



WP3 : SURVEILLANCE ET ATTENUATION DE L'ÉROSION CÔTIÈRE

Note Technique

Spatialisation d'un indice de vulnérabilité côtière caractérisant le rôle des écosystèmes littoraux (mangrove, herbiers, récifs coralliens) dans la protection contre l'érosion côtière dans les Caraïbes par télédétection

Introduction

Bien que les côtes insulaires soient naturellement soumises à l'érosion, le contexte actuel de changement climatique et les activités anthropiques font partie des sources d'accentuation de ce phénomène. En particulier, l'augmentation de la fréquence et/ou de l'intensité des perturbations climatiques (cyclones, houles, élévation du niveau marin...) est responsable d'une amplification de cette érosion qui impacte fortement les littoraux des Caraïbes et menace à la fois les écosystèmes naturels, et des activités comme le tourisme. Outre des infrastructures anthropiques, les écosystèmes littoraux ont naturellement un rôle à jouer dans la protection et l'atténuation du phénomène d'érosion côtière. C'est en particulier le cas de la mangrove, des herbiers et des récifs coralliens. Une partie des travaux menés par l'IRD/UMR Espace-Dev en collaboration avec la Telescop dans le WP3 du projet CARIBCOAST a consisté à mettre à profit la disponibilité d'images satellite et de produits cartographiques issus de l'observation de la Terre pour spatialiser un indice de vulnérabilité côtière (EBCVI pour Ecosystem-Based Coastal Vulnerability Index) caractérisant le rôle de ces écosystèmes littoraux dans la protection contre l'érosion côtière dans les Caraïbes.

Un indice de vulnérabilité vise à simplifier un certain nombre de paramètres complexes et interactifs, représentés par divers types de données, sous une forme plus facilement compréhensible et donc plus utile en tant qu'outil de gestion. Sur la base de la littérature, nous avons choisi de rassembler des variables qui définissent

- les caractéristiques géomorphologiques et des écosystèmes du littoral
- le degré auquel la côte est exposée au forçage côtier et à l'énergie
- des informations sur les éléments socio-économiques pour lesquels l'érosion côtière représente un risque

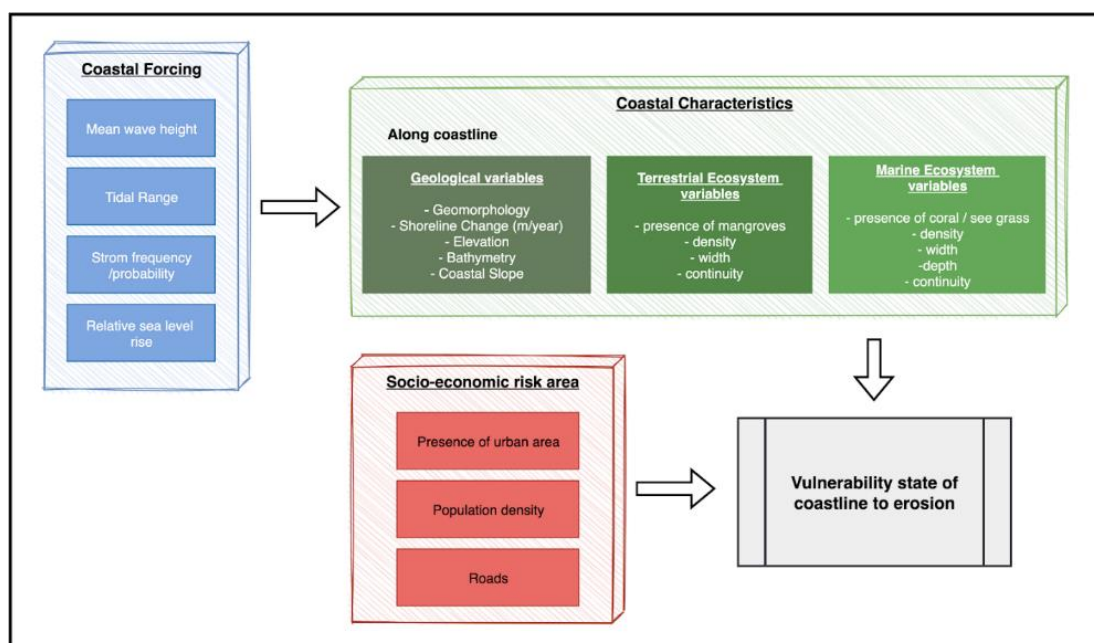


Figure 1 : Approche générale de production de l'EBCVI utilisée dans ces travaux

La figure 1 ci-dessus résume l'approche générale qui sera utilisée pour produire l'EBCVI sur les sites pilotes du projet (figure 2).

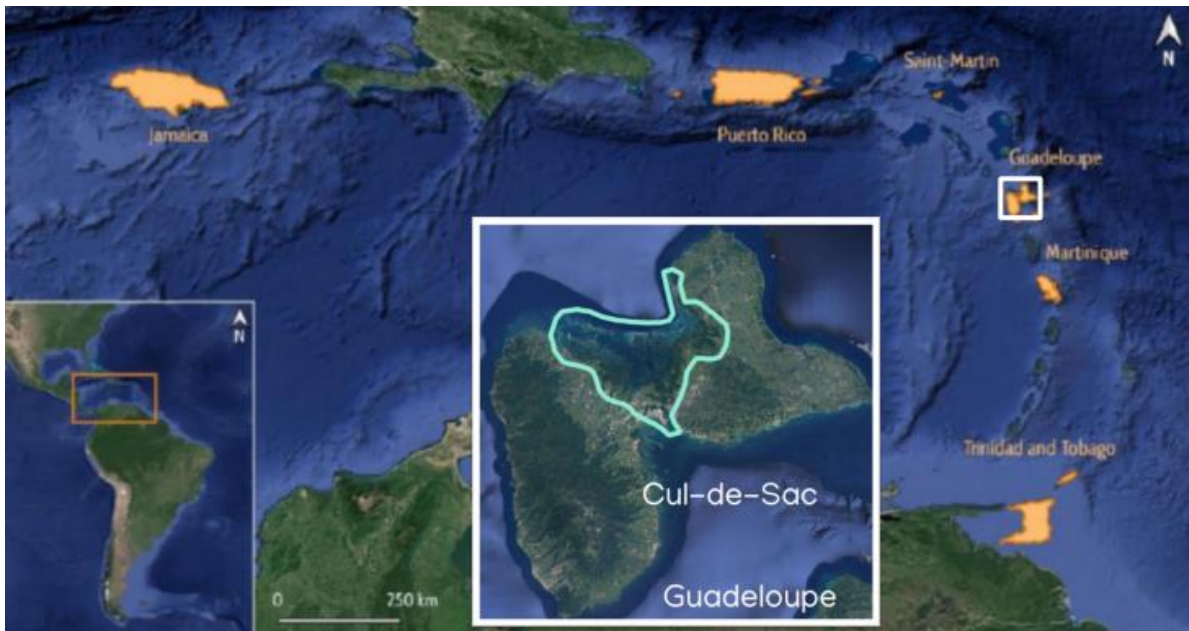


Figure 2 : Les sites d'étude du projet CARIBCOAST, dont Cul-de-Sac en Guadeloupe qui servira à illustrer ce rapport.

Données d'entrée du calcul de l'EBCVI

Pour chacun des sites pilotes du projet, la création de cet indice spatialisé repose sur l'utilisation comme données d'entrée d'imagerie satellite, de Modèles numériques de terrain et de bases de données externes. En particulier, la méthode de calcul de l'EBCVI repose sur (figure 3) :

- de la cartographie des zones urbaines et des mangroves à 10m de résolution spatiale issue de Sentinel 2, produite dans le cadre de ce projet par l'IRD/UMR Espace-Dev en collaboration avec la Telescop,
- de la cartographie des herbiers marins et des récifs coralliens issue de l'Allen Coral Atlas (<https://allencoralatlas.org/>), et obtenue à partir d'images Planetscope (<https://earth.esa.int/eogateway/missions/planetscope>),
- de la carte de hauteur de mangrove fournie par le produit Global Forest Canopy Height de 2019 (<https://glad.umd.edu/dataset/gedi>) issu du capteur GEDI.

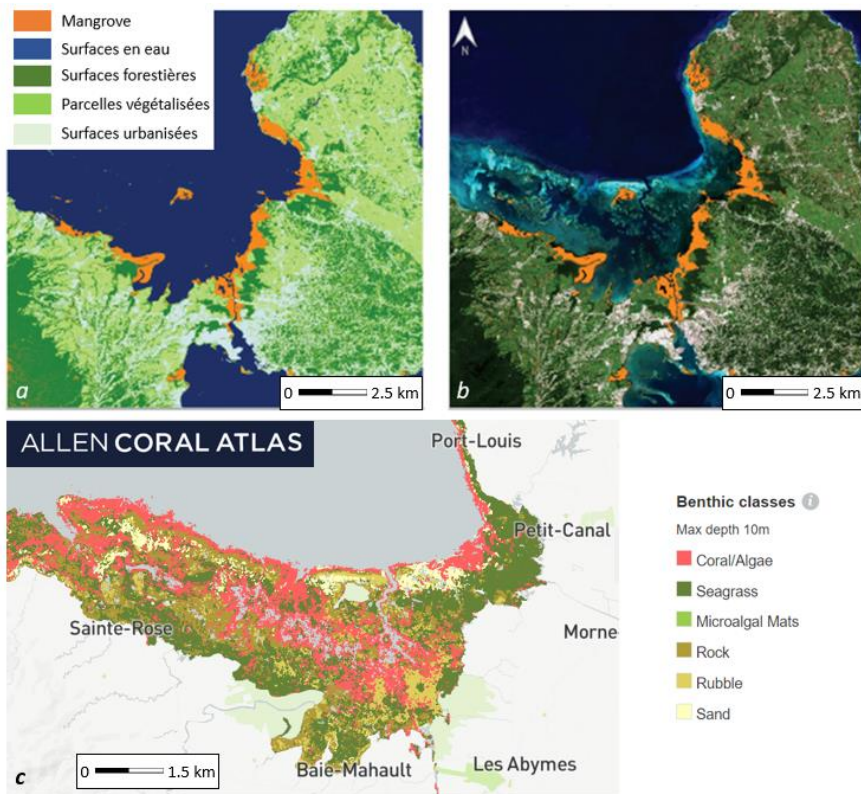


Figure 3 : Exemple des données d'entrée utilisées sur la zone de Cul-de-Sac (Guadeloupe) pour la production des variables de base de l'EBCVI. (a) Occupation du sol, dont classe urbaine, (b) distribution de la mangrove et (c) distribution des coraux et herbiers fournie par la carte benthique de l'Allen Coral Atlas

En plus des données sur les écosystèmes littoraux, le calcul de l'EBCVI nécessite en entrée des données sur la morphologie de la zone étudiée, comme indiquée sur la figure 4.

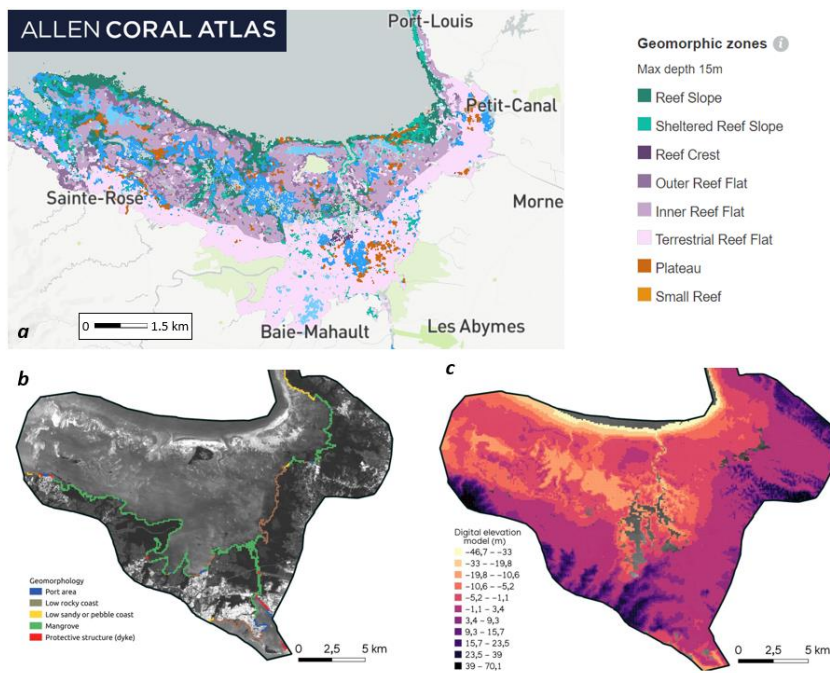


Figure 4 : Données géomorphologiques utilisées en entrée du calcul de l'EBCVI : (a) carte morphologique du substrat corallien et des herbiers (Allen Coral Atlas), (b) géomorphologie du trait de côte (BRGM Guadeloupe) et (c) Modèle Numérique de Terrain issu de la BD Litto 3D.

Méthodologie de calcul de l'EBCVI

D'un point de vue méthodologique, les cartes de distributions des écosystèmes et les données sur la morphologie de la zone d'étude mentionnées ci-dessus (figures 3 et 4) permettent de calculer et spatialiser un ensemble de métriques paysagères décrivant la configuration du paysage littoral (densité des écosystèmes, fragmentation, hauteur, largeur...). Comme le montre la figure 5, ces métriques fournissent un ensemble de variables descriptives qui sont pondérées. Ces variables pondérées sont enfin associées dans une combinaison linéaire pour calculer un EBCVI par site pilote.

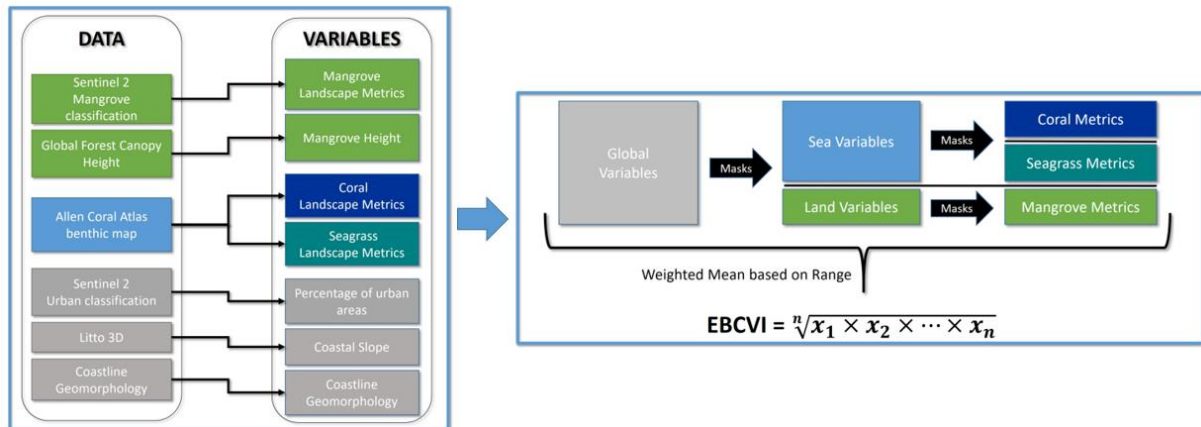


Figure 5 : Schéma méthodologique simplifié de la production de l'EBCVI sur les sites pilote de CARIBCOAST

Pour chaque écosystème, quatre métriques paysagères ont été calculées à l'aide du package python Pylandstats (<https://pypi.org/project/pylandstats/>, Bosch et al., 2019), sur une grille de surface donnée, ici chaque cellule de la grille mesurant 100x100 m :

- la proportion du paysage (surface de couverture de l'écosystème sur une surface de référence donnée)
- la densité des patches (nombre de patch d'un écosystème donnée sur une surface de référence)
- la dimension fractale (une mesure de la complexité de la forme selon une large gamme de tailles de patch)
- l'indice de forme de paysage (une mesure de la complexité de la forme)

Avec les variables issues de la géomorphologie, ces métriques paysagères sont les entrées du calcul de l'EBCVI (figures 7 à 9). Toutes les variables sont moyennées sur une grille avec des cellules de 100x100 m. Ensuite, pour obtenir la valeur de l'EBCVI ramenée aux pixels du trait de côte, les valeurs des variables dans les cellules entourant chaque pixel côtier sont moyennées avec un poids variant avec la distance au trait de côte, dans un cercle de 5 km de diamètre centré sur chaque pixel du trait de côté (figure 6). Enfin, chaque valeur moyenne se voit affecter un rang de vulnérabilité (de faible à fort) qui est utilisé pour calculer l'EBCVI.

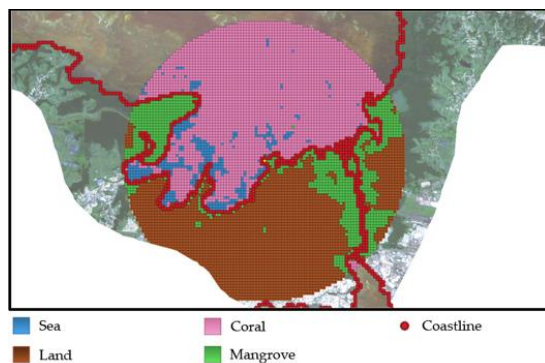


Figure 6 : Moyenne et pondération des variables descriptives des écosystèmes littoraux en fonction de la distance au trait de côte dans un cercle de 5 km de diamètre centré sur les pixels côtiers.

La production de cet indice spatialisé de vulnérabilité côtière est semi-automatisée et repose sur l'utilisation d'un algorithme développé sous Python et QGIS, dont les étapes détaillées sont présentées en figure 7 (voir annexe pour le détail des commandes utilisées).

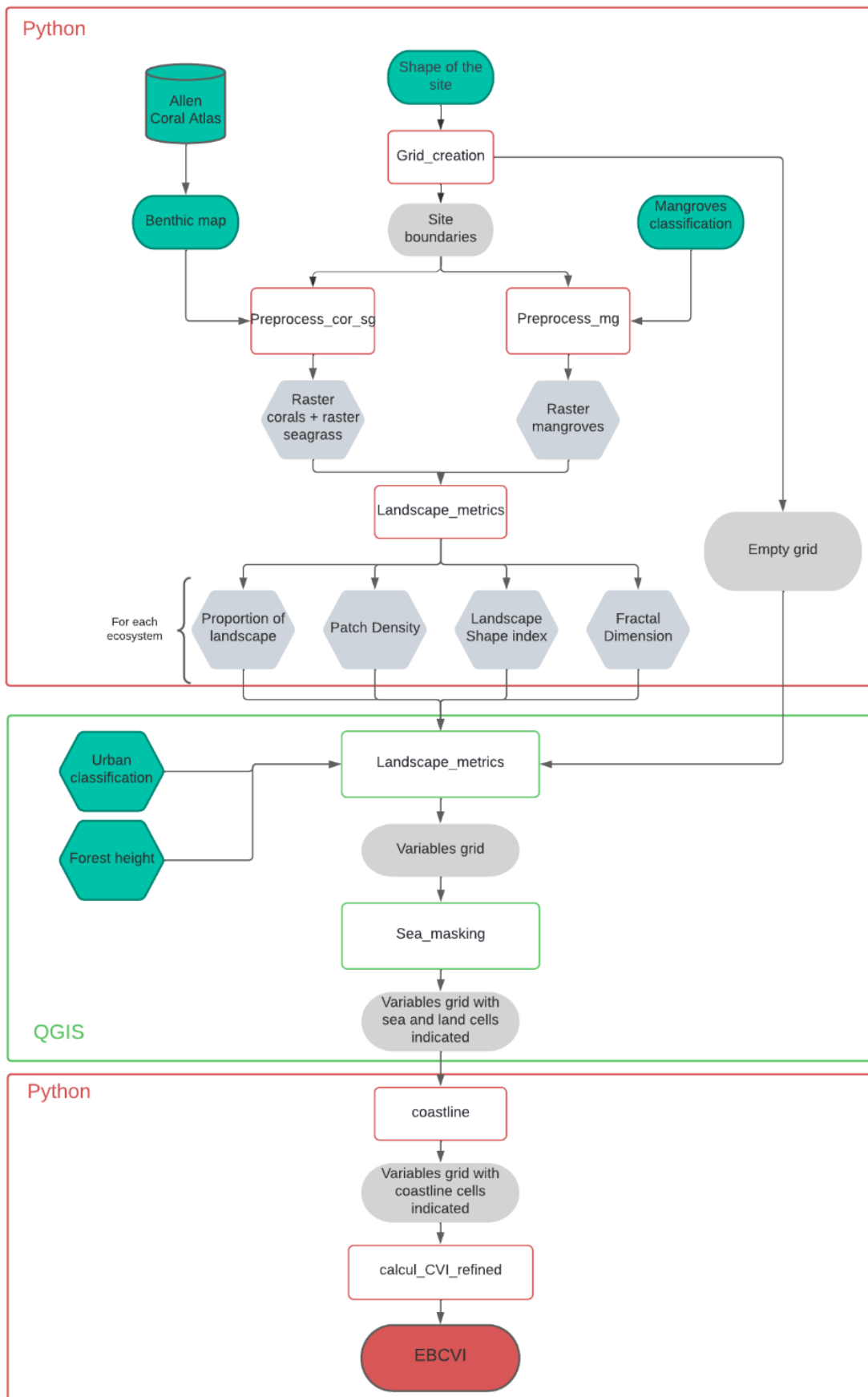


Figure 7 : Détail des étapes de la méthode de production de l'EBCVI, basée sur un algorithme développé sous Python et QGIS.

Résultats

L'utilisation des produits issus de l'imagerie satellite ainsi que des variables descriptives de la configuration du paysage permettent la spatialisation de l'indice de vulnérabilité côtière caractérisant le rôle joué par les écosystèmes littoraux dans la protection contre l'érosion.

Les figures 8, 9 et 10 présentent les principales variables produites lors des étapes initiales du calcul de l'EBCVI.

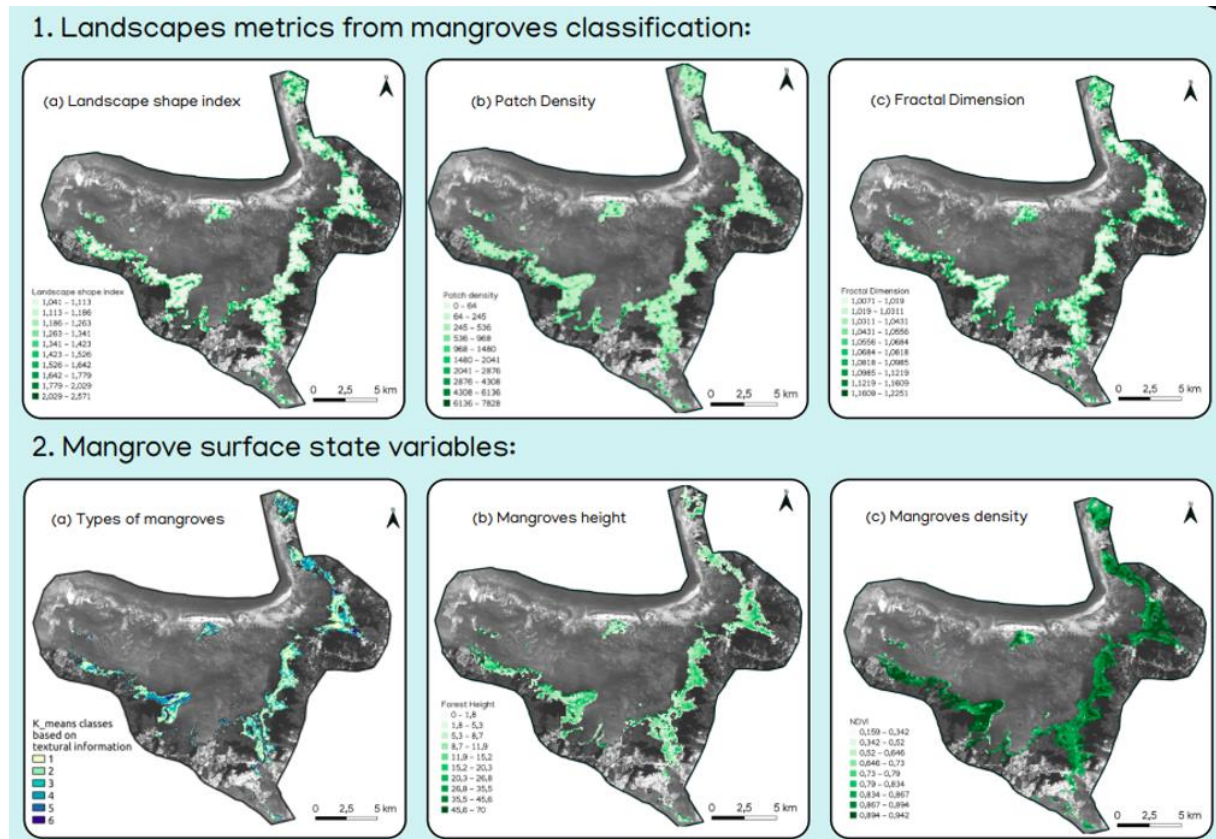


Figure 8 : Les différentes variables issues des métriques paysagères décrivant la configuration de la mangrove utilisées pour la production de l'EBCVI à Cul-de-Sac (Guadeloupe).

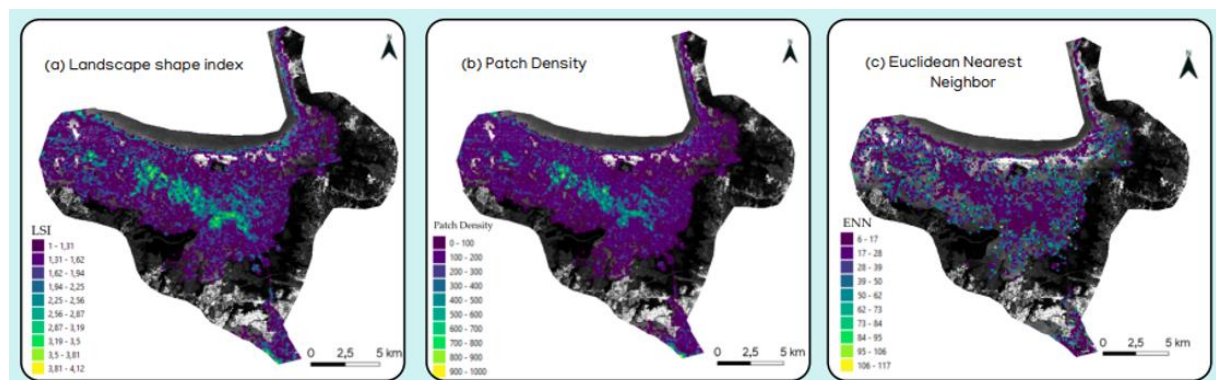


Figure 9 : Les différentes variables issues des métriques paysagères décrivant la configuration des herbiers et des coraux utilisées pour la production de l'EBCVI à Cul-de-Sac (Guadeloupe).

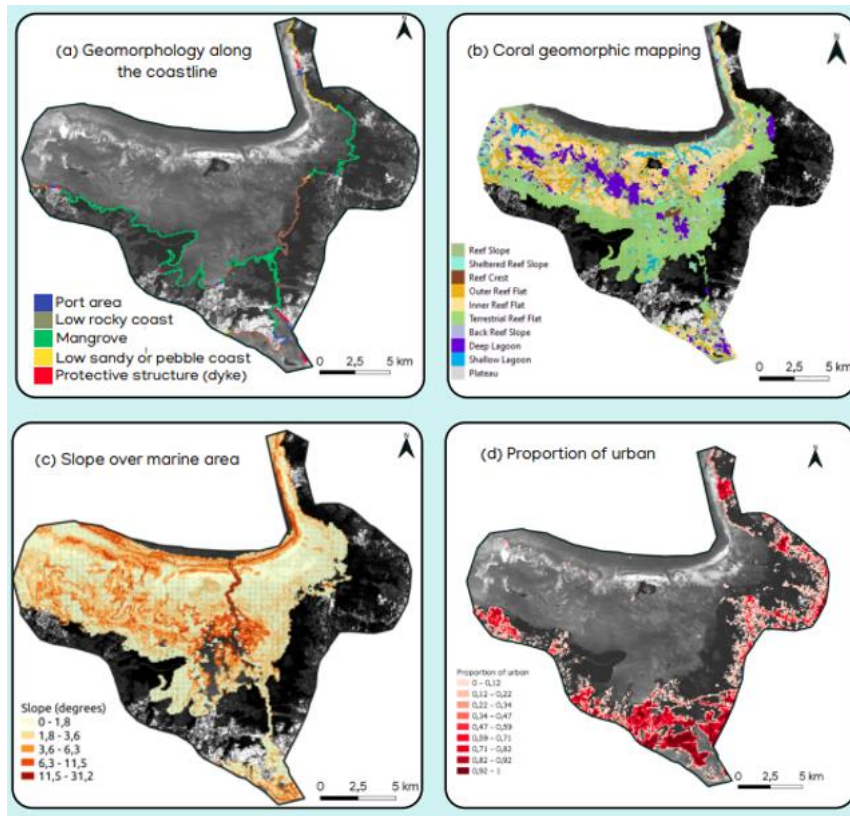


Figure 10 : Autres variables décrivant la configuration de la zone d'étude pour la production de l'EBCVI à Cul-de-Sac (Guadeloupe).

Cet indice est calculé suivant la combinaison linéaire présentée en figure 5, puis normalisé. Les valeurs minimales correspondant à une faible vulnérabilité côtière (forte protection de la côte par les écosystèmes littoraux) et les valeurs maximales à une forte vulnérabilité côtière (faible protection de la côte par les écosystèmes littoraux). La figure 11 représente l'EBCVI sur la zone de Cul-de-Sac (Guadeloupe)

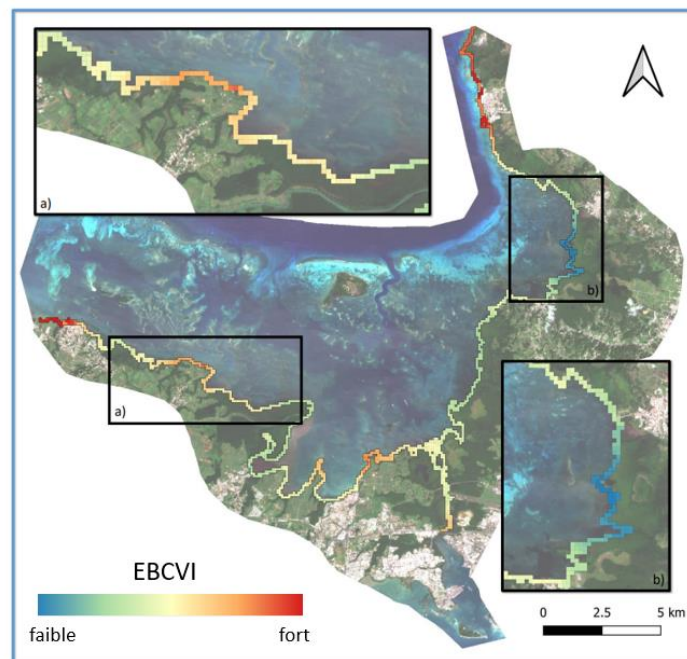


Figure 11 : Résultat du calcul de l'EBCVI, l'indice de vulnérabilité côtière caractérisant le rôle des écosystèmes littoraux dans la protection contre l'érosion, à Cul-de-Sac (Guadeloupe).

Les cartes d'EBCVI sur l'ensemble des sites pilote du projet sont présentées ci-dessous (figure 12 à 18), et sont visualisables sur le portail numérique du projet CARIBCOAST.

Guadeloupe

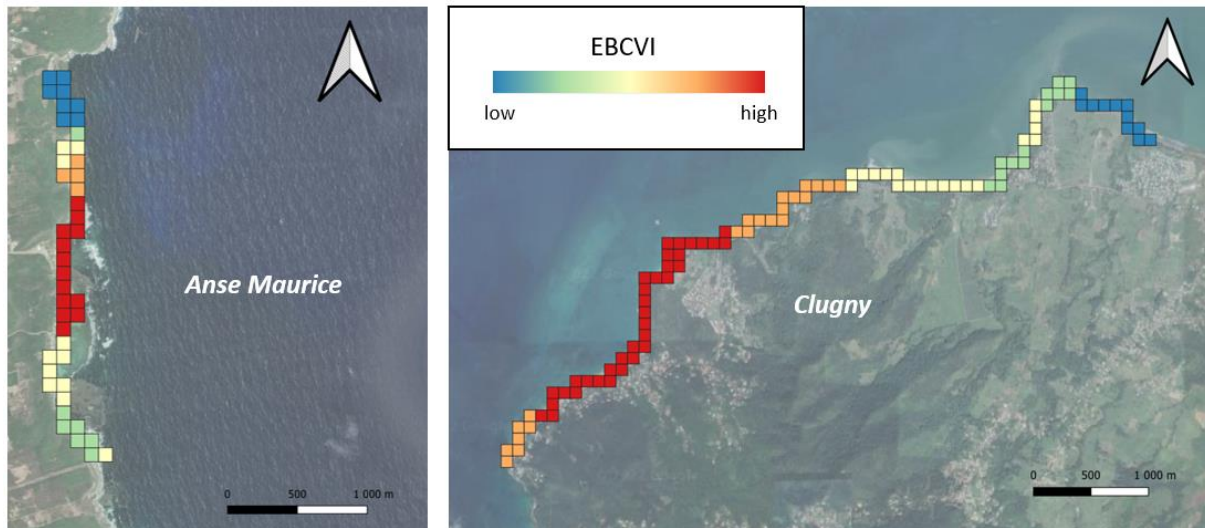


Figure 12 : EBCVI sur les sites pilotes d'Anse Maurice et Clugny (Guadeloupe). Voir figure 10 pour le site de Cul de Sac.

Jamaïque

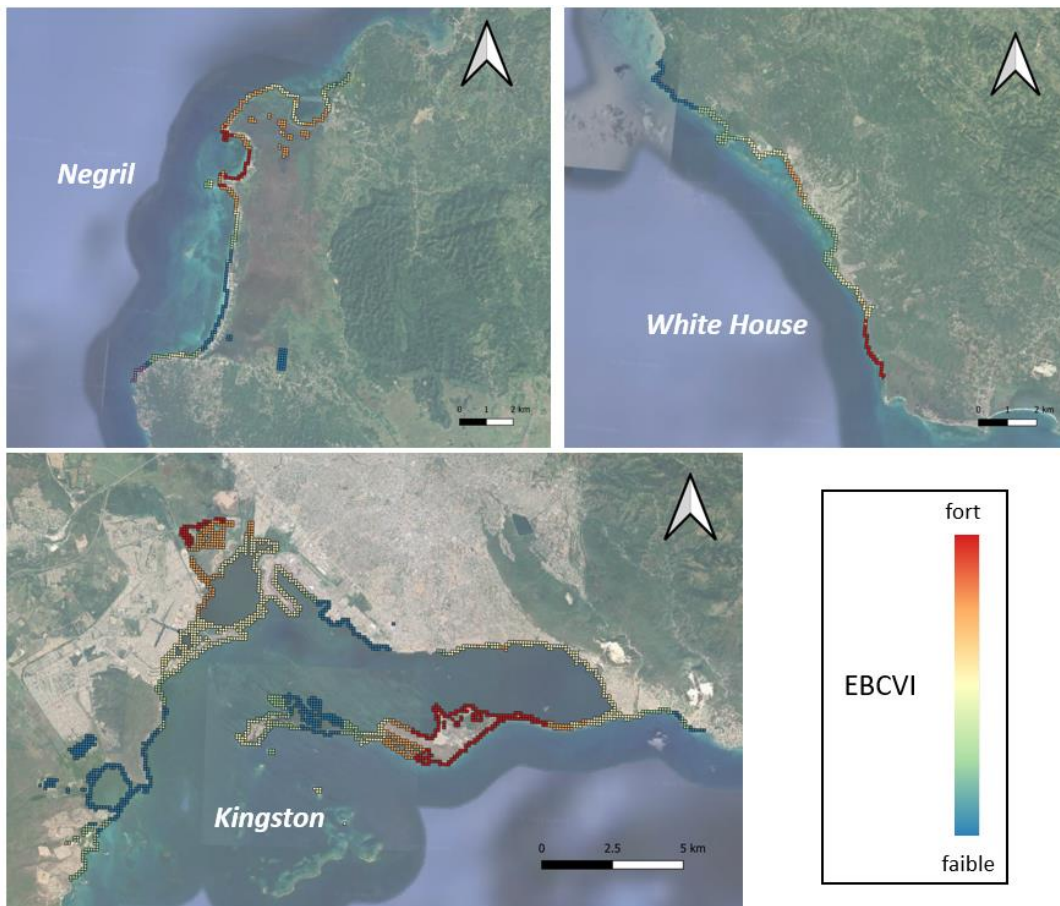


Figure 13 : EBCVI sur les sites pilotes de Jamaïque, Kingston, White House et Negril.

Martinique

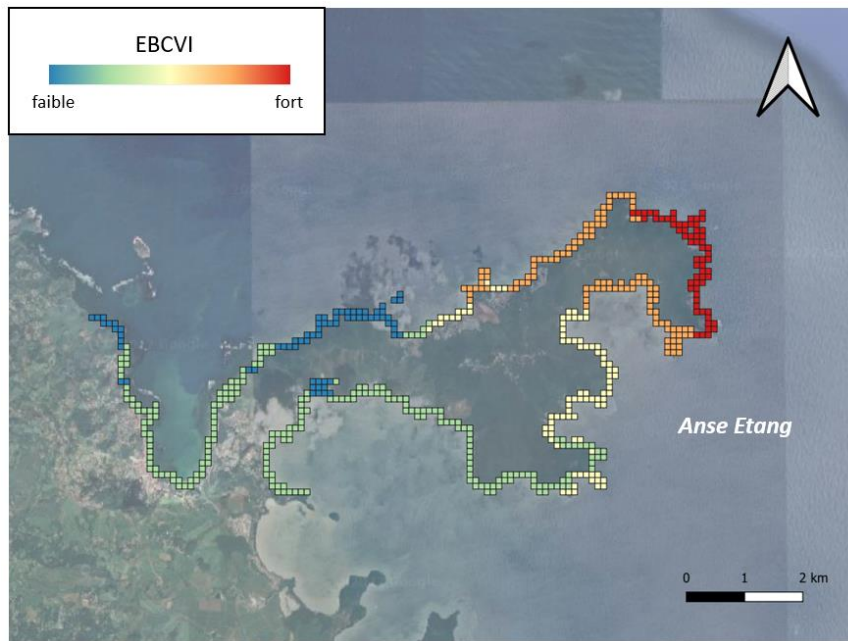


Figure 14 : EBCVI sur le site pilote d'Anse-Etang (Martinique).

Porto-Rico

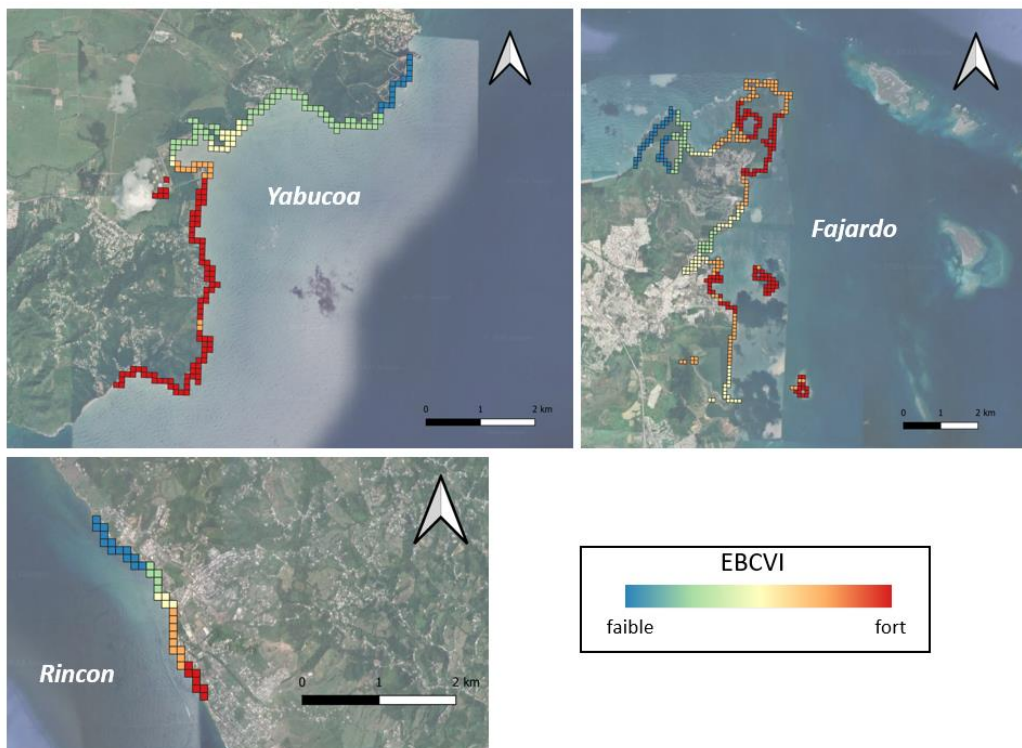


Figure 15 : EBCVI à Porto Rico sur les sites de Yabucoa, Rincon et Fajardo.

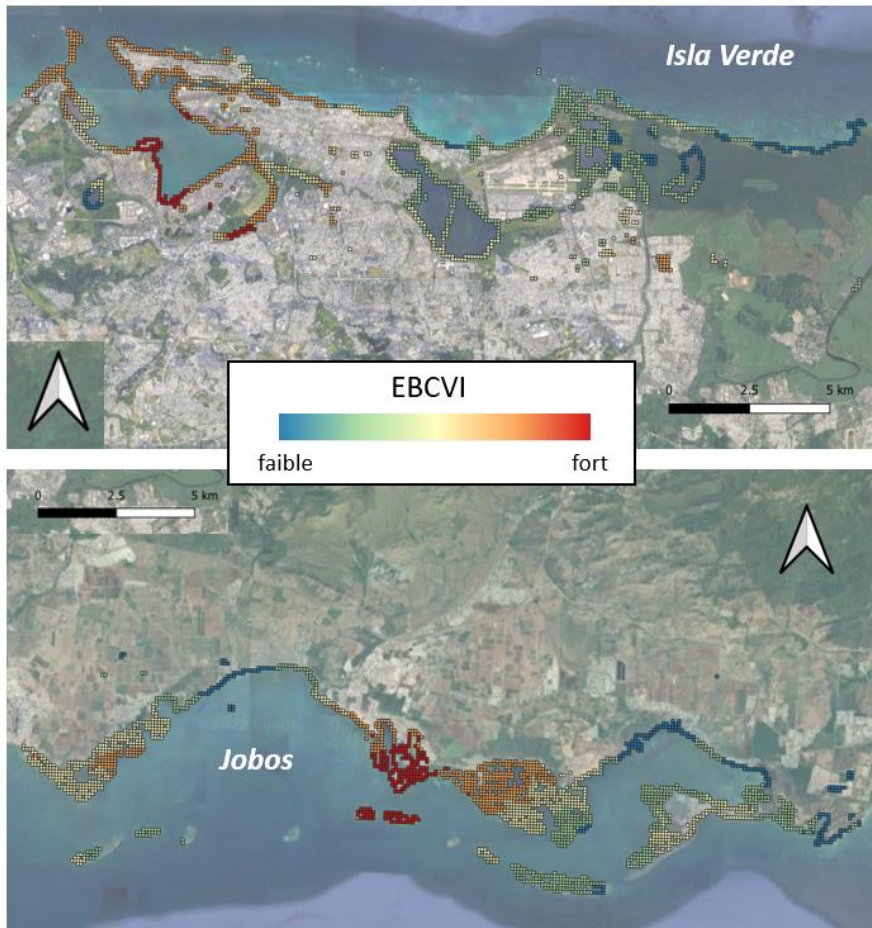


Figure 16 : EBCVI à Porto Rico sur les sites d'Isla Verde et Jobos.

Trinidad et Tobago

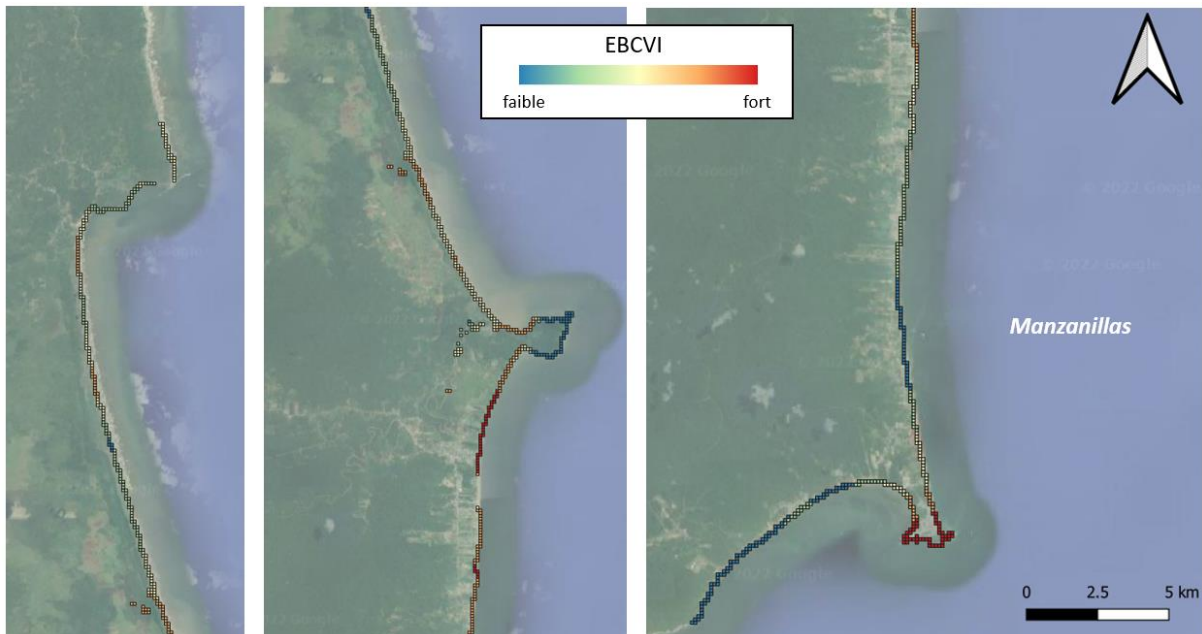


Figure 17 : EBCVI à Trinidad et Tobago sur les sites de Manzanillas.

Saint-Martin

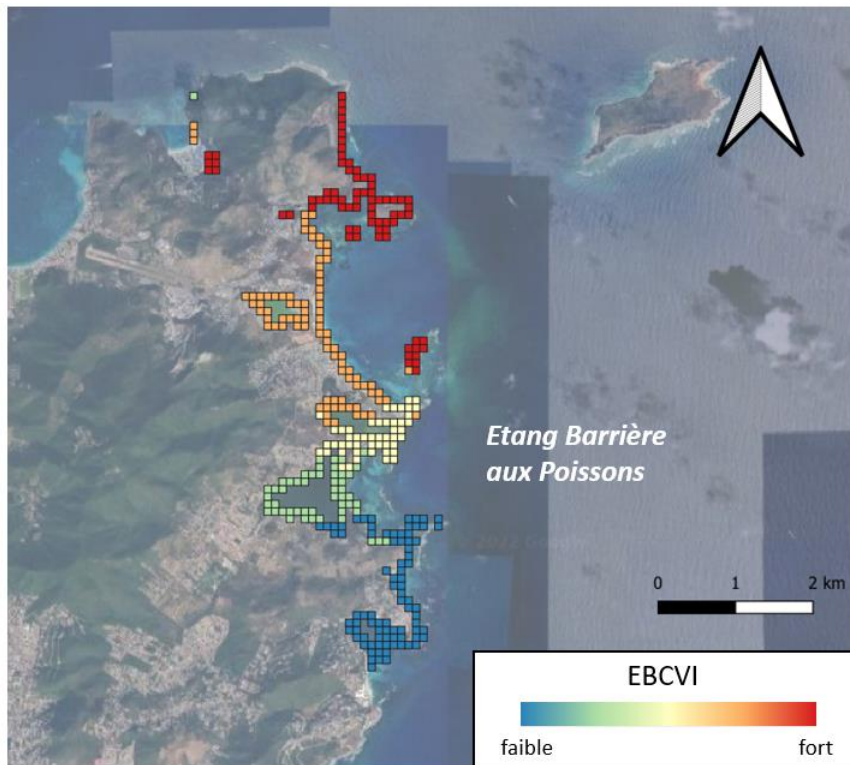


Figure 18 : EBCVI à Saint-Martin sur le site de l'Etang Barrière aux Poissons.

Références

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>

Annexe

Les codes permettant le calcul de l'EBCVI sont présentés ci-dessous.

Prétraitement des produits issus des données d'observation de la Terre (mangroves), herbiers et coraux)

```
import geopandas as gpd
from geocube.api.core import 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.tif"
#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)

```

Prétraitement des produits issus des données d'observation de la Terre (herbiers et coraux)

```

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)

```

Calcul des variables (métriques paysagères) à partir des données prétraitées et du module Pylandstats

```

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")

```

Génération de la grille de 100x100 m et moyenne des variables sur la grille

```

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 pyrasta.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')

```

Pondération et attribution des valeurs des variables au pixel de trait de côte

```

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')

```

Calcul de l'EBCVI normalisation par combinaison linéaire pondérée des variables précédentes

```

import geopandas as gpd
import pandas as pd
import numpy as np
import jenkspsy #https://github.com/mthh/jenkspsy
#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')

```