cor[0], diff_cor[0]]}
cor_vect = gpd.GeoDataFrame(data_cor, crs=crs)
data_sg = {'seagrass' : [1,2], 'geometry' : [inter_sg[0], diff_sg[0]]}
sg_vect = gpd.GeoDataFrame(data_sg, crs=crs)

#Enregistrement shp
cor_vect.to_file(output_cor_vect, driver='ESRI Shapefile')
sg_vect.to_file(output_sg_vect, driver='ESRI Shapefile')

#Création Geocube
cor_cube = make_geocube(cor_vect, measurements=['coral'], resolution=(-
3,3))
sg_cube = make_geocube(sg_vect, measurements=['seagrass'], resolution=(-
3,3))

#Enregistrement raster
cor_cube.coral.rio.to_raster(output_cor_raster)
sg_cube.seagrass.rio.to_raster(output_sg_raster)

```

Calculation of variables (landscape metrics) from preprocessed data and the Pylandstats package

```

import pylandstats as pls
#from https://github.com/martibosch/pylandstats-notebooks

site_name = "anseetang"

```

```
input_filepath_mg = "C:/Users/catry/Documents/CaribCoast
Nathan/Global/Classif_mangrove/Martinique/Preprocessed/anseetang_mg_raster.
tif"
```

```
input_folder_cor_sg = "C:/Users/catry/Documents/CaribCoast
Nathan/Global/ACA/Martinique/AnseEtang/"
input_filepath_cor = input_folder_cor_sg + site_name + "_cor_raster.tif"
input_filepath_sg = input_folder_cor_sg + site_name + "_sg_raster.tif"
```

```
output_folder = "C:/Users/catry/Documents/CaribCoast
Nathan/Global/Metrics/Martinique/AnseEtang/"
```

```
zone_pixel_width, zone_pixel_height = 33, 33 #Taille d'une cellule de
calcul des métriques en nombre de pixels du raster
#Ici, nos rasters ont une résolution de 3m donc on aura des métriques avec
une résolution de 99m
```

```
class_val = 1 #Valeur correspondant à la classe étudiée dans les rasters
metrics = ['proportion_of_landscape', 'patch_density',
'landscape_shape_index', 'fractal_dimension_mn']
```

```
# Métriques disponibles :
```

```
# total_area, proportion_of_landscape, number_of_patches, patch_density,
largest_patch_index, total_edge ,edge_density
# landscape_shape_index, effective_mesh_size, area_mn,
fractal_dimension_md, fractal_dimension_ra, fractal_dimension_sd
# fractal_dimension_cv, euclidean_nearest_neighbor_mn,
euclidean_nearest_neighbor_am, euclidean_nearest_neighbor_md,
# euclidean_nearest_neighbor_ra euclidean_nearest_neighbor_sd,
euclidean_nearest_neighbor_cv
```

```
#Mangroves
```

```
zga = pls.ZonalGridAnalysis(input_filepath_mg,#num_zone_rows=num_zone_rows,
#num_zone_cols=num_zone_cols)
zone_pixel_width=zone_pixel_width,
zone_pixel_height=zone_pixel_height)
```

```
mg_pol = zga.compute_zonal_statistics_arr('proportion_of_landscape',
class_val=class_val,
```

```
dst_filepath= output_folder +
```

```
"Mangroves/" + site_name + "_mg_pol.tif")
```

```
mg_pd = zga.compute_zonal_statistics_arr('patch_density',
class_val=class_val,
```

```
dst_filepath= output_folder +
```

```
"Mangroves/" + site_name + "_mg_pd.tif")
```

```
mg_lsi = zga.compute_zonal_statistics_arr('landscape_shape_index',
class_val=class_val,
```

```
dst_filepath= output_folder +
```

```
"Mangroves/" + site_name + "_mg_lsi.tif")
```

```
mg_fd = zga.compute_zonal_statistics_arr('fractal_dimension_mn',
class_val=class_val,
```

```
dst_filepath= output_folder +
```

```
"Mangroves/" + site_name + "_mg_fd.tif")
```

```
#Coraux
```

```
zga =
```

```
pls.ZonalGridAnalysis(input_filepath_cor,#num_zone_rows=num_zone_rows,
#num_zone_cols=num_zone_cols)
```

```
zone_pixel_width=zone_pixel_width,
zone_pixel_height=zone_pixel_height)
```

```

cor_pol = zga.compute_zonal_statistics_arr('proportion_of_landscape',
class_val=class_val,
dst_filepath= output_folder +
"Coraux/" + site_name + "_cor_pol.tif")
cor_pd = zga.compute_zonal_statistics_arr('patch_density',
class_val=class_val,
dst_filepath= output_folder +
"Coraux/" + site_name + "_cor_pd.tif")
cor_lsi = zga.compute_zonal_statistics_arr('landscape_shape_index',
class_val=class_val,
dst_filepath= output_folder +
"Coraux/" + site_name + "_cor_lsi.tif")
cor_fd = zga.compute_zonal_statistics_arr('fractal_dimension_mn',
class_val=class_val,
dst_filepath= output_folder +
"Coraux/" + site_name + "_cor_fd.tif")

#Herbiers
zga = pls.ZonalGridAnalysis(input_filepath_sg,#num_zone_rows=num_zone_rows,
#num_zone_cols=num_zone_cols)
zone_pixel_width=zone_pixel_width,
zone_pixel_height=zone_pixel_height)

sg_pol = zga.compute_zonal_statistics_arr('proportion_of_landscape',
class_val=class_val,
dst_filepath= output_folder +
"Herbiers/" + site_name + "_sg_pol.tif")
sg_pd = zga.compute_zonal_statistics_arr('patch_density',
class_val=class_val,
dst_filepath= output_folder +
"Herbiers/" + site_name + "_sg_pd.tif")
sg_lsi = zga.compute_zonal_statistics_arr('landscape_shape_index',
class_val=class_val,
dst_filepath= output_folder +
"Herbiers/" + site_name + "_sg_lsi.tif")
sg_fd = zga.compute_zonal_statistics_arr('fractal_dimension_mn',
class_val=class_val,
dst_filepath= output_folder +
"Herbiers/" + site_name + "_sg_fd.tif")

```

Generation of the 100x100 m grid and average of the variables on the grid

```

import geopandas as gpd
from shapely.geometry import Polygon
import numpy as np
import rasterio
import rasterio.mask
from rasterio.warp import calculate_default_transform, reproject,
Resampling
from pyrastra.raster import Raster
from rasterstats import zonal_stats
#https://pythonhosted.org/rasterstats/manual.html

if __name__ == "__main__":

    #Rajouter la ligne d'avant pour pouvoir faire fonctionner PyRasta
    zonalstats

#Paramètres :

```

```

#Emplacement du vecteur qui donne les limites du site
site = gpd.read_file("C:/Users/catry/Documents/CaribCoast
Nathan/Global/Etendues_sites/Martinique/M_ANSE_ETANG.shp")
#SCR du site (Attention de bien choisir un scr projeté ça sera utile
plus tard)
crs = "EPSG:32620"
#Emplacement de sortie de l'emprise
emprise_path = "C:/Users/catry/Documents/CaribCoast
Nathan/Global/Etendues_sites/Martinique/Emprise_anseetang.shp"
#Emplacements de la grille et de la grille découpée selon les
frontières du site
output_grid = "C:/Users/catry/Documents/CaribCoast
Nathan/Global/Grids/Martinique/anseetang_grid.shp"
output_grid_cut = "C:/Users/catry/Documents/CaribCoast
Nathan/Global/Grids/Martinique/anseetang_grid_cut.shp"
#Taille de la grille
length = 100
wide = 100

xmin, ymin, xmax, ymax = site.total_bounds

emprise_geo = [Polygon([(xmin, ymin), (xmax, ymin), (xmax, ymax),
((xmin, ymax))])]
emprise = gpd.GeoDataFrame({'geometry': emprise_geo}, crs=crs)
emprise.to_file(emprise_path, driver='ESRI Shapefile')

#On ajoute wide/length au max pour être sûr de tout couvrir (on dépasse
potentiellement un peu)
cols = list(np.arange(xmin, xmax + wide, wide))
rows = list(np.arange(ymin, ymax + length, length))

#Création d'une liste de carrés Shapely : les cellules de la grille
grid_cells = []
for x in cols[:-1]:
    for y in rows[:-1]:
        grid_cells.append(Polygon([(x, y), (x + wide, y), (x + wide, y
+ length), (x, y + length)]))

#Transformation en database
grid = gpd.GeoDataFrame({'geometry': grid_cells}, crs=crs)
grid.to_file(output_grid, driver='ESRI Shapefile')

#Déoupage de la grille pour qu'elle matche l'emprise du site
geome = site.geometry[0]
grid_cut_serie = grid.intersection(site.geometry[0])
grid_cut = gpd.GeoDataFrame(geometry=grid_cut_serie, crs=crs)
grid_cut = grid_cut.loc[~ grid_cut.geometry.is_empty]#Les opérateurs
logiques "classiques" (and, or, not) ne fonctionnent
#pas avec les Series de booléens, on doit donc utiliser les "bitwise
operators" (&, |, ~), ici cela revient à not geometry.is_empty
grid_cut.to_file(output_grid_cut, driver='ESRI Shapefile')

#Ici essais de zonalstats qui n'ont pas fonctionné
"""
#Essai avec pyраста
stats = ["mean", "max"]

rstats = Raster(raster_path).zonal_stats(grid_cut, stats=stats)

```

```

df_rstats = pd.DataFrame(rstats)

grid_cut4 = grid_cut
grid_cut4['mean'] = df_rstats['mean']
grid_cut4['max'] = df_rstats['max']

#grid_cut4.to_file("C:/Users/catry/Documents/CaribCoast
Nathan/Global/EBCVI/Test/Kingston_height_stats4.shp",
# driver='ESRI Shapefile')
#grid_cut_seagrass.to_file("C:/Users/catry/Documents/CaribCoast
Nathan/Global/EBCVI/Kingston_seagrass_stats.shp",
# driver='ESRI Shapefile')
"""
"""
#Essai avec rasterstats
with rasterio.open("C:/Users/catry/Documents/CaribCoast
Nathan/Global/Forest_height/Kingston_mg_height.tif") as src:
    affine = src.transform
    array = src.read(1)
    df_zonal_stats = pd.DataFrame(zonal_stats(grid_cut_4326, array,
affine=affine))
    grid_cut2 = pd.concat([grid_cut_4326, df_zonal_stats], axis=1)

    grid_cut2.to_file("C:/Users/catry/Documents/CaribCoast
Nathan/Global/EBCVI/Kingston_height_stats.shp", driver='ESRI Shapefile')

```

Weighting and assignment of variable values to the coastline pixel

```

import geopandas as gpd

sea_mask = gpd.read_file("C:/Users/catry/Documents/CaribCoast
Nathan/Global/Sea_masks/Trinidad/Sea_mask_manzanilla.shp")

grid = gpd.read_file("C:/Users/catry/Documents/CaribCoast
Nathan/Global/EBCVI/Trinidad/ManzanillaCocos/Var_manzanilla_sea.shp")

#On utilise les limites de la zone maritime (déterminée sur Qgis grâce à la
Forest_height) pour trouver la coastline
lines = sea_mask.boundary
coastline = lines.geometry[0]

mask = grid.intersects(coastline)

grid['coastline'] = 0
grid.loc[mask, 'coastline'] = 1

grid.to_file("C:/Users/catry/Documents/CaribCoast
Nathan/Global/EBCVI/Trinidad/ManzanillaCocos/Var_manzanilla_coastline.shp",
driver = 'ESRI Shapefile')

```

Calculation of the EBCVI normalization by weighted linear combination of the preceding variables

```

import geopandas as gpd
import pandas as pd
import numpy as np
import jenkspy #https://github.com/mthh/jenkspy
#https://pbpython.com/natural-breaks.html

```

```

variables_grid_path = "C:/Users/catry/Documents/CaribCoast
Nathan/Global/EBCVI/Martinique/AnseEtang/Var_anseetang_coastline.shp"
data = gpd.read_file(variables_grid_path)

output_file_unranked = "C:/Users/catry/Documents/CaribCoast
Nathan/Global/EBCVI/Martinique/AnseEtang/Index/Unranked_coast_anseetang.shp
"
output_file_ranked = "C:/Users/catry/Documents/CaribCoast
Nathan/Global/EBCVI/Martinique/AnseEtang/Index/CVI_anseetang.shp"

no_mangrove = False #Mettre True si il n'y a pas de mangroves sur le site
no_benthic = False #Idem pour coraux/herbiers

coast_data = data.loc[data.coastline == 1]
coast_centroides = coast_data.centroid
shp = coast_centroides.shape

land_variables = ['mpol_mean', 'u_mean']
sea_variables = ['cpol_mean', 'spol_mean']
mg_variables = ['mpd_mean', 'mfd_mean', 'mlsi_mean', 'h_mean']
cor_variables = ['cpd_mean', 'cfd_mean', 'clsi_mean']
sg_variables = ['spd_mean', 'sfd_mean', 'slsi_mean']
missing_variables = []

up_variables = ['cpd_mean', 'clsi_mean', 'cfd_mean',
               'mpd_mean', 'mlsi_mean', 'mfd_mean',
               'spd_mean', 'slsi_mean', 'sfd_mean',
               'u_mean'] # Variables proportionnelles à la vulnérabilité

down_variables = ['cpol_mean', 'mpol_mean', 'spol_mean', 'h_mean'] #
Variables INVERSEMENT proportionnelles à la vulnérabilité

if no_mangrove :
    land_variables = ['u_mean']
    up_variables = ['cpd_mean', 'clsi_mean', 'cfd_mean',
                  'spd_mean', 'slsi_mean', 'sfd_mean',
                  'u_mean']
    down_variables = ['cpol_mean', 'spol_mean']
    missing_variables = ['mpol_mean', 'mpd_mean', 'mlsi_mean', 'mfd_mean',
                        'h_mean']

if no_benthic :
    up_variables = ['mpd_mean', 'mlsi_mean', 'mfd_mean',
                  'u_mean']
    down_variables = ['mpol_mean', 'h_mean']
    missing_variables = ['cpol_mean', 'cpd_mean', 'clsi_mean', 'cfd_mean',
                        'spol_mean', 'spd_mean', 'slsi_mean', 'sfd_mean']

n_var = len(up_variables) + len(down_variables) + len(missing_variables) #
Variables_plus + Variables_minus

xmin, ymin, xmax, ymax = data.total_bounds
max_range_sea = 5000 #Rayon du disque dans lequel on considère que les
pixels ont une influence sur le pixel de côte
max_range_land = 2000

coast_data = coast_data.reset_index(drop=True)
coast_centroides = coast_centroides.reset_index(drop=True)

```

```

for i in range(shp[0]) :#Faire une boucle for pour chaque centroide

    center = coast_centroides.geometry[i]

    #Setup coeffs Mer
    mask_sea_circle = data.centroid.distance(center) < max_range_sea #
masque disque autour du pixel de côte

    circle_sea = data.loc[mask_sea_circle]
    circle_sea['coeff'] = (max_range_sea -
circle_sea.centroid.distance(center)) / max_range_sea #colonne contenant
les coeffs de pondération en fonction de la distance

    sea_mask = data['sea_majori'] == 1 #masque mer seulement
    # On voudrait appliquer les masques circle + terre/mer d'un coup mais
l'un est une Série de booléen alors que l'autre est une liste
    sea_area = circle_sea.loc[sea_mask]
    sea_coeff_tot = sea_area['coeff'].sum() # Somme des coeffs sur la zone
mer (pour calculer la moyenne pondérée ensuite)

    #Setup coeffs Terre
    mask_land_circle = data.centroid.distance(center) < max_range_land #
masque disque autour du pixel de côte

    area_land = data.loc[mask_land_circle]
    area_land['coeff'] = (max_range_land - area_land.centroid.distance(
center)) / max_range_land # colonne contenant les coeffs de
pondération en fonction de la distance

    land_mask = data['sea_majori'] == 0 # masque terre
    land_area = area_land.loc[land_mask]
    land_coeff_tot = land_area['coeff'].sum()

    #Variables Mer
    if not no_benthic :
        for var in sea_variables :
            coast_data.loc[i, var] = (sea_area[var] *
sea_area['coeff']).sum() / sea_coeff_tot

            cor_mask = sea_area['cpol_mean'] > 0
            cor_area = sea_area.loc[cor_mask] #Application du masque coraux
seulement pour variables secondaires
            cor_coeff_tot = cor_area['coeff'].sum()

            for var in cor_variables :
                coast_data.loc[i, var] = (cor_area[var] *
cor_area['coeff']).sum() / cor_coeff_tot

            sg_mask = sea_area['spol_mean'] > 0
            sg_area = sea_area.loc[sg_mask] # Application du masque coraux
seulement pour variables secondaires
            sg_coeff_tot = sg_area['coeff'].sum()

            for var in sg_variables :
                coast_data.loc[i, var] = (sg_area[var] *
sg_area['coeff']).sum() / sg_coeff_tot

    #Variables Terre
    for var in land_variables :

```

```

        coast_data.loc[i, var] = (land_area[var] *
land_area['coeff']).sum() / land_coeff_tot

        if not no_mangrove :
            mg_mask = land_area['mpol_mean'] > 0 # Ce test considère bien les
NaN comme < 0 donc ça marche
            mg_area = land_area.loc[mg_mask]
            mg_coeff_tot = mg_area['coeff'].sum()

            for var in mg_variables :
                coast_data.loc[i, var] = (mg_area[var] *
mg_area['coeff']).sum() / mg_coeff_tot

        print(shp[0] - i)

coast_data.to_file(output_file_unranked, driver='ESRI Shapefile')

coast_data = gpd.read_file(output_file_unranked)
#Ranking Jenks
coast_data_ranked = coast_data.copy()
coast_data_ranked['cvi'] = 1 #initialisation de la colonne à 1

for var in up_variables :
    breaks = jenkspy.jenks_breaks(coast_data[var], nb_class=6)
    coast_data_ranked[var] = pd.cut(coast_data[var], bins=breaks,
labels=[1, 2, 3, 4, 5, 6], include_lowest=True)
    # Attention : Le type des labels n'est pas int mais Categorical
    coast_data_ranked[var] = coast_data_ranked[var].astype(np.double)
    #conversion en double numpy pour pouvoir faire des opérations avec, On
choisit double car les NaN produits par pd.cut sont des NaN float
    coast_data_ranked['cvi'] = coast_data_ranked['cvi'] *
coast_data_ranked[var]

for var in down_variables :
    breaks = jenkspy.jenks_breaks(coast_data[var], nb_class=6)
    # Attention au duplications de bords
    if breaks[-2] == breaks[-1] :
        breaks[-1] = breaks[-1] * 1.01
    for k in range(len(breaks)-2) :
        if breaks[k] == breaks[k+1] :
            breaks[k+1] += (breaks[k+2] - breaks[k+1]) * 1.01
    coast_data_ranked[var] = pd.cut(coast_data[var], bins=breaks,
labels=[6, 5, 4, 3, 2, 1], include_lowest=True)
    coast_data_ranked[var] = coast_data_ranked[var].astype(np.double)
    coast_data_ranked['cvi'] = coast_data_ranked['cvi'] *
coast_data_ranked[var]

for var in missing_variables:
    coast_data_ranked[var] = 6
    coast_data_ranked['cvi'] = coast_data_ranked['cvi'] *
coast_data_ranked[var]

# Formule du CVI : produit des variables normalisé entre 1 et 6, puis -1
pour échelle entre 0 et 5 plus claire
coast_data_ranked['cvi'] = coast_data_ranked['cvi'] ** (1 / n_var) - 1

coast_data_ranked.to_file(output_file_ranked, driver='ESRI Shapefile')

```